APLIKASI DETEKSI GERAK DALAM AUGMENTED REALITY

Oleh:

Liliana, M.Eng.

Ir.Kartika Gunadi, M.T

Andreas Yohan

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS KRISTEN PETRA

2012

LAPORAN PENELITIAN

126/Pen/Informatika/II/2012

APLIKASI DETEKSI GERAK DALAM AUGMENTED REALITY

Oleh:

Liliana, M.Eng.

Ir.Kartika Gunadi, M.T

Andreas Yohan

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS KRISTEN PETRA

2012

LEMBAR IDENTITAS DAN PENGESAHAN

LAPORAN HASIL PENELITIAN

1.	a. Judul Penelitian	: APLIKASI DETEKSI GERAK DALAM AUGMENTED REALITY
	b. No Penelitian	: 126/Pen/Informatika/I/2012
	c. Jalur Penelitian	: I/~~II~~/~~III~~/~~IV~~
2.	Ketua Peneliti	
	a. Nama Lengkap dan Gelar	: Liliana, M.Eng
	b. Jenis Kelamin	: Perempuan
	c. Pangkat / Golongan / NIP	: Pembina/ Gol. 4A/ 03-024
	d. Bidang Ilmu yang Diteliti	: Teknologi Perangkat Lunak
	e. Jabatan Akademik	: Dosen
	f. Fakultas/Jurusan	: Fakultas Teknologi Industri, Jurusan Teknik Informatika
	g. Universitas	: Universitas Kristen Petra
	Anggota Tim Peneliti	
	a. Nama Lengkap dan Gelar	: Ir.Kartika Gunadi, M.T
	b. Jenis Kelamin	: Laki-laki
	c. Pangkat / Golongan / NIP	: Pembina/ Gol. 4A/ 88004
	d. Bidang Ilmu yang Diteliti	: Teknologi Perangkat Lunak
	e. Jabatan Akademik	: Dosen
	f. Fakultas/Jurusan	: Fakultas Teknologi Industri, Jurusan Teknik Informatika
	g. Universitas	: Universitas Kristen Petra
	Anggota Tim Peneliti	
	a. Nama Lengkap dan Gelar	: Andreas Yohan
	b. Jenis Kelamin	: Laki-laki
	c. Pangkat / Golongan / NIP	: -
	d. Bidang Ilmu yang Diteliti	: Teknologi perangkat lunak
	e. Jabatan Akademik	: -
	f. Fakultas/Jurusan	: Fakultas Teknologi Industri, Jurusan Teknik Informatika
	g. Universitas	: Universitas Kristen Petra
3.	Lokasi Penelitian	: Surabaya
4.	Kerjasama dengan Instansi Lain	:
5.	Tanggal Penelitian	: September 2011 s/d Juli 2012
6.	Biaya	: -

Surabaya, 19 Juli 2012

Mengetahui,
Ketua Jurusan Teknik Informatika

Ketua Tim Peneliti,

Yulia, M.Kom
NIP. 99036

Liliana, M.Eng.
NIP. 03024

Menyetujui,
Dekan Fakultas Teknologi Industri

Djoni Haryadi Setiabudi
NIP.85009

ABSTRAK

Pada saat ini dunia animasi sangat berkembang baik dalam pembuatan game, film, edukasi, maupun periklanan. Pada awal mulanya animasi–animasi yang ada masih berupa gambar dua dimensi, tetapi dengan semakin berkembangnya teknologi, pembuatan animasi sudah beralih ke ranah 3 dimensi untuk *virtual reality*. Tahap berikutnya adalah manusia mengkombinasikan representasi virtual dengan persepsi dari dunia nyata. Sehingga interaksi antara manusia dengan komputer dapat lebih erat yang disebut dengan *Augmented Reality(AR)*.

Dari latar belakang masalah tersebut, sebagai tahap pertama dalam pembuatan *Augmented Reality*, dibuatlah aplikasi untuk mendeteksi papan acuan yang berupa tanda (*marker*) khusus dalam suatu gambar tertentu dengan bantuan *webcam*. Dari hasil pendeteksian terebut, akan dilakukan estimasi rotasi yang terjadi pada *marker*, Kemudian akan ditambahkan (*overlay*) suatu objek 3D ke tepat diatas gambar *marker* tadi.

Berdasarkan hasil pengujian yang dilakukan, pendeteksian *marker* dan estimasi rotasi relative kurang akurat dengan perbedaan presisi +- 10 derajat dengan sudut aslinya.

Kata Kunci :

Augmented Reality, Marker Detection.

ABSTRACT

At this time, the world of animation is very well developed in the manufacture of games, movies, education, and advertising. In the early days of existing animations are still in a two-dimentional image, but as the technology increasing, creation of animation has been transferred to the realm of 3-dimentional virtual reality world. The next stage is people wanted to combine the virtual representation with the real world perception. So that the interaction between humans and computers can more closer. That's called Augmented Reality (AR).

