BAB II

TINJAUAN PUSTAKA

2.1 Sistem Visi Komputer

Sebelum menuju materi yang lebih dalam kita harus mengetahui hubungan antara komputer *vision*, grafika komputer, pengolahan citra dan pengenalan pola (*pattern recognition*) adalah suatu proses atau rangkaian pekerjaan yang bertujuan mengklasifikasikan data numerik dan simbol (Ahmad, 2005). Komputer *Vision* atau Visi Komputer di definisikan sebagai ilmu pengetahuan yang mempelajari bagaimana komputer dapat mengenali objek yang diamati atau diobservasi (Nusantara, 2005). Komputer *vision* merupakan kombinasi antara pengolahan citra dan pengenalan pola.

2.1.1 Komputer Grafika

Komputer grafik melalui pemrograman grafik menghasilkan citra dari bentuk goemetri yang *primitive* seperti titik, garis lurus dan garis lengkung. Lingkaran dan bentuk-bentuk dasar geometri lainnya. Komputer grafik memainkan peranan penting dalam visualisasi dan *virtual-reality*. Sedangkan sistem visual bekerja sebaliknya, menduga bentuk geomentri primitive dan ciri lainnya yang merupakan penyederhanaan dari citra asal yang sifatnya lebih komples. Jadi komputer grafik memadukan unsur-unsur pembentuk citra untuk membentuk atau mensintesa citra sedangkan sistem visual mengerjakan hal

kebalikannya menjadi bentuk-bentuk yang lebih sederhana, agar dapat dinilai secara kuantitatif.

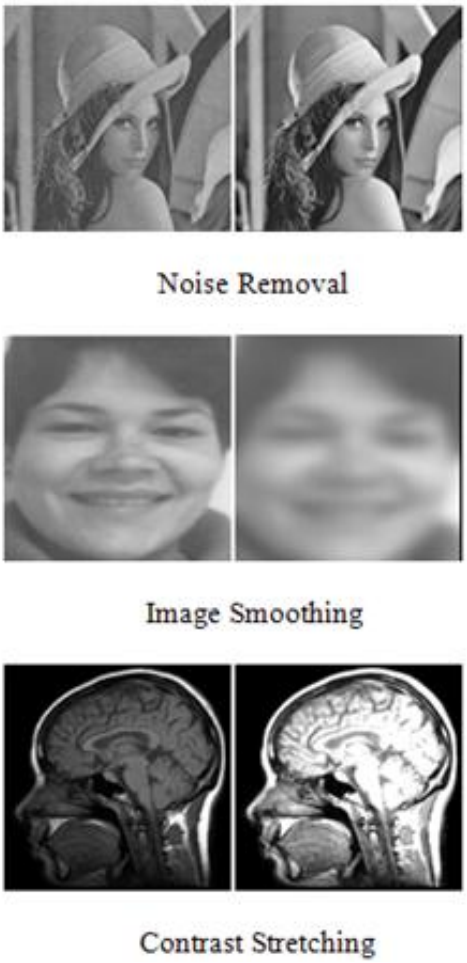
Pada jaman dahulu kala, ternyata manusia sudah mengunakan media gambar untuk saling berkomunikasi dengan manusia lainnya. Hal ini terbukti dengan ditemukannya peninggalan bersejarah berupa lukisan-lukisan bangsa aborigin yang tinggal di Australia yang terdapat di goa-goa dinding batu. Lukisan atau gambar-gambar tersebut mengilustrasikan tentang keharmonisan alam dan lingkungan bangsa aborigin.

Pada jaman modern setelah komputer ditemukan, komputer grafik tidak berkembang dengan pesat, hal itu disebabkan harga dari hardware komputer yang tidak terjangkau, sehingga pemakai komputer pada saat itu hanya terbatas pada kalangan militer saja. Tetapi pada waktu berikutnya harga *hardware* semakin terjangkau sehingga masyarakat umum dapat memakai komputer untuk kebutuhan sehari-hari, maka pemrograman grafik juga ikut berkembang. Perkembangan komputer grafik sangat terasa setelah berbagai bahasa tingkat tinggi seperti C dan *Pascal* ditemukan, dengan bantuan *compiler* seperti ini maka masyarakat umum bisa berkreasi tanpa dibatasi untuk selalu menciptakan yang baru (Heriady, 2007).

2.1.2 Pengolahan Citra Digital

Image Processing atau pengolahan citra adalah bidang tersendiri yang sudah cukup berkembang sejak orang mengerti bahwa komputer tidak hanya dapat menangani data teks, tetapi juga data citra (Ahmad, 2005). Teknik-teknik

pengolahan citra bisaanya dgunakan untuk melakukan transformasi dari satau citra kepada citra yang lain, sementara tugas perbaikan informasi terletak pada manusia melalui penyusunan algoritmanya. Bidang ini meliputi penajaman citra, penonjolan fitur tertentu dari suatu citra, kompresi citra dan koreksi citra yang tidak forkus atau kabur. Sebaliknya, sistem *visual* menggunakan citra sebagai masukan tetapi menghasilkan keluaran jenis lain seperti representasi dari kontur obyek di dalam citra, atau menghasilkan gerakan dari suatu peralatan mekanis yang terintegrasi dengan sistem visual.



Gambar 2.1 Hasil pengolahan citra digital

Berkat adanya mata sebagai indera penglihatan yang sangat penting dalam kehidupan sehari-hari, manusia dapat melakukan banyak hal dengan lebih mudah. Berbagai aktifitas seperti berjalan, mengambil sesuatu benda, menulis, apalagi membaca buku, menjadi sangat mudah dilakukan bila melibatkan fungsi mata. Peristiwa melihat yang begitu sederhana bagi kita dalam kehidupan sehari-hari ternyata melibatkan banyak proses dan aliran data yang besar. Dengan menggunakan sifat-sifat seperti halnya mata, maka hal di atas dapat diaplikasikan dalam perangkat keras pengolahan citra seperti *Webcam*, *handycam*, *camera* digital, *scanner* dan lain-lain.

2.1.3 Sistem Visual

Sebagaimana layaknya mata dan otak, sistem visual buatan atau *vision system* (komputer vision) adalah suatu sistem yang mempunyai kemampuan untuk menganalisis objak secara visual, setelah data objek yang bersangkutan dimasukkan dalam bentuk citra (*image*). Secara umum tujuan dari sistem visual adalah untuk membuat model nyata dari sebuah citra. Citra yang dimaksud di sini adalah citra digital hasil konversi suatu objek menjadi citra melalui suatu sensor yang prosesnya disebut digisi. Dengan demikian citra lain seperti foto, gambar cetak, gambar sketsa, dan lain-lain yang berada pada media cetak seperti kertas atau media lainnya, setelah dikonversi ke dalam citra digital baru dapat diproses melalui poses digitasi.

Sebagai sistem visual mempunyai kemampuan untuk memperbaiki informasi yang berguna dari sebuah pemandangan (*scene*) hasil proyeksi dua

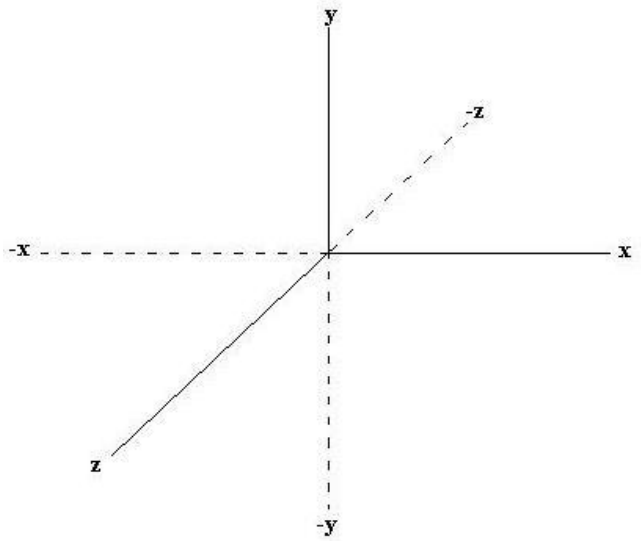
dimensi. Citra merupakan hasil proyeksi dua dimensi dari obyek atau benda tiga dimensi, sehingga informasi sesungguhnya ada bagian informasi yang hilang setelah benda diproyeksikan ke dalam citra. Untuk memperbaiki informasi diperlukan pengetahuan dan proyeksi geometri dari obyek dalam suatu pemandangan. Contoh yang mudah dipaparkan dalam hal ini misalnya jarak antara sensor (bisanya kamera) dengan obyek tertentu dalam sebuah citra tidak dapat diketahui langsung karena informasi kedalaman (*depth*) yang merupakan jarak itu sendiri hilang saat obyek dalam ruang (tiga dimensi) dikonversi menjadi citra dalam bidang dua dimensi. Dengan demikian bila ada dua atau lebih obyek yang bertumpukan dalam suatu bidang pandang, suatu obyek tidak dapat langsung diketahui apakah ia berada di depan atau di belakang obyek lainnya. Namun demikian persoalan ini dapat dijawab setelah dilakukan satu atau bebrapa proses pada citra tersebut dengan memanfaatkan informasi yang tersedia sehingga informasi yang hilang tadi didapatkan kembali.