From the problem background, as the first stage to develop Augmented Reality, an application is built to detect the reference board in the form of a sign (marker) in a particular image from webcam. From the detection result, a defined 3D object will be added (overlay) on the top of marker.

Based on the results of tests performed, marker detection and rotation estimation relatively less accurate with a difference of precision +- 10 degrees.

Keywords :

Augmented Reality, marker detection

PRAKATA

Puji syukur ke hadirat Tuhan Yesus Kristus atas segala rahmat dan kasih karunia-Nya, sehingga pada akhirnya penyusun dapat menyelesaikan penelitian ini. Penulis mengucapkan terima kasih kepada semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu terselesaikannya penelitian ini.

Akhir kata, penyusun menyadari sepenuhnya bahwa penelitian ini masih terdapat kekurangan-kekurangan yang memerlukan penyempurnaan lagi sehingga penulis sangat mengharapkan dan dengan senang hati menerima segala kritik dan saran yang diberikan oleh siapapun yang bersifat membangun demi kebaikan dan kesempurnaan penelitian ini.

Surabaya, Juli 2012

Peneliti

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

1. PENDAHULUAN

1.1. Latar Belakang

Dewasa ini dunia animasi semakin berkembang, baik dalam pembuatan game, film, edukasi, maupun periklanan. Pada awal mulanya animasi–animasi yang ada tersebut masih berupa gambar dua dimensi, tetapi dengan semakin berkembangnya teknologi, pembuatan animasi sudah beralih ke ranah 3 dimensi. Untuk memudahkan pembuatan animasi tersebut maka sudah banyak dikembangkan aplikasi – aplikasi dengan menggunakan komputer sebagai media utama.

Untuk memudahkan merepresentasikan suatu objek 3 dimensi yang sudah dibuat ke dalam dunia nyata, dapat dilakukan pelacakan suatu tanda (*marker*) khusus. Selain itu, kemampuan untuk melacak suatu tanda tertentu dalam suatu kumpulan objek memberikan keuntungan yang cukup besar terhadap interaksi manusia dan komputer. Untuk pelacakan tersebut dapat digunakan media kamera sebagai input.

Beberapa aplikasi *virtual reality* didesain untuk mengkombinasikan representasi virtual dengan persepsi dari dunia nyata. Representasi virtual memberikan informasi tambahan yang mana tidak didapatkan oleh user yang tidak memakai alat bantu.

Augmented reality dapat digunakan untuk membantu memberikan gambaran atau simulasi kepada *user*. Misalnya untuk visualisasi pengenalan *concept car* atau visualisasi buku cerita 3 dimensi.

Oleh karena itu sebagai tahap pertama dalam pembuatan *Augmented Reality*, dibuatlah aplikasi untuk mendeteksi papan acuan yang berupa tanda (*marker*) khusus dalam suatu gambar tertentu.

1.2. Perumusan Masalah

- Bagaimana mendeteksi apakah suatu bagian *image* merupakan tanda

- Bagaimana menentukan posisi papan acuan di dunia nyata pada dunia maya
- Bagaimana cara mengenali apakah terjadi perubahan posisi (translasi atau rotasi) papan acuan
- Bagaimana merepresentasikan perubahan posisi tersebut terhadap objek yang akan diaplikasikan kedalam lingkungan virtual (maya)

1.3. Tujuan

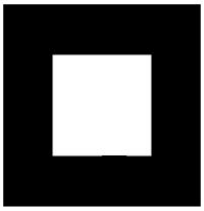
Tujuan dari skripsi ini adalah merancang perangkat lunak (*software*) untuk mendeteksi gerak yang dilakukan terhadap suatu simbol tertentu.

1.4. Ruang Lingkup

Pengambilan gambar bergerak akan dilakukan melalui webcam dengan pencahayaan yang cukup

Penggunaan papan acuan berupa kertas yang datar. Pada papan acuan terdapat tanda (*marker*) berupa kotak yang diberikan kode atau identitas khusus.

Bentuk *marker* minimal sebagai berikut :



Langkah yang digunakan dalam proses image preprocessing adalah camera calibration untuk memperbaiki citra kamera dari distorsi lena kamera. Kemudian dilakukan proses pencarian koordinat dari papan acuan.

Pendeteksian hanya dibatasi rotasi dan translasi yang dilakukan pada meja datar. Pendeteksian dilakukan hanya 2 dimensi yaitu sumbu-x (lebar) dan sumbu-z (kedalaman) tanpa mendeteksi perpindahan sumbu-y (tinggi).

Output yang ditampilkan *non real-time,* tidak ditampilkan secara langsung. Antara input dan output dapat delay.

1. Objek ditangkap kamera (webcam) dan dirubah ke bentuk simulasi 3D di layar monitor dengan background, lalu bergerak sesuai pengenalan objek.

2. Pembuatan aplikasi menggunakan *Micorosoft Visual Studio 2008.*

1.5. Sistematika Penulisan

Garis besar penulisan Penelitian ini adalah sebagai berikut:

BAB 1 : Pendahuluan

Berisi latar belakang masalah, perumusan masalah dan ruang lingkup, tujuan Tugas Akhir, dan metodologi penelitian yang dipakai.

BAB 2 : Teori Penunjang

Berisi berbagai teori yang berhubungan dengan definisi augmented reality dan algoritma yang digunakan untuk proses pendeteksian tanda (*marker*)

BAB 3 : Desain Sistem

Berisi perencanaan pembuatan keseluruhan sistem dalam aplikasi yang akan dibuat.

BAB 4 : Implementasi Sistem

Berisi pembuatan aplikasi dan segmen-segmen program yang ada dalam aplikasi yang dibuat.

BAB 5 : Pengujian Sistem

Berisi penjelasan dan hasil pengujian yang dilakukan terhadap aplikasi yang dibuat.

BAB 6 : Kesimpulan dan Saran

Berisi kesimpulan dari hasil pengujian perangkat lunak dan beberapa saran untuk pengembangan lebih lanjut

2. DASAR TEORI

2.1. Definisi Augmented Reality

Menurut William R & Allan B, 2003; Bowo Dwi, 2012; Stephen Cawood & Mark Fiala, 2008, *Augmented reality* memiliki pengertian sebagai sebuah istilah untuk lingkungan buatan yang menggabungkan dunia *virtual* yang dibuat oleh komputer dengan dunia nyata pada media tertentu seperti monitor / layar LCD / *Head Mounted Display*. Sehingga terjadi interaksi antara objek nyata tersebut dengan objek virtual.

Augmented reality menampilkan objek nyata dalam dunia virtual secara *overlay* (menumpuk) dengan batasan suatu pola yang ditentukan. Sehingga secara otomatis tampilan virtual tersebut menjadi lebih dekat dengan lingkungan nyata yang sebenarnya.

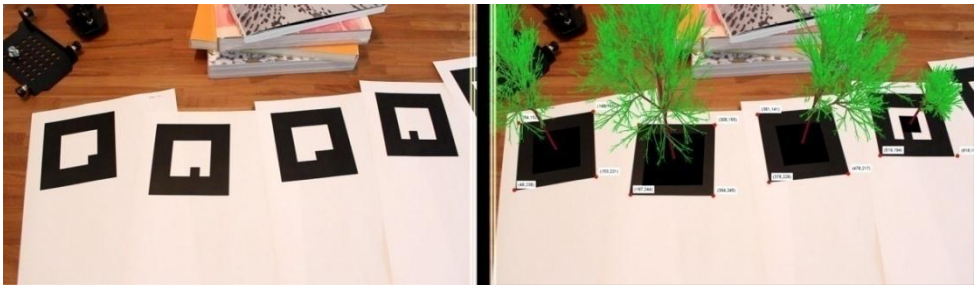
Augmented reality berbeda dengan *virtual reality*. Secara singkat, *Virtual reality* dapat lebih didefinisikan sebagai lingkungan virtual yang dihasilkan oleh komputer. Semua object yang dihasilkan adalah hasil *generate* dari komputer. Sedangkan *Augmented reality* sedikit berbeda dengan *virtual reality*. Dimana pada *Augmented reality* terdapat penggabungan antara dunia virtual dengan dunia nyata. Dengan bantuan *Augmented reality* seperti *computer vision* dan pengenalan objek, maka lingkungan nyata disekitar akan dapat berinteraksi dalam bentuk digital (virtual). Informasi–informasi yang ada di lingkungan sekitar dapat ditangkap dan diolah menjadi inputan bagi dunia virtual. Dari inputan tersebut, dapat diolah hingga menjadi informasi–informasi baru yang lain, yang kemudian ditambahkan ke dalam dunia nyata secara *real time*, yang nantinya seolah olah informasi tersebut adalah nyata seperti terlihat pada Gambar 2.1.

Beberapa aplikasi *virtual reality* didesain untuk mengkombinasikan representasi virtual dengan persepsi dari dunia nyata. Representasi virtual tersebut memberikan informasi tambahan yang mana tidak didapatkan oleh *user* yang tidak memakai alat bantu seperti terlihat pada Gambar 2.2. Aplikasi semacam inilah yang disebut dengan *Augmented reality (AR)*

Gambar 2.1. *Augmented reality* untuk simulasi perencanaan pembangunan gedung baru

Sumber : http://www.augmentedplanet.com/wp-content/uploads/2010/08/worldslargest.jpg

Pada Gambar 2.1, Suatu *marker* yang besar diletakkan pada daerah pembangunan, kemudian dilakukan pengambilan gambar dengan helikopter. Dari imput yang didapat kemudian dilakukan suatu simulasi perencanaan pembangunan gedung yang baru