Pengertian pengolahan citra (*image processing*) sedikit berbeda dengan pengertian mesin visual (*machine vision*), meskipun keduanya seolah-olah dapat dipergunakan dengan maksud yang sama. Terminologi pengolahan citra dipergunakan bila hasil pengolahan data yang berupa citra, adalah berbentuk citra yang lain, yang mengandung atau memperkuat informasi khusus pada citra hasil pengolahan sesuai dengan tujuan pengolahannya. Sedangkan terminologi mesin visual digunakan bila data hasil pengolahan citra langsung diterjemahkan dalam bentuk lain, mislanya grafik yang siap diinterpretasikan untuk tujuan tertentu, gerak peralatan atau bagian dari peralatan mekanis atau aksi lainnya yang berarti

bukan merupakan citra lagi. Dengan demikian jelaslah bahwa pengolahan citra merupakan bagian dari mesin visual, karena untuk menghasilkan keluaran selain citra, informasi dari citra yang ditangkap oleh kamera juga perlu diolah dan dipertajam pada bagian-bagian tertentu (Ahmad, 2005).

2.1.4 Sistem Koordinat 3D

Sistem koordinat tiga dimensi memiliki tiga sumbu yaitu x, y, z dan setiap sumbu memiliki nilai yang berbeda-beda, bagian dari koordinat yang bernilai negatif digambarkan dengan garis putus-putus, sedangkan bagian yang bernilai positif digambarkan dengan garis lurus berkesinambungan(tidak putus-putus).



Gambar 2.2 Sistem koordinat 3D

2.2 Opengl

OpenGL adalah suatu *graphics library* yang sebagian bersifat *open source*, dapat dipakai pada banyak *platform* (*window* s, linux) dan dapat digunakan pada berbagai jenis compiler seperti C++ dan Delphi. File DLL yang diperlukan adalah

glu32.dll, glut32.dll dan opengl32.dll. file header yang diperlukan adalah gl.h, glaux.h, glu.h dan glut.h. Sedangkan file lib yang dibutuhkan adalah glaux.lib, glu32.lib, glut.lib, glut32.lib, dan opengl32.lib (Heriady, 2007). Berikut ini adalah table-tabel perintah dalam OpenGL:

Tabel 2.1 Fungsi OpenGL

Fungsi / perintah dalam OpenGL	**Kegunaan**
glPointSize(Glfloat Size);	Menentukan besar titik
glBegin(GL_POINTS) ... glEnd();	Menggambar titik
glColor3f(...);	Menentukan warna
glVertex3f(...);	Menentukan posisi sebuah titik
glTranslatef(...);	Mengubah posisi titik pusat sumbu koordinat
glPushMatrix(); .. glPopMatrix()	Membuat baris kode diantaranya menjadi tidak berlaku untuk bagian luar.
glLineWidth(Glfloat Width);	Menentukan lebar garis
glBegin(GL_LINES); ... glEnd();	Menggamabar garis
glBegin(GL_LINE_STRIP); ... glEnd();	Menggambar garis terhubung
glBegin(GL_LINE_LOOP); ... glEnd();	Menggambar garis terhubung
glBegin(GL_TRIANGLES); ... glEnd();	Menggambar segitiga
glBegin(GL_TRIANGLE_STRIP); ... glEnd();	Menggambar segitiga terhubung
glBegin(GL_TRIANGLE_FAN); ... glEnd();	Menggambarsegitiga terhubung

Setiap sistem operasi memiliki cara yang berbeda-beda dalam cara penginstalan OpenGL disini hanya akan membahas cara *instal* OpenGL dengan menggunakan sistem operasi *Window* XP. Agar OpenGL bisa digunakan dalam sistem operasi terutama *window* berikut adalah cara menginstal OpenGL:

1) Ada tiga file penting opengl yaitu glut.h, glut32.lib dan glut32.dll.

2) Letakan glut.h di direktori include pada visual C (vc98/include/GL)

3) Letakan glut32.lib di direktori lib pada visual C (vc98/lib)

4) Letakan glut32.dll di direktori *window* s/system.

Pada opengl untuk memulai suatu pemrograman grafik amat sangatlah sederhana, pembaca hanya memerlukan empat fungsi yang telah disediakan oleh opengl (Ahmad, 2005), fngsi-fungsi tersebut adalah:

1. glutInit*Window* size

fungsi ini untuk membuat *window* yang akan digunakan untuk men-*display* objek-objek 3D, format dari fungsi ini adalah: void glutInit*window* size (init *width*, int *height*), *width* adalah lebar dari *window* yang akan dibuat (dalam *pixel*). Sedangkan *height* adalah tinggi dari *window* (dalam *pixel*).

2. glutInit

fungsi untuk memberitahu MS VC++ bahwa opengl *library* dipakai pada program. Formatnya:

void glutInit(int *argcp, char **argv);

Argcp dan argv adalah parameter identik yang dipakai oleh fungsi main() pada C++, jadi pastikan bahwa parameter ini sama dengan parameter yang ada pada fungsi main() pada setiap program grafik 3D.

3. glutInitDisplayMode

Fungsi yang dipakai untuk menginisialisasi model dari tampilan (*display mode)*, yang mempunyai format:

void glutInitDisplayMode(unsigned int mode);

variable mode diisi dengan:

GLUT_DOUBE|GLUT_RGB|GLUT_DEPTH.

Pilihan untuk variable mode amat sangat bervariasi, tetapi pada buku ini semua program hanya akan menggunakan tiga kombinasi di atas. GLUT_DOUBLE berarti menggunakan *buffer* pada *window* sebesar dua kali. GLUT_RGB menggunakan *Red Green Blue Alpha* untuk pewarnaan. GLUT_DEPTH berarti menggunakan *depth buffer* agar objek 3D yang ditampilkan akan terlihat lebih nyata. Untuk mengkombinasikan *variable mode* yang dipakai gunakan garis vertikal (|).

4. GlutCreate*window*

Fungsi ini dipakai untuk membuat/meng-create *window* , mempunyai format sebagai berikut:

Int glutCreateWindow (char name);*

Name adalah nama dari *window* yang akan dibuat. Dengan memakai empat fungsi tersebut para pembaca telah berhasil melakukan inisialisasi pada opengl. Masih banyak lagi fungsi-fungsi yang disediakan oleh opengl, tetapi para pembaca cukup menggunakan empat fungsi tersebut saja. Berikut ini adalah sebagian dari isi fungsi main untuk menginisialisasi opengl.

```
Int main(int argc, char**argv)
{
Int L_window =750, T_window  =500;
….
….
//inisialisasi opengl
```

```
Printf(“inisialisasi Opengl \n);
glutIntitWindow size (L_window , T_window );
glutInit (&argc, argv);// parameter sama dengan yang di main
glutinitdisplaymode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
glutcreatewindow  ("Belajar Grafik 3D”);
….
….
}
```

Untuk dapat menset tampilan opengl telah menyediakan fungsi gluPersvective yang mempunyai format:

Void gluPerspective(float fovy, float aspect, float znear, float zfar);

Fovy adalah sudut pandang terhadap sumbu Y, *aspect* adalah rasio perbandingan anatar lebar dan tinggi objek, zNear adalah titik dekat dan zFar adalah titik jauh (keduanya harus bernilai positif). Fungsi lain yang dipakai untuk mendukung fungsi gluPerspective adalah fungsi glMatrixMode, fungsi ini berguna untuk mengubah model *matrix* yang sedang aktif. Fungsi ini mempunyai format:

Void glMatrixmode(GLenum mode);

Untuk mengaktifkan matrix proyeksi (*projection matrix*) maka *mode* diisi dengan GL_PROJECTION, sedangkan utuk mengaktifkan matrix pandangan mode diisi dengan GL_MODELVIEW. Beriktu ini adalah penggalan program untuk *setting* tampilan di layar monitor:

```
Int main (int argc, char**argv)
{
……
glMatrixmode (GL_PROJECTION);
//(float) L_window  berarti merubah tipe data dari int ke float
glPerspective (45.0, (float) L_window /(float) T_window , 1.0f, 200.0f);
```

```
glMatrixmode(GL_MODELVIEW);
……
}
```

Untuk dapat menampilkan objek 3D pada layar monitor hanya dibutuhkan fungsi glutDisplayFunct yang mempunyai format:

Void glutDisplayFunc(void (*func) (void));

Dengan format sperti itu maka para pembaca harus membuat suatu fungsi yang bertipe void agar dapat dipanggil oleh fungsi glutDisplayFunct, di dalam fungsi bertipe void tersebutlah objek 3D harus diletakan. Berikut ini adalah pengglaan program cara untuk mendisplay objek 3D:

```
Int main (int argc, char ** argv)
{
……
glutDisplayFunct(glut_display_F);
……
}
Void glut_display_F()
{
Glclear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentitiy();
……
//object 3D di letakan di sini
……
glutSwapBuffers();
}
```

Pada fungsi void_display_F() sebelum memulai untuk menampilkan objek 3D harus dilakukan pembersihan *buffer* (*clearbuffer*) lalu memanggil *matrix identitiy*/*matrix* identitas dan setelah itu *buffer* yang telah dibersihkan diisi dengan objek 3D yang hendak ditampilkan, untuk melakukan tiga tugas ini OpenGL menyediakan fungsi glClear, glLoadIdentitiy dan glutswapBuffers, yang masing-masing mempunyai format:

Void glclear(GLbitfield mask); Mask diisi dengan GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT.