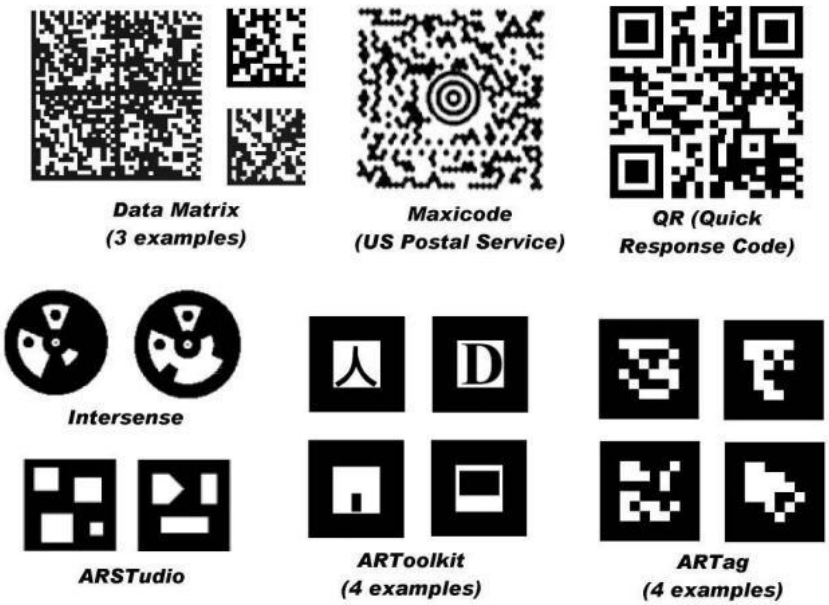
Gambar 2.2. perbandingan apa yang dilihat (kiri) dengan output *Augmented reality* (kanan)

Sumber: http://www.creativeapplications.net/wp-content/uploads/2011/12/Amnon-Owed-ARtut-01.jpg

Pada Gambar 2.2 kiri merupakan gambar apa yang dilihat oleh user tanpa memakai bantuan apapun. Sedangkan gambar kana merupakan visualisasi apa yang dilihat dengan menggunakan *augmented reality system*

2.2. *Planar Marker System*

Menurut Martin Hirzer (2008), Ada banyak praktisi *computer vision* menggunakan pola 2 dimensi untuk menyimpan informasi seperti terlihat pada Gambar 2.3. Area yang menggunakan pola tersebut beraneka ragam, dari sistem industri (dimana tanda didesain untuk digunakan dalam pencarian lokasi. Contohnya *Maxicode* yang digunakan oleh kantor Pos di Amerika) maupun *DataMatrix* dan *QR code* yang digunakan untuk memberi label suatu barang ataupun yang sudah umum di kalangan masyarakat.



Gambar 2.3. Macam – macam planar marker system

Untuk mengurangi sensitivitas kondisi pencahayaan, *planar marker system* biasanya menggunakan warna bitonal (2 warna) yaitu kombinasi hitam dan putih. sehingga tidak diperlukan identifikasi warna yang rumit. Selain itu beberapa *marker system* menggunakan sistem data bit untuk menyimpan data. Yang kemudian dikembangkan lagi dengan pengaplikasian *error detection* dan *error correction*.

Desain suatu *marker* biasa nya ditentukan dari kebutuhan aplikasi. Sebagai contoh, pada ban berjalan (*conveyor belt*) suatu pabrik, jenis *marker* yang digunakan adalah *Maxicode* ataupun *QR code* yang mempunyai kemampuan untuk meng-*encoding* informasi. Tetapi jenis *marker* seperti ini memiliki

keterbatasan sudut pandang. *Marker* jenis ini tidak didesain untuk sudut pandang yang terdistorsi. Selain itu, tidak terdapat point yang cukup pada image untuk digunakan sebagai titik acuan mengkalkulasi posisi.

Sedangkan pada aplikasi *Augmented reality*, sudut pandang yang luas merupakan hal yang sangat penting. Hal itu berarti aplikasi harus tetap dapat mendeteksi meskipun *image* di yang ditangkap kamera terdistorsi.

Informasi yang disimpan pada *marker* sebaiknya tidak terlalu padat untuk meningkatkan jarak pendeteksian. Tetapi pada umumnya *marker Augmented reality* hanya menyimpan informasi yang sedikit, yaitu hanya id yang digunakan untuk membedakan *marker* yang satu dengan yang lain.

Augmented reality System memerlukan setidaknya minimal 4 *unique point* yang dapat digunakan untuk memprediksi jarak. Pada umumnya *marker* yang digunakan terdapat garis tepi persegi (*quadrilateral outline*). Sehingga 4 garis tepi tersebut dapat digunakan sebagai titik acuan dalam perhitungan perkiraan jarak dalam 3 dimensi.

2.3. Contoh Aplikasi komersial yang menggunakan *Augmented Reality*

Gambar 2.4. Iklan Nissan Juke
Sumber : http://www.autoguide.com/gallery/gallery.php/d/246831-4/nissan_juke.jpg

Gambar 2.5. Iklan Nissan Juke
Sumber :
http://media.photobucket.com/image/juke%20augmented%20reality/wizardtoys/NewPicture3.jpg

Gambar 2.4 dan Gambar 2.5 merupakan contoh aplikasi augmented reality yang digunakan untuk media promosi mobil Nissan Juke. Pada aplikasi tersebut user dapat berinteraksi secara langsung dengan marker. Tujuan dari media promosi tersebut adalah mendekatkan produk yang dijual dengan pelanggan. Sehingga pelanggan tidak perlu harus datang ke dealer untuk melihat gambaran mobil yang dijual.

A child read a 3D animation story book in Helsinki, capital of Finland on Oct. 5, 2010. (Xinhua/Zhao Changchun)

Gambar 2.6. Buku Cerita menggunakan AR
Sumber : http://www.dibidogs.com/

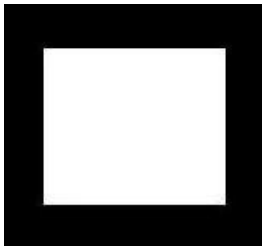
Pada Gambar 2.6 dapat dilihat merupakan salah satu implementasi *augmented reality* dalam dunia *entertainment*. Buku tersebut merupakan buku cerita untuk anak kecil. Pada buku tersebut terdapat beberapa marker yang berbeda-beda tiap halamannya. Kemudian dari marker tersebut akan muncul karakter binatang beserta cerita nya.

2.4. Syarat marker kode AR yang digunakan

Menurut *Martin Hirzer (2008), Marker* sebaiknya memiliki *setidaknya 4* point yang berbeda agar dapat dilakukan perhitungan posisi. Pada umumnya berbentuk *quadrilateral outline*, dan ke-empat tepi sudut tersebut digunakan untuk estimasi pose 3D. *Marker* Kode AR yang digunakan dalam tugas akhir ini adalah *marker* yang memiliki spesifikasi seperti terlihat pada Gambar 2.7:

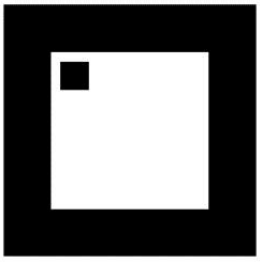
- *Bitonal Marker* (terdiri dari kombinasi hitam dan putih)
- Memiliki garis tepi yang berbentuk persegi
- Memiliki bidang berwarna putih ditengahnya

Contoh *marker* sebagai berikut:



Gambar 2.7. Spesifikasi minimal *marker*

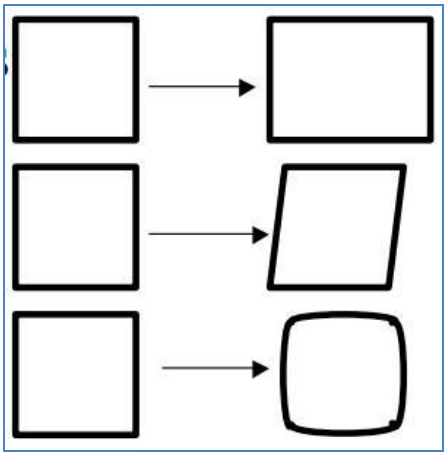
Spesifikasi diatas merupakan spesifikasi standar yang harus ada. Oleh karena bentuk fitur yang simetris, maka apabila dilakukan rotasi marker, penentuan lolasi awalnya sulit untuk ditentukan. Sehingga diperlukan suatu fitur khusus seperti pada Gambar 2.8



Gambar 2.8. Contoh *marker* yang memiliki fitur khusus

2.5. *Camera Calibration*

Terkadang output yang dikeluarkan oleh lensa kamera tidak sama dengan tampilan sebenarnya. Hasil citra yang ditangkap oleh kamera berubah bentuk, baik dikarenakan oleh titik fokus lensa, *skew factor*, *scaling factor* maupun *lens distortion* seperti pada Gambar 2.9.



Gambar 2.9. Output kamera yang terdistorsi

Oleh karena itu diperlukanlah *camera calibration* untuk mendapatkan beberapa parameter dari kamera digital yang digunakan. Parameter ini meliputi matrix intrinsik dan ekstrinsik yang nantinya digunakan untuk melakukan perhitungan, sehingga dapat ditentukan letak dari suatu benda dalam ruang 3 dimensi.