GL_COLOR_BUFFER_BIT dipakai untuk membersihkan *buffer* yang dipakai untuk pewarnaan, sengakan GL_DEPTH_BUFFER_BIT dipakai untuk membersihkan *buffer depth* (ingat glutInitDisplayMode).

Void glLoadIdentitiy (void);

Void glutSwapBuffers(void);

Setiap program grafik membutuhkan waktu tunda (*delay*) agar tampilan grafik yang dihasilkan dapat dimanipulasi sehingga sedap untuk dipandang oleh *user* yang menggunakan program tersebut.

Waktu tunda adalah selang/jeda waktu yang ditentukan oleh *programmer* agar program yang dibuat terhenti sesaat, setelah selang/jeda waktu habis maka program akan berjalan kembali.

Dengan memanfaatkan waktu tunda maka tampilan grafik dapat dibuat animasi. OpenGL telah menyediakan fungsi untuk waktu tunda tetapi berapa lama waktu tunda yang dikehendaki harus disetting sendiri oleh programmer. Fungsi yang disediakan oleh OpenGL untuk waktu tunda adalah glutIdleFunc yang mempunyai format sebagai berikut:

Void glutIdleFunc(void (*func) (Void));

Sedangkan untuk setting berapa lama waktu yang dikehendaki dapat dipakai cara kuno tetapi efektif, yaitu seperti contoh dibawah ini:

```
Waktu_awal=timeGetTime();
Do
{
Waktu_sekarang = timeGetTime();
}
While ((wktu_sekarang – waktu_awal) <= waktu_selisih);
Berikut ini adalah penggalan dari program grafik 3D untuk setting waktu tunda:
Int main(int argc, char **argv)
{
…….
glutIdleFunct(glut_idle_F);
…….
}
Void glut-idle_F()
{
Waktu_awal=timeGetTime();
Do
{
Waktu_sekarang = timeGetTime();
}
While ((wkatu_sekarang – waktu_awal) <= waktu_selisih);
…….
glutPostRedisplay();
}
```

Fungsi glutPostRedisplay berguna untuk menampilkan kembali objek 3D yang ada di fungsi glut_display_F. Format dari fungsi glutPostRedisplay adalah Void glutPostRedisplay(void);

2.3 *Augmented Reality*

Istilah *Augmented reality* (AR) tidak diterjemahkan karena belum ditemukan padanan kata yang sesuai, dalam glosarium pusat bahasa diknas, *reality*=realitas, *augment*=augmen. Secara umum, AR adalah penggabungan antara obejek virtual dengan objek nyata (Wikipedia, 2009). Sebagai contoh, adalah saat stasiun televisi, menyiarkan pertandingan sepak bola, terdapat objek virtual, tentang skor pertandingan yang sedang berlangsung.

Menurut Ronald Azuma pada tahun 1997, AR adalah menggabungakan dunia nyata dan virtual, bersifat interaktif secara *real time*, dan merupakan animasai 3D. Paul Milgram dan Fumio Kishino pada tahun 1994, mendefinisikan *Milgram's Reality-Virtuality Continuum*. Mereka menggambarkan sebuah kontinum yang membentang dari lingkungan nyata untuk untuk lingkungan *virtual* murni. Mereka menyimpulkan bahwa AR lebih dekat ke lingkungan nyata dan *augmented virtuality* lebih dekat ke lingkungan *virtual.*

AR yang dalam bahasa Indonesia diterjemahkan sebagai realitas tertambah atau aplikasi realitas yang diperluas, adalah teknologi yang menggabungkan benda maya dua dimensi dan ataupun tiga dimensi ke dalam sebuah lingkungan nyata tiga dimensi lalu memproyeksikan benda-benda maya tersebut dalam waktu nyata (Wikipedia, 2009).

Dalam pembuatan AR ini menggunakan *Webcam* sebagai perangkat untuk menangkap citra. Sebelum citra diubah ke dalam bentuk digital maka proses manipulasi citra digital tidak bisa dilakukan. Citra digital (f(x,y))mempunyai dua unsur. Unsur yang pertama merupakan kekuatan sumber cahaya yang melingkupi pandangan kita terhadap objek (*illumination*). Unsur yang kedua merupakan besarnya cahaya yang direfleksikan oleh objek ke dalam pandangan mata kita atau disebut juga *reflectance components*. Kedua unsur tersebut dituliskan sebagai fungsi i(x,y) dan r(x,y).

2.3.1 Sejarah *Augmented Reality*

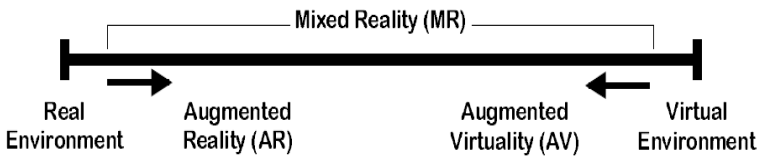
Sejarah tentang *Augmented Reality* (AR) dimulai dari tahun 1957-1962, ketika seorang penemu yang bernama Morton Heilig, seorang sinematografer, menciptakan dan mempatenkan sebuah *simulator* yang disebut Sensorama dengan *visual*, getaran dan bau. Pada tahun 1966, Ivan Sutherland menemukan *head-mounted display* yang dia klaim adalah, jendela ke dunia virtual. Tahun 1975 seorang ilmuwan bernama Myron Krueger menemukan *Videoplace* yang memungkinkan pengguna dapat berinteraksi dengan objek *virtual* untuk pertama kalinya. Tahun 1989, Jaron Lanier, memeperkenalkan *Virtual Reality* dan menciptakan bisnis komersial pertama kali di dunia maya, Tahun 1992 mengembangkan AR untuk melakukan perbaikan pada pesawat boeing, dan pada tahun yang sama, LB Rosenberg mengembangkan salah satu fungsi sistem AR, yang disebut *Virtual Fixtures*, yang digunakan di Angkatan Udara AS Armstrong Labs, dan menunjukan manfaatnya pada manusia, dan pada tahun 1992 juga, Steven Feiner, Blair Maclntyre dan dorée Seligmann, memperkenalkan untuk pertama kalinya *Major Paper* untuk perkembangan *Prototype* AR. Pada tahun 1999, Hirokazu Kato, mengembangkan ArToolkit di HITLab dan didemonstrasikan di SIGGRAPH, pada tahun 2000, Bruce. H. Thomas, mengembangkan ARQuake, sebuah mobile games AR yang ditunjukan di international *symposium on wearable komputers*. Pada tahun 2008, witiude AR Travel Guide, memperkenalkan Android G1 telephone yang berteknologi AR, tahun 2009, Saqoosha memperkenalkan FLARToolkit yang merupakan perkembangan dari ArToolkit. FLARToolkit memungkinkan kita memasang

teknologi AR di sebuah *website*, karena *output* yang dihasikan FLARToolkit berbentuk Flash. Ditahun yang sama, *wikitude Drive* meluncurkan sistem navigasi berteknologi AR di *platform android*. Tahun 2010, *Acrossair* menggunakan teknologi AR pada I-Phone 3GS.

2.3.2 *Augmented Reality* dan *Virtual Reality*

Virtual reality, adalah teknologi tentang *spectrum* yang luas. Ada tiga definisi tentang *virtual reality*, yang pertama *virtual reality* adalah komputer yang menghasilkan *output* 3D dengan grafis yang tinggi. Definisi yang kedua adalah, *virtual reality* adalah dunia 3D yang interaktif, karena seorang pengguna bisa berinteraktif dengan komputer secara *real time*. Definisi yang ketiga adalah, *virtual reality* adalah dunia maya, yang penggunanya dapat memasuki dunia *virtual* tersebut. Perbedaan dari AR dan *virtual reality* hanya dari cara kerja *immersiveness* sistem. *Virtual reality*, mempunyai lingkungan virtual yang lebih dalam dari AR, karena *virtual reality* mengontrol alam bawah sadar indera manusia. Sebaliknya AR, menggabungkan antar objek nyata dan objek virtual.

Miligram (Miligram and Kishino 1994; Miligram, Takemura et al. 1994) menjelaskan bagaimana hubungan antara AR dan virtual reality.



Gambar 2.3 Miligram's Reality – Virtuality Continuum

Pada gambar 2.3 terlihat bahwa dunia nyata dan dunia virtual berada di antara dua ujung wilayah yang disebut mix reality (AR). AR berada di dekat dunia nyata (*real environment*), dengan presepsi AR lebih cenderung ke dunia nyata. AR merupakan suatu istilah yang diciptakan oleh Miligram untuk mengidentifikasi sistem, yang sebagian besar sistemnya memasukan objek nyata ke dalam objek virtual.

2.3.3 Manfaat Teknologi AR

Bidang-bidang yang pernah menerapkan teknologi AR adalah:

1) Hiburan (*entertainmen)*: Dunia hiburan membutuhkan AR sebagai penunjang efek-efek yang akan dihasilkan oleh hiburan tersebut. Sebagai contoh, ketika sesorang wartawan cuaca memperkirakan ramalan cuaca, dia berdiri di depan layar hijau atau biru, kemudian dengan teknologi AR, layar hijau atau biru tersebut berubah menjadi gambar animasi tentang cuaca tersebut, sehingga seolah-olah wartawan tersebut, masuk ke dalam animasi tersebut. Latihan Militer (*Military Training*): Militer telah menerapkan AR pada latihan tempur mereka. Sebagai contoh, militer menggunakan AR untuk membuat sebuah permainan perang, dimana prajurit akan masuk kedalam dunia game tersebut, dan seolah-olah seperti melakukan perang sesungguhnya
2) *Enginering Design*: Seorang *engineering design* membutuhkan AR untuk menampilkan hasil design mereka secara nyata terhadap klien. Dengan AR klien akan tahu, tentang spesifikasi yang lebih detail tentang desain mereka.

3) *Robotics dan Telerobotics*: Dalam bidang robotika, seorang operator robot, mengunakan pengendari pencitraan visual dalam mengendalikan robot itu. Jadi, penerapan AR dibutuhkan di dunia robot.
4) *Consumer Design*: *Virtual reality* telah digunakan dalam mempromosikan produk. Sebagai contoh, seorang pengembang menggunkan brosur virtual untuk memberikan informasi yang lengkap secara 3D, sehingga pelanggan dapat mengetahui secara jelas, produk yang ditawarkan.
5) Kedokteran (*Medical*): Teknologi pencitraan sangat dibutuhkan di dunia kedokteran, seperti misanya, untuk pengenalan operasi, pengenalan pembuatan vaksin virus, dll. Untuk itu, bidang kedokteran menerapkan AR pada visualisasi penelitian mereka.

2.4 ArToolkit

ArToolkit adalah *software library*, untuk membangun AR. Aplikasi ini adalah aplikasi yang melibatkan *overlay* pencitraan *virtual* ke dunia nyata. Untuk melakukan ini, ArToolkit menggunakan pelacakan video, untuk menghitung posisi kamera yang nyata dan mengorientasikan pola pada kertas *Marker* secara *realtime*. Setelah posisi kamera yang asli telah diketahui, maka *virtual camera* dapat diposisikan pada titik yang sama, dan objek 3D dapat digambarkan diatas *Marker*. Jadi ArToolkit memecahkan masalah pada AR yaitu, sudut pandang pelacakan objek dan interaksi objek *virtual.*

ArToolkit merupakan software library yang dirancang untuk dapat dihubungkan ke dalam program aplikasi. ArToolit membutuhkan webcam yang

sudah terinstall di computer, selain itu juga dibutuhkan marker dan juga harus sudah terinstal DirectX minimal versi 9. Selain itu ada hal yang tidak kalah penting yaitu menyiapkan *file* glut (*Graphic Library Utility Toolkit*). Langkah-langkah dalam pengaturan file glut diantaranya adalah:

1) Download terlebih dahulu ArToolkit di:
   http://artoolkit.sourceforge.net/apidoc
2) Buat folder AR di *drive* C
3) Simpan ArToolkit-2.72-bin-win32.zip dan glut-3.7.6-bin zip ke folder AR.
4) Ekstrak ArToolkit-2.72.1-bin-win.zip ke *drive* C
5) Ekstrak glut-3.7.6-bin.zip pada folder AR.
6) *Copy* file glut32.dll dari folder glut-3.7.6-bin ke dalam folder C:\ArToolkit\bin.
7) Tambahkan *folder* GL di dalam C:\ArToolkit\include kemudian *copy file* glut.h dari *folder* glut-3.7.6-bin ke folder C:\ArToolkit\include\GL.
8) *Copy* file glut32.lib dari folder glut-3.7.6-bin ke *folder* C:\ArToolkit\lib.

ArToolkit menggunakan teknik visi komputer untuk mengkalkulasi sudut pandang kamera nyate ke *marker* yang nyata. Ada lima langkah dalam proses kerja ArToolkit.

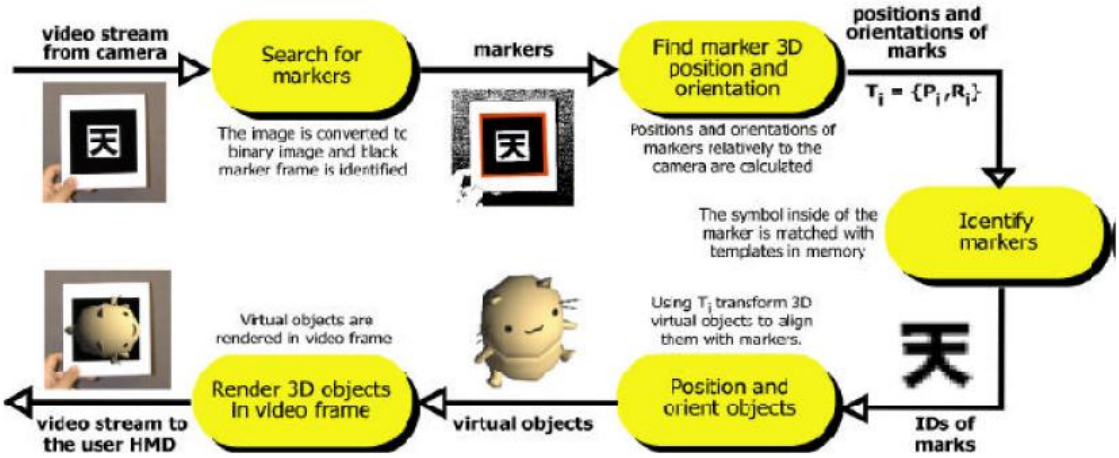
a. Kamera mencari *marker,* kemudian *marker* yang dideteksi diubah menjadi *binary,* kemudian *black frame* atau bingkai hitam terdeteksi oleh kamera.
b. Kamera menemukan posisi *marker* 3D dan dikalkulasikan dengan kamera nyata.

c. Kamera mengidentifikasi *marker*, apakah pola *marker* sesuai dengan *templates memory*.

d. Transformasikan posisi *marker*.

e. Objek 3D di *render* diatas *marker*.

Cara kerja ArToolkit dapat dilihat pada gambar 2.6 dibawa ini.



Gambar 2.4 ArToolkit sebagai pelacakan video

1) Kelebihan ArToolkit

a. ArToolkit dapat menempatkan objek 3D yang dihasilkan komputer sehingga seolah-olah sudut pandang berada pada dunia nyata.

b. ArToolkit bersifat *opensources* sehingga dalam pengembangannya dapat dilakukan dengan mudah.

c. Menggunakan bahasa tingkat tinggi yaitu bahasa C.

d. ArToolkit menggunakan teknik *computer vision tracking* dalam menghitung posisi kamera dan orientasi yang relatif terhadap *marker*.

e. ArToolkit dapat berjalan dalam segala sistem operasi.