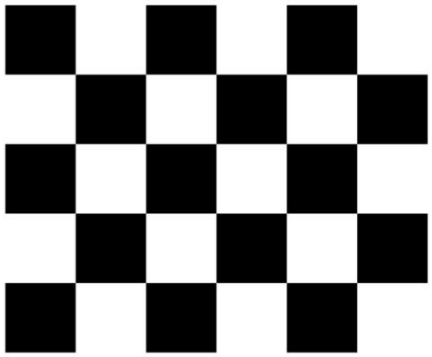
Parameter dari matrix intrinsic terdiri 4 unsur yaitu :

- Nilai fokus kamera, yaitu jarak antara lensa kamera dengan bidang gambar.
- Titik pusat proyeksi, yaitu lokasi titik tengah gambar dalam pixel koordinat

- Ukuran pixel efektif.
- Koefisien distorsi, yaitu koefisien tingkat kelengkungan lensa meliputi radial dan tangensial distorsi.

Sedangkan parameter ekstrinsik terdiri dari 2 buah matrix, yaitu matrix translasi dan matrix rotasi. Parameter ekstrinsik ini menggambarkan orientasi posisi dari kamera terhadap sistem koordinat sebenarnya dalam ruang 3 dimensi atau *world coordinate*.

Untuk melakukan proses kalibrasi kamera, digunakan papan catur (*chessboard pattern*) seperti pada Gambar 2.10



Gambar 2.10. Chessboard pattern

Intrinsic camera adalah hubungan antara sentris koordinat kamera dan koordinat gambar. Sistem koordinat kamera berawal di pusat proyeksi, sumbu z di sepanjang sumbu optik, dan x dan sumbu y sejajar dengan x dan y sumbu gambar. Koordinat kamera dan koordinat gambar berhubungan dengan persamaan proyeksi sebagai berikut:

$$\frac{x\mathbf{1} - x\mathbf{0}}{f} = \frac{xc}{zc} \quad \text{dan} \quad \frac{y\mathbf{1} - y\mathbf{0}}{f} = \frac{yc}{zc} \qquad (1)$$

Sumber: *Berthold K.P. Horn* (2000)

Dimana f adalah jarak dari pusat proyeksi ke gambar, dan (x_0,y_0) adalah titik utama (kaki tegak lurus) dari pusat proyeksi ke bidang gambar. Artinya, pusat proyeksi di $(x_0,y_0)^T$, seperti yang diukur dalam sistem koordinat gambar

Sedangkan *extrinsic camera* adalah hubungan antara sistem koordinat *scene* pusat dan sistem koordinat pusat kamera. Transformasi dari *scene* ke kamera terdiri dari rotasi dan translasi. Transformasi ini memiliki enam derajat kebebasan, 3 untuk rotasi dan 3 untuk translasi.

Jika rs adalah koordinat titik yang diukur dalam sistem koordinat lokasi dan rc diukur dalam sistem koordinat kamera, maka:

$$Rc = R\,(Rs) + t \tag{2}$$

Dimana t adalah translasi dan R (…) adalah rotasi.

Jika memilih untuk menggunakan matrix *orthonormal* untuk mewakili rotasi, maka dapat ditulis :

$$\begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} x_S \\ y_S \\ z_S \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \tag{3}$$

Dimana Rc = $(Xc,Yc,Zc)^T$, rs = $(Xs,Ys,Zs)^T$, dan t = $(tx,ty,tz)^T$.

(*University of Maryland*, "*Camera Callibration*", n.d)

2.6. *Smoothing*

Langkah pertama yang harus dilakukan dalam melakukan *Line Detection* adalah melakukan pengurangan *noise*. Tidak dapat dipungkiri, gambar yang didapatkan dari *webcam* memiliki beberapa *noise. Noise* tersebut apabila dibiarkan akan mempengaruhi hasil dari pengolahan gambar. Baik dalam hal pengenalan maupun pencarian garis tepi. Oleh karena itu *noise* tersebut harus dikurangi. proses pengurangan *noise* menggunakan *filter Gaussian*. *Kernel* yang biasa dipergunakan seperti pada rumus (4) :

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

(4)

Sumber : http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm

Contoh hasil dari *Gaussian filter* ini dapat ditunjukkan pada gambar 2.11:

Gambar 2.11. Original image dan Smoothing image

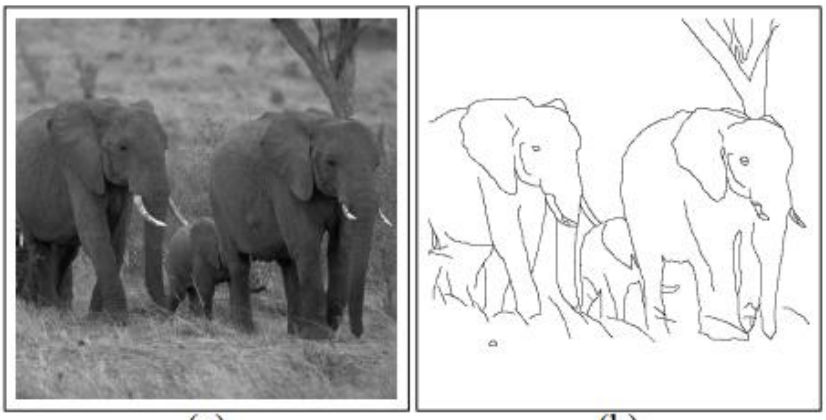
Gambar 2.11 merupakan contoh gambar sebelum di *smoothing* dengan gambar sesudah di-*smoothing* dengan metode *gauss*

2.7. Contour Detection

Bagi manusia, suatu *image* bukanlah hanya sebatas kumpulan pixel yang acak, melainkan sebagai suatu kumpulan area dan objek. Manusia mampu untuk membedakannya dengan mudah. Tetapi lain halnya bagi komputer, komputer tidak dapat mendeteksi dengan sendirinya.

Image segmentation dapat dibedakan menjadi 2 pendekatan yang berbeda. Yaitu *region-based* dan *contour-based*. Pendekatan *Region-based* menggunakan cara pencarian perbedaan *pixel* dalam *image* tergantung pada beberapa factor seperti *brightness*, *color* dan *texture*. Sedangkan pendekatan *contour-based* memulai dengan *edge detection*, diikuti dengan proses penyambungan dan mencari kontinuitas lengkung(Serge B.,Thomas Leung, Jianbo S.,2001).

Menurut gopal Datt Joshi, 2009, pemetaan kontur dari suatu image merupakan representasi yang efisien. Karena hanya mengandung informasi yang menonjol. Sehingga lebih berharga dan berguna dalam *computer vision*.



Gambar 2.12. Perbedaan gambar asli dengan kontur nya
Sumber : http://cvit.iiit.ac.in/papers/Gopal06Asimple.pdf

Pada tugas akhir ini, pendeteksian *marker* akan menggunakan pendekatan yang berbasis kontur. Sebelum mencari kontur dari suatu *image*, akan dilakukan proses pemisahan antara objek latar belakang (*background*) dengan objek latar depan (*foreground*) dengan menggunakan metode *otsu thresholding*. Metode *otsu thresholding* akan dijelaskan lebih detil pada subbab **Error! eference source not found.**.

2.8. Tranformasi

Transformasi merupakan suatu proses untuk melakukan perubahan pada titik-titik koordinat pada objek yang ada. Proses transformasi ini terdiri dari beberapa proses diantaranya proses translasi, rotasi dan penskalaan.

Pada pembuatan aplikasi Tugas Akhir ini proses transformasi yang digunakan hanya meliputi proses translasi dan rotasi pada meja datar saja. Translasi adalah memindahkan posisi objek dari posisi awal menuju posisi tertentu sejauh x terhadap sumbu x, dan sejauh z terhadap sumbu z. Rotasi adalah memutar objek searah maupun berlawanan arah jarum jam.

Jenis transformasi yang sering digunakan dalam komputer grafik adalah *affine transformation* (Hill,1990). Transformasi *affine* lebih sering digunakan

karena memiliki struktur matrix yang sederhana dan mudah dikombinasikan menjadi satu kesatuan dengan keseluruhan transformasi *affine*.

Affine transformation termasuk transformasi linear yaitu transformasi yang menghasilkan formasi yang sama walaupun urutan transformasinya diubah.transformasi *affine* mempunyai keuntungan, yaitu mempertahankan bentuk garis, mempertahankan parallelism (garis yang sejajar akan tetap sejajar setelah ditransformasikan) dan mempertahankan proporsi jarak objek.

2.8.1. Scaling

Scaling adalah mengubah ukuran objek dengan skala tertentu. Karena objek dimodelkan dengan menggunakan koordinat kartesian, maka pengubahan bentuk objek bisa dilakukan pada masing-masing sumbu. Dengan asumsi bahwa objek diskala sebesar sx searah sumbu x, sy searah sumbu y dan sz searah sumbu z, maka hasil penyekalaan dapat dijabarkan sebagai berikut.

X baru = sx * X awal

Y baru = sy * Y awal

Z baru = sz * Z awal

Apabila dibentuk sebagai matrix maka akan menjadi

$$\begin{pmatrix} x_{baru} \\ y_{baru} \\ z_{baru} \end{pmatrix} = \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & sz \end{bmatrix} \begin{pmatrix} x_{awal} \\ y_{awal} \\ z_{awal} \end{pmatrix} \quad (5)$$

2.8.2. Rotation

Rotasi adalah memutar objek pada sebuah sumbu acuan. Untuk objek 2D, maka Cuma terdapat satu jenis putaran. Namun pada objek 3D, terdapat 3 jenis putaran, yaitu putaran searah sumbu x,y dan z. Rotasi tidak mengubah bentuk, namun memindahkan posisi objek. Rotasi sebesar θ derajat arah berlawanan jam, terhadap sumbu x dapat dijabarkan sebagai berikut.