2) Kekurangan ArToolkit

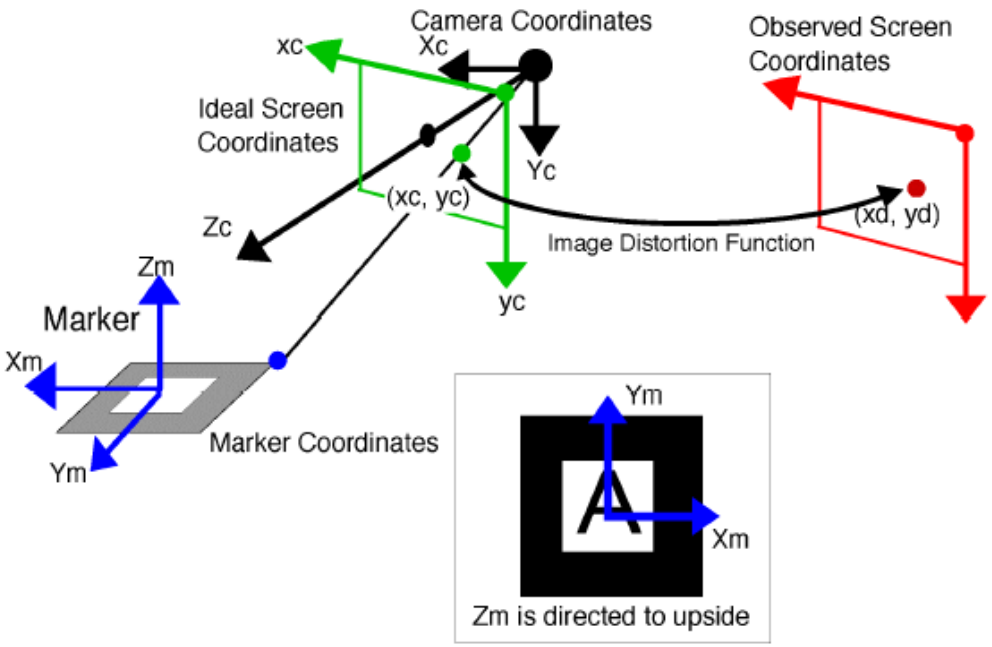
a. Objek virtual akan muncul jika *marker* berada pada posisi yang dapat dilihat oleh kamera.

b. Objek virtual akan hilang apabila terhalang objek lain misalnya tangan.

c. Masalah jangkauan dan masalah cahaya. Semakin kecil atau semakin jauh marker terhadap kamera, maka semakin kecil kemungkinan *marker* dapat dideteksi oleh kamera.

d. Agar objek *virtual* dapat di deteksi dengan baik, *marker* yang dibuat sebaiknya terbuat dari bahan yang tidak memantulkan cahaya.

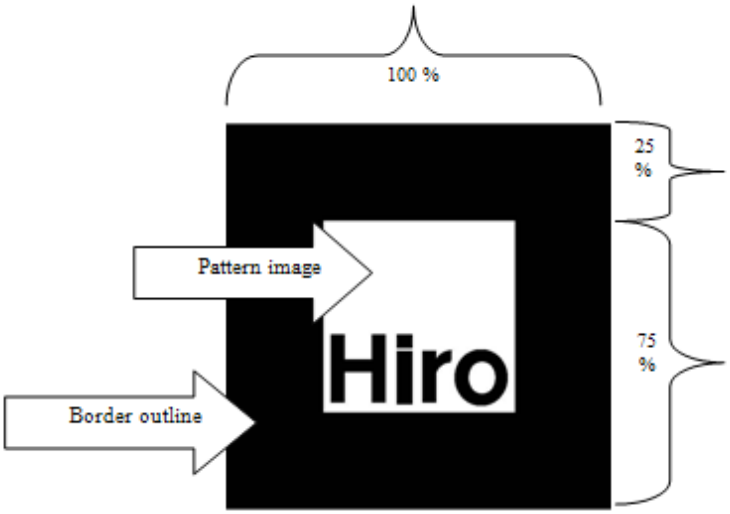
Ilustrasi sistem koordinat dalam ArToolkit ditunjukan pada gambar 2.5



Gambar 2.5 Koordinat ArToolkit

2.5 *Marker*

Marker adalah pola yang dibuat dalam bentuk gambar yang telah dicetak dengan *printer* yang akan dikenali oleh kamera. *Marker* pada ArToolkit merupakan gambar yang terdiri atas *border outline* dan *pattern image* seperti terlihat pada gambar dibawah ini 2.5.

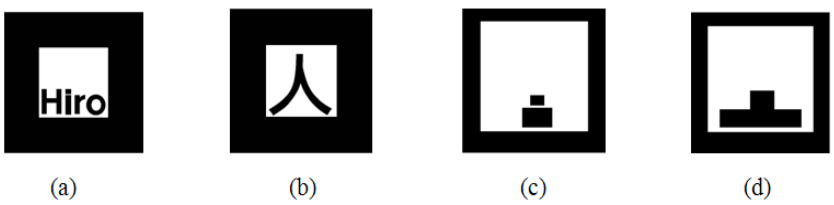


Gambar 2.6 *Marker*

Marker biasanya dengan warna hitam dan putih. Cara pembuatannya pun sederhana tetapi harus diperhatikan ketebalan *Marker* yang akan dibuat, ketebalan *Marker* jangan kurang dari 25 % dari panjang garis tepi agar pada saat proses deteksi *Marker* dapat lebih akurat. Nama Hiro yang ada pada gambar 2.5 merupakan sebuah pembeda saja. Sedangkan objek warna putih sebagai *background*, yang nantinya akan digunakan sebagai tempat objek yang akan direnderer.

Ciri-ciri yang umum digunakan untuk mengenali sata atau beberapa obyek di dalam citra adalah ukuran, posisi atau lokasi, dan orientasi atau sudut

kemiringan obyek terhadap garis acuan yang digunakan. *Marker* terdapat dua intensitas warna yaitu warna hitam dan putih atau sering disebut sebagai citra biner. Citra biner memisahkan daerah (*region*) dan latar belakang dengan tegas, walau pun potensi munculnya kekeliruan selalu ada. Kekeliruan di sini adalah kesalahan mengelompokan piksel ke dalam golongannya, apakah piksel milik suatu daerah dikelompokan sebagai latarbelakang atau sebaliknya. Kesalahan seperti ini sering disebut dengan *noise*. Warna putih pada *Marker* menunjukan warna sebuah objek, sedangkan warna hitam menunjukan latar belakang. Intensitas warna pada suatu objek memiliki warna yang lebih rendah (gelap), sedangkan latar belakang mempunyai intensitas yang lebih tinggi (terang). Namun pada kenyataannya dapat saja berlaku kebalikannya, yaitu objek mempunyai intensitas tinggi dan latar belakang mempunyai intensitas rendah. Kombinasi ini biasanya tergantung pada sifat latar belakang pada saat citra tidak tampil terang sekali (putih) atau gelap sekali (hitam), melainkan di antaranya dengan demikian suatu objek yang sama dapat tampil lebih terang atau lebih gelap daripada latar belakangnya dalam citra, tergantung pada gelap atau terangnya warna yang melatarbelakanginya. (Ahmad, 2005).



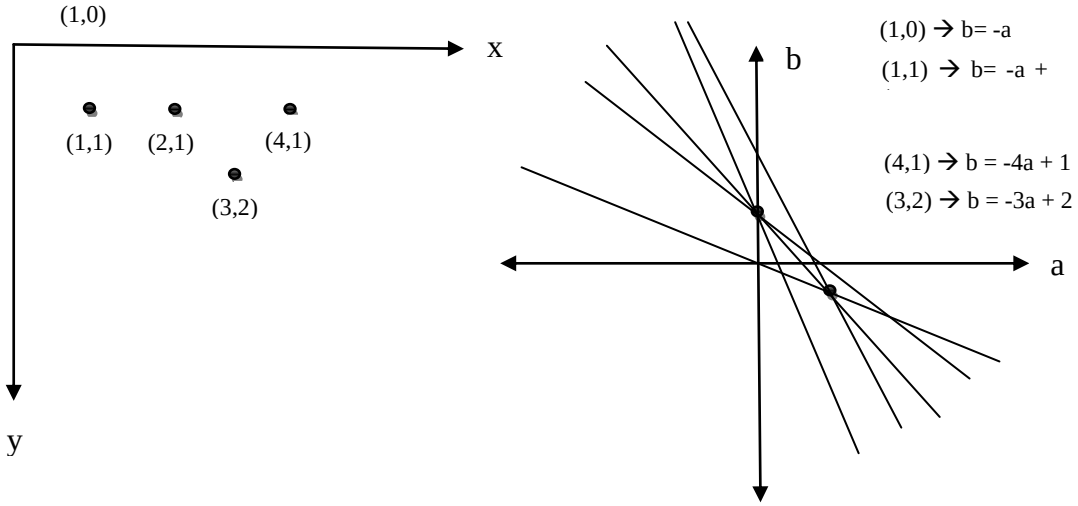
Gambar 2.7 Contoh *Marker*

Ukuran *Marker* yang digunakan dapat mempengaruhi penangkapan pola *Marker* oleh kamera. Semakin besar ukuran *Marker* semakin jauh jarak yang bisa ditangkap oleh kamera dalam mendeteksi *Marker*. Namun disinilah masalahnya, ketika *Marker* bergerak menjauhi kamera, jumlah pixel pada layar kamera menjadi lebih sedikit dan ini bias mengakibatkan pendeteksian tidak akurat.

2.6 Hough Transform

Hough Transform (HT) merupakan suatu teknik untuk menentukan lokasi suatu bentuk dalam citra. HT dicetuskan pertama kali oleh P.V.C. Hough (1962), dilihat dari potensinya sebagai salah satu algoritma dalam pemrosesan citra oleh Rosenfeld (1969), kemudian Duda (1972) mengimplementasikan deteksi garis. Sejak saat itulah HT mengalami perkembangan yang sangat pesat karena banyak keunggulan dan besarnya potensi untuk pembangunan lebih lanjut yang ditawarkan oleh algoritma ini.

HT dapat melakukan pemetaan terhadap titik-titik pada citra ke dalam parameter *space*. Berdasarkan suatu fungsi yang mendefinisikan bentuk yang ingin dideteksi. HT umumnya digunakan untuk melakukan ekstraski garis, lingkaran atau elips pada citra, namun dalam perkembanganya, HT juga telah dapat digunakan untuk melakukan ekstraksi bentuk-bentuk yang lebih kompleks.



Gambar 2.8 Hubungan gambar dan garis dalam transformasi

HT ditransformasikan dari sebuah titik (xi,yi) menjadi garis pada ruang koordinat –mc (Yeni, 2004).

Tabel 2.2 Transformasi persamaan garis

x	Y	y=mx+c	C= (-x)m + y
1	3	3=m.1+c	C= -1m+3
2	2	2=m.2+c	C= -2m+2
3	3	3=m.3+c	C= -3m+3
4	0	0=m.4+c	C= -4m+0

2.6.1 Kuantisasi Parameter

Kuantisasi parameter merupakan salah satu faktor yang penting dalam penerapan HT. Dalam proses kuantisasi, range parameter yang mungkin dipecah menjadi *cell-cell*, kemudian HT melakukan pemetaan dari piksel-piksel pada citra ke dalam *cell-cell* parameter tersebut. Besarnya resolusi untuk proses pemecahan

(misalkan untuk parameter-parameter p,q dan r yang merepsentasikan suatu objek pada *parameter space)* didefinisikan sebagai:

$$\Delta p = p_{i+1} - p_i \qquad (2.1)$$

$$\Delta q = q_{j+1} - q_j \qquad (2.2)$$

$$\Delta r = r_{k+1} - r_k \qquad (2.3)$$

Dimana p_i, q_j dan r_k menyatakan parameter-parameter untuk cell ke (i, j, k), serta Δp, Δq dan Δr menyatakan resolusi dari *cell-cell* untuk parameter p, q dan r. Untuk memperoleh tingkat akurasi hasil deteksi yang paling tinggi, biasanya resolusi yang digunakan adalah:

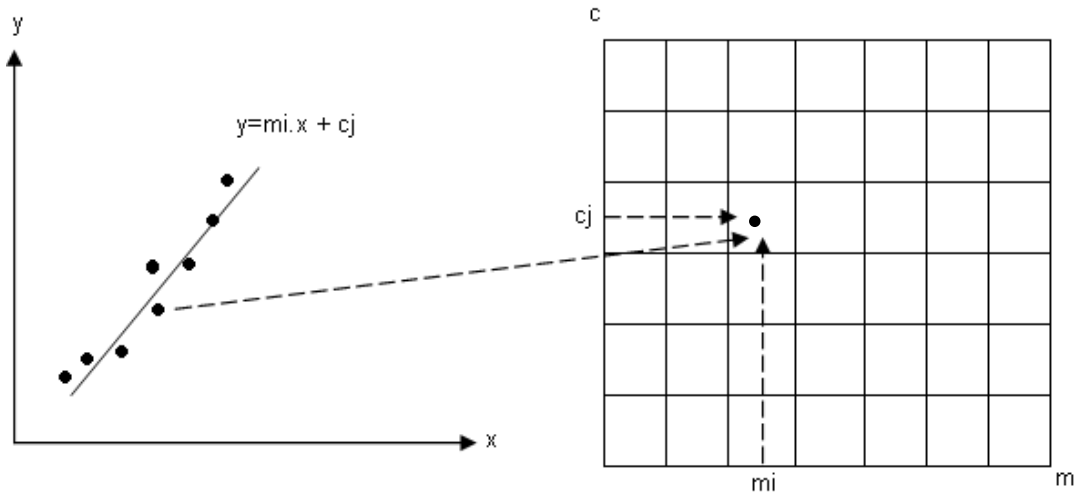
$$\Delta p = \Delta q = \Delta r = 1 \qquad (2.4)$$

Tetapi dengan mempertimbangkan ketersediaan memori dan biaya komputasi, dapat juga digunakan nilai resolusi yang lebih besar. Kuantisasi tidak hanya didapat diaplikasikan pada bidang parameter, tetapi juga pada bidang gambar. Dengan konsep yang sama, bidang gambar dapat dikuantisasi (*down sampling*) dengan:

$$\Delta x = x_{i+k} - x_i \qquad (2.5)$$

$$\Delta y = y_{j+k} - y_j \qquad (2.6)$$

Dimana x_i dan y_j menyatakan koordinat piksel pada gambar, dan $k \geq 1$ menyatakan besar resolusi yang diinginkan. Dengan cara ini, jumlah piksel (atau ukuran input untuk sistem) dapat dikurangi sampai dengan $1/k^2$. Selain itu, *down sampling* juga dapat meningkatkan toleransi terhadap *noise* yang terjadi karena adanya deviasi piksel pada citra *input*.



Gambar 2.9 Kuantisasi Paremeter

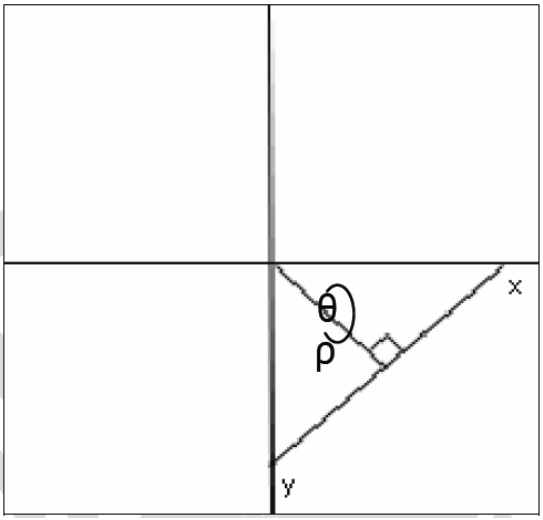
Dapat disimpulkan bahwa dalam proses kuantisasi, baik kuantisasi terhadap bidang paremeter maupun kuantisasi *down sampling* terhadap bidang gambar, perlu mempertimbangakan beberapa faktor, yaitu:

1) Bahwa parameter suatu objek dapat diestimasi secara lebih akurat.
2) Bahwa kuantisasi yang lebih besar/kasar terhadap bidang gambar dapat meningkatkan efisiensi waktu dan *space*.
3) Bahwa kuantisasi yang lebih besar/kasar dapat meningkatkan toleransi terhadap *noise*.

2.6.2 Pemakaian HT untuk Deteksi Garis

Pemakaian HT pertama kali diperkenalkan untuk melakukan deteksi garis. Suatu garis dapat dinyatakan dengan representasii polar sebagai:

$$X \cos(\Theta) + y \sin(\Theta) = p \qquad (2.7)$$



Gambar 2.10 Representasi polar suatu garis dalam bidang gambar

Dari persamaan 2.7, dengan representasi polar gari dapat didefinisikan dalam 2 parameter, yaitu Θ yang menyatakan sudut antar garis dengan sumbu x positif dan p yang menyatakan jarak normal antara garis dengan titik pusat koordinat. Sehingga diperoleh algoritma HT untuk deteksi garis sebagai berikut:

1. Kuantisasi parameter space

   $P[p_{min,\ldots,}p_{max}][\ \Theta_{min,\ldots,}\ \Theta_{max}]$

2. Set semua elemen pada P [p][Θ] menjadi 0
3. Untuk tiap titik (x,y) pada citra, lakukan langkah 4
4. Langkah 4 untuk ($\Theta = \Theta_{min}$; $\Theta <= \Theta_{max}$; Θ++), hitung nilai p:

   $P = x \cos(\Theta) + y \sin(\Theta)$

5. Jika $p >= p_{min}$ && $p <= p_{max}$

   P[p][Θ] = P[p][Θ] + 1 //voting

6. Cari local maxima dari P [p][Θ]

Dari persamaan 2.7 tersebut, jika Θ dikuantisasi pada interval [0,π), maka akan diperoleh parameter yang unik untuk tiap garis. Kemudian dapat dilakukan kuantisasi terhadap nilai maksimum dan minimum untuk masing-masing parameter sebagai berikut:

$0 \leq \Theta < \pi$

$-0.5d \leq p \leq 0.5d$ dengan d=diagonal bidang gambar.

HT melakukan pemetaan terhadap titik pada bidang gambar ke bidang parameter. Sehingga dapat dilihat beberapa sifat yang menjelaskan hubungan diantara kedua bidang tersebut yaitu (Richard, 1971) :

1) Sebuah titik pada bidang gambar berkorespondensi dengan suatu kurva sinusoidal pada parameter space.
2) Sebuah titik pada *parameter space* berkorespondensi dengan suatu garis lurus pada bidang gambar.
3) Titik-titik yang berada pada garis yang sama pada baidang gambar, berkorespondensi dengan kurva-kurva yang melalui titik yang sama pada parameter space. Titik tersebut merupakan parameter dari garis yang bersangkutan pada bidang gambar (bidang x- y)
4) Titik-titik yang berada pada kurva yang sama pada paremeter space berkorespondensi dengan garis-garis yang melalui titik yang sama pada bidang gambar.

Contoh hasil penggunaan HT untuk deteksi garis dapat dilihat pada gambar 2.9.



(a) Citra asli (b) Citra hasil *edge detection*

Gambar 2.11 *Marker* hasil penggunaan Hough Transform

Pada gambar 2.8 dapat terlihat adanya *variable vote threshold. Vote threshold* sendiri merupakan suatu nilai yang dipilih unuk memberikan batasan seberapa besar *vote* yang harus dikumpulkan oleh sepasang parameter dalam *array accumulator* agar parameter tersebut dapat dianggap signifikan sebagai representasi suatu garis pada bidang gambar. Bisa dilihat perbedaannya, citra asli yang berwarna hitam berubah menjadi warna putih dan citra warna putih berubah menjadi hitam.

2.6.3 Kelebihan dan Kekurangan HT

Dari uraian diatas, maka dapat diperoleh keunggulan dan kekurangan algoritma HT, diantaranya:

Kelebihan:

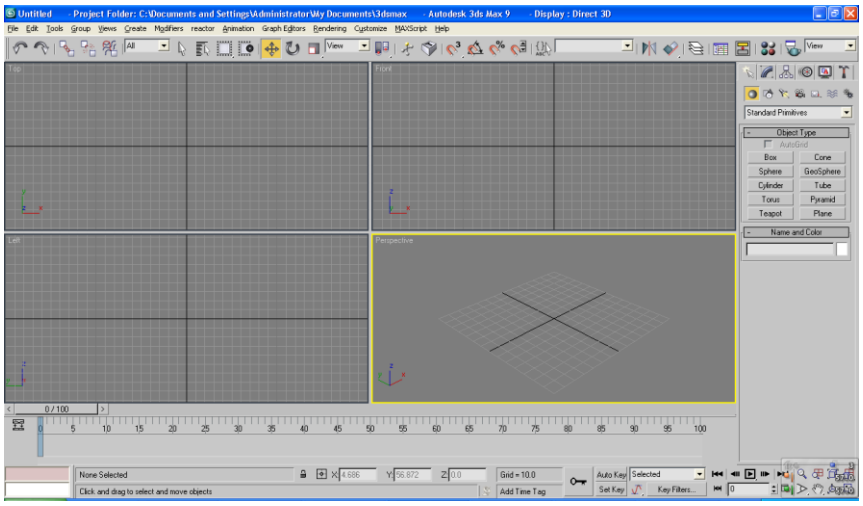
1) HT dapat mendeteksi keberadaan objek yang memiliki pola tertentu misal,objek yang dapat direpresentasikan secara matematis dengan parametertertentu), walaupun tidak diketahui ukuran maupun posisinya.
2) HT relatif tidak terpengaruh oleh *noise* atau data hilang pada citra.

Kekurangan :

1) Kompleksitas dan ukuran *accumulator array* meningkat secara *polynomial* sebanding dengan bertambahnya jumlah parameter yang merepresentasikanobjek yang ingin di deteksi.
2) Hasil deteksi sensitif terhadap kuantisasi bidang parameter dan bidang gambar.
3) Hasil deteksi sensitif terhadap pemilihan nilai *vote threshold.*

2.7 3DS Max

Dalam pembuatan AR ini kita menggunakan *software* Autodesk 3ds Max 9 32-bit. 3D Studio Max adalah *software* visualisasi (modeling dan animasi) tiga dimensi yang popular dan serbaguna. Hasil yang dibuat di 3D Studio Max sering digunakan di pertelevisian, media cetak, games, *web* dan lain-lain.



Gambar 2.12 Tampilan awal 3DS MAX

3DS MAX memberikan tiga kemungkinan untuk menetukan sistem koordinat sebuah titik dalam ruang, yaitu dengan memperhatikan terhadap sumbu-sumbu x, y, z dan sudut yang terjadi. Ketiga kemungkinan sistem koordinat itu ialah:

a) Koordinat *Cartesian* (*rectangular coordinat*).

   Menentukan koordinat dengan menggunakan sumbu-sumbu x, y, z. yaitu (x), (y), (z). Penulisannya (0.5,0.9,0.0); (0.42,0.39,0.82)

b) Koordinat *cylindrical*

   Cara ini mengabungkan antara jarak, sudut dan koordinat sumbu z yaitu: (jarak)< (sudut),(z) Penulisannya: (.03<60.95,0.0);(0.57<43,0.82)

c) Koordinat *spherical*

   Cara ini menggabungkan antara jarak dan dua sudut, dan masing-masing besaran dipisahkan dengan tanda<, yaitu: (jarak)<(sudut)<(sudut), penulisannya: (1.03<60.95<0); (1<43<55).

2.8 *Webcam*

Webcam (singkatan dari *web camera*) adalah sebutan bagi kamera *real-time* (bermakna keadaan pada saat ini juga) yang gambarnya bisa diakses atau dilihat melalui *World Wide Web*, program *instan messaging*, atau aplikasi *video call*. Istilah *Webcam* merujuk pada tekologi secara umumnya, sehingga kata *web* kadang-kadang diganti dengan kata lain yang mendeskripsikan pemandangan yang ditampilkan di kamera, misalnya *StreetCam* yang memperlihatkan pemandangan jalan. Ada juga *Metrocam* yang memperlihatkan pemandangan

panorama kota dan pedesaan, *TraffiCam* yang digunakan untuk memonitor keadaan jalan raya, cuaca dengan *Weathercam*, bahkan keadaan gunung berapi dengan *VolcanoCam*. *Webcam* atau *web camera* adalah sebuah kamera video digital kecil yang dihubungkan ke komputer melalui *port USB* ataupun *port* COM (Wikipedia, WEB).



Gambar 2.13 *Webcam*

Pada awalnya, bentuk *web camera* terbatas pada bentuk-bentuk standar yang hanya terdiri dari lensa dan papan sirkuit serta *casing* yang bisa. Seiring dengan perkembangan teknologi, bentuk *web camera* pun sudah makin bervariasi dengan fitur-fitur yang makin canggih.

Salah satu bentuk *web camera* yang unik adalah bentuk boneka yang lucu, *web camera* ini dapat disalah artikan hanya sebagai boneka dan bukan *Webcam*. Sebuah penemuan oleh *Microsoft* pada tahun 2004 menggambarkan kemajuan perkembangan teknologi *web camera*. i2i adalah sebuah sistem dua-kamera yang dengan sangat hati-hati mengikuti pergerakan individu. Kamera ini menggunakan perhitungan algoritma yang secara spesial dikembangkan untuk memfusikan apa yang setiap kamera lihat untuk membuat gambar '*cyclopean' stereo* yang akurat. Kamera ini juga dapat menampilkan *emoticon* 3D yang melayang. Sistem i2i

dapat juga menghasilkan gambar *background* yang realistis sehingga pengguna dapat berpura-pura berada di tempat lain. Kemampuan sistem i2i ini, diantaranya yaitu kemampuan *tracking* (disebut *smart framing*) dan juga kemampuan *smart focusing*, dapat menambah pengalaman berkonferensi bagi pengguna.

Teknologi *Webcam* pada awalnya mendapat dukungan komersial dari industri pornografi. Industri ini membutuhkan gambar-gambar '*live*' dan meminta pembuatan *software* yang mampu melakukannya tanpa *web browser plugins*. Hal ini melahirkan teknologi *live streaming Webcam* yang masih tetap ada hingga sekarang. Sekarang ini *web camera* yang ada di pasaran pada umumnya terbagi ke dalam dua tipe: *web camera* permanen (*fixed*) dan *revolving web camera*. Pada *web camera* permanen terdapat pengapit untuk mengapit lensa standar di posisi yang diinginkan untuk menangkap gambar pengguna. Sedangkan pada *revolving web camera* terdapat landasan dan lensa standar dipasang di landasan tersebut sehingga dapat disesuaikan ke sudut pandang yang terbaik untuk menangkap gambar pengguna.

Penggunaan *web camera* mencakup *video conferencing*, *internet dating*, video *messaging*, *home monitoring*, *images sharing*, *video interview*, *video phone-call*, dan banyak hal lain. Kamera untuk *video conference* bisaanya berbentuk kamera kecil yang terhubung langsung dengan komputer. Kamera analog juga kadang-kadang digunakan, kamera ini terhubung dengan video *capture card* dan tersambung dengan internet (baik langsung maupun tidak langsung). Saat ini kamera untuk *video conference* sudah makin maju, sudah ada *web camera* yang di dalamnya terdapat *microphone* maupun *noise cancellation* untuk memfokuskan

audio ke *speaker* yang terletak di depan kamera sehingga *noise* yang ada tidak mengganggu jalannya konferensi.

2.8.1 Tipe-Tipe *Webcam*

Slim1320 (*True* 1.3 *Mega Pixels High Performance WebCam*), Slim 2020AF (*Mega Pixel Web Camera Auto Focus*), Eye 312 (*Simplify Instant Video and Chat*), *Eye* 110 (*Instant Video Messenger Webcam*) serta i-Look 1321 (*Advance* 1.3 *Mega Pixel Camera*), dan lain-lain. Sekarang hampir semua kamera digital dan HP bisa dijadikan sebagai kamera web (*Webcam*). Dalam pembuatan ARtatasurya ini digunakan *Webcam* MicroPack 5 MP. Karena teknologi AR membutuhkan kualitas video yang bagus dalam mendeteksi *Marker* . (Wikipedia, WEB).

2.8.2 Cara Kerja *Webcam*

Sebuah *web camera* yang sederhana terdiri dari sebuah lensa standar, dipasang di sebuah papan sirkuit untuk menangkap sinyal gambar, *casing* (cover), termasuk *casing* depan dan *casing* samping untuk menutupi lensa standar dan memiliki sebuah lubang lensa di *casing* depan yang berguna untuk memasukkan gambar, kabel *support*, yang dibuat dari bahan yang *fleksibel*, salah satu ujungnya dihubungkan dengan papan sirkuit dan ujung satu lagi memiliki *connector*, kabel ini dikontrol untuk menyesuaikan ketinggian, arah dan sudut pandang *web camera*. Sebuah *web camera* bisaanya dilengkapi dengan *software*, *software* ini mengambil gambar-gambar dari kamera digital secara terus menerus ataupun

dalam interval waktu tertentu dan menyiarkannya melalui koneksi internet. Ada beberapa metode penyiaran, metode yang paling umum adalah *software* merubah gambar ke dalam bentuk file JPEG dan menguploadnya ke *web server* menggunakan *File Transfer Protocol* (FTP) (Wikipedia, WEB).

Frame rate mengindikasikan jumlah gambar sebuah *software* dapat ambil dan transfer dalam satu detik. Untuk *streaming* video, dibutuhkan minimal 15 *frame per second* (fps) atau idealnya 30 fps. Untuk mendapatkan *frame rate* yang tinggi, dibutuhkan koneksi internet yang tinggi kecepatannya. Sebuah web *camera* tidak harus selalu terhubung dengan komputer, ada *web camera* yang memiliki *software Webcam* dan *web server bulit-in*, sehingga yang diperlukan hanyalah koneksi internet. *Web camera* seperti ini dinamakan "*network camera*". Kita juga bisa menghindari penggunaan kabel dengan menggunakan hubungan radio, koneksi *Ethernet* ataupun *Wifi* (Wikipedia, WEB).

2.9. Definisi UML (*Unified Modeling Language*)

Dalam penggambaran informasi-informasi secara grafis penulis menggunakan notasi UML (*Unified Modeling Language*) yang akan digambarkan dengan bantuan komputer melalui perangkat lunak *Rational Rose* 2000. Dengan pemodelan menggunakan UML ini diharapkan dapat melakukan:

1) Tinjauan umum dapat menggambarkan arsitektur sistem secara keseluruhan.
2) Penelaahan bagaimana objek-objek dalam sistem saling mengirimkan pesan dan saling bekerjasama satau sama lain.

3) Menguji apakah sistem atau perangkat lunak sudah berfungsi seperti yang seharusnya.
4) Dokumentasi sistem atau perangkat lunak untuk keperluan-keperluan tertentu dimasa yang akan datang.

Setiap sistem yang kompleks seharusnya bisa dipandang dari sudut yang berbeda-beda sehingga kita bisa mendapatkan pemahaman secara menyeluruh. Untuk upaya tersebut UML menyediakan sembilan jenis diagram yang dapat dikelompokan berdasarkan sifatnya statis atau dinamis (Nugroho, 2005).

1. Diagram kelas. Bersifat statis, diagram ini memperlihatkan himpunan kelas, antarmuka, kolaborasi, serta aksi relasi. Diagram ini umumnya dijumpai pada pemodelan sistem berorientasi objek. Meskipun bersifat statis, namun sering pula diagram kelas memuat kelas-kelas aktif.
2. Diagram objek. Bersifat statis. Diagram ini memperlihatkan objek-objek serta relasi-relasi antarobjek. Diagram objek memperlihatkan instansiasi statis darisegala sesuatu yang dijumpai pada diagram kelas.
3. *Use-case Diagram*. Bersifat statis. Diagram ini memperlihatkan himpunan usecase dan actor-aktor (suatu jenis khusus darikelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku dari suatusistem yang dibutuhkan serta diharapkan pengguna.
4. *Sequence diagr*am. Bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan (*message*) dalam suatu waktu tertentu.

5. *Collaboration Diagram*. Bersifat dinamis. Diagram kolaborasi adalah diagram interaksi yang menekankan organisasi strutkural dari objek-objek yangmenerima serta mengirim pesan (*message*).
6. *Statechart Diagram.* Bersifat dinamis. diagram *state* ini memperlihatkan statestate pada sistem, memuat *state*, transisi, *event*, serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka (*interface*), kelas, kolaborasi dan terutama penting pada pemodelan sistem sistem yang reaktif.
7. *Activity diagram.* Bersifat dinamis. Diagram aktivitas ini adalah tipe khusus dari diagram *state* yang memperlihatkan aliran dari suatu aktifitas ke aktifitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi dalam suatu sistem dan memberikan. Diagram ini terutama penting dalam pemodelan fungsi-fungsi dalam suatu sistem dan memberikan tekanan pada aliran kendali antar objek.
8. *Component Diagram.* Bersifat statis. Diagram komponen ini memperlihatkan organisasi serta kebergantungan sistem/perangkt lunak pada komponen komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan ke dalam satu atau lebih kelas-kelas, antarmuka (*interface*), serta kolaborasi-kolaborasi.
9. *Deployment Diagram.* bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (saat *run-time)*. Diagram ini memuat simpul-simpul (*node*) beserta komponen-komponen yang ada di dalamnya. Deployment diagram berhubungan erat dengan diagram komponen dimana *deployment*

diagram memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing*).

2.10 Tata Surya

Tata Surya adalah kumpulan benda langit yang terdiri atas sebuah bintang yang disebut Matahari dan semua objek yang terikat oleh gaya gravitasinya. Objek-objek tersebut termasuk delapan buah planet yang sudah diketahui dengan orbit berbentuk elips, lima planet kerdil/katai, 173 satelit alami yang telah diidentifikasi, dan jutaan benda langit (meteor, asteroid, komet) lainnya. Tata Surya terbagi menjadi Matahari, empat planet bagian dalam, sabuk asteroid, empat planet bagian luar, dan di bagian terluar adalah Sabuk Kuiper dan piringan tersebar. Awan Oort diperkirakan terletak di daerah terjauh yang berjarak sekitar seribu kali di luar bagian yang terluar.

Berdasarkan jaraknya dari matahari, kedelapan planet Tata Surya ialah Merkurius (57,9 juta km), Venus (108 juta km), Bumi (150 juta km), Mars (228 juta km), Yupiter (779 juta km), Saturnus (1.430 juta km), Uranus (2.880 juta km), dan Neptunus (4.500 juta km). Sejak pertengahan 2008, ada lima objek angkasa yang diklasifikasikan sebagai planet kerdil. Orbit planet-planet kerdil, kecuali Ceres, berada lebih jauh dari Neptunus. Kelima planet kerdil tersebut ialah Ceres (415 juta km. di sabuk asteroid; dulunya diklasifikasikan sebagai planet kelima), Pluto (5.906 juta km.; dulunya diklasifikasikan sebagai planet

kesembilan), Haumea (6.450 juta km), Makemake (6.850 juta km), dan Eris (10.100 juta km).