X baru = X lama

Y baru = cos (θ) * Y awal + sin(θ) * Z awal

Z baru = -sin(θ) * Y awal + cos(θ) * Z awal

Apabila dibentuk sebagai matrix maka akan menjadi

$$\begin{pmatrix} x_{baru} \\ y_{baru} \\ z_{baru} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & sin(\theta) \\ 0 & -sin(\theta) & cos(\theta) \end{bmatrix} \begin{pmatrix} x_{awal} \\ y_{awal} \\ z_{awal} \end{pmatrix}$$

$$\begin{pmatrix} x_{baru} \\ y_{baru} \\ z_{baru} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & sin(\theta) \\ 0 & -sin(\theta) & cos(\theta) \end{bmatrix} \begin{pmatrix} x_{awal} \\ y_{awal} \\ z_{awal} \end{pmatrix} \quad (6)$$

2.8.3. Translation

Translasi adalah perpindahan posisi karena adanya pergeseran. Dalam koordinat kartesian, pergeseran yang dimaksudkan adalah pergeseran searah masing-masing sumbu, yaitu sumbu x,y dan z (jika dimaksudkan 3 dimensi). Secara sederhana, pergeseran posisi searah sumbu tertentu bisa dilakukan dengan cara menambah atau mengurangi nilai pada sumbu acuan geser tersebut. Misalnya, bergeser kearah sumbu x positif sebesar 5, berarti nilai x pada vector posisi ditambahkan dengan 5.

Jika translasi ini dibentuk dalam matrix, maka akan mengalami kesulitan, karena semua operasi pada matrix berupa perkalian, bukan penambahan. Berikut adalah penjabaran dari translasi sebesar tx searah sumbu x, ty searah sumbu y dan tz searah sumbu z.

X baru = tx + X awal

Y baru = ty + Y awal

Z baru = tz + Z awal

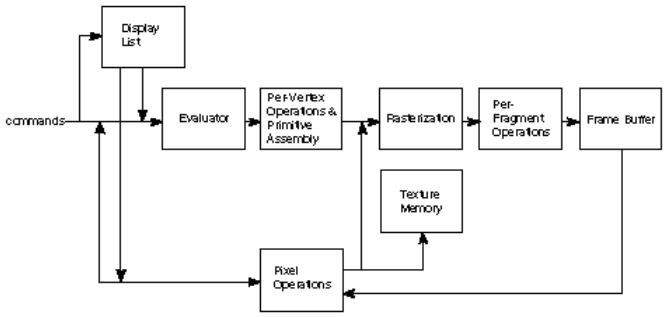
Penjabaran tersebut diatas tidak bisa diubah dalam bentuk matrix. Oleh sebab itu, diperlukan koordinat homogenous. Pada koordinat ini, ditambahkan sebuah nilai pada vector. Nilai ini bukan menyatakan posisi tetapi merupakan pembagi dari ketiga nlai lainnya. Secara umum vector homogenous dituliskan sebagai (xω,yω,zω, ω). Untuk mengubah koordinat homogenous kembali menjadi koordinat kartesius, maka tinggal membagi tiga komponen pertama dengan komponen keempat, setelah itu komponen

keempat dihilangkan. Dengan menggunakan vector homogenous, dan nilai ω yang selalu satu, maka translasi dapat ditulis menjadi

$$\begin{pmatrix} x_{baru} \\ y_{baru} \\ z_{baru} \\ \omega_{baru} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_{awal} \\ y_{awal} \\ z_{awal} \\ \omega_{awal} \end{pmatrix} \quad (7)$$

2.9. **Pustaka Grafis OpenGL**

OpenGL dibuat sebagai sebuah perangkat lunak antar muka yang bisa diimplementasikan pada banyak perangkat keras yang berlainan. *OpenGL* tidak menyediakan perintah tingkat tinggi untuk menggambar model tiga dimensi seperti sebuah pesawat, mobil dan manusia. Untuk menggambar model tiga dimensi di atas dengan *OpenGL* dapat dibuat dari sebuah *geometric primitives* (kelompok objek dasar) seperti titik, garis dan poligon. *OpenGL* mempunyai sebuah *library* (GLU) yang mendukung untuk pembuatan model tiga dimensi.



Gambar 2.13. Proses Penggambaran *OpenGL*
(Sumber : "Blue Book OpenGL" – Addison - Wesley Publishing Company)

OpenGL adalah antarmuka perangkat lunak untuk perangkat keras dirancang sebagai sebuah antarmuka independen yang dapat digunakan untuk berbagai macam platform perangkat keras. Dengan OpenGL dan library bantuan (GLU, GLUT, GLEW, OSG, SDL) untuk windowing serta user input, programmer grafis mampu membuat berbagai aplikasi grafis dengan tampilan efek 2D atau 3D. Proses di OpenGL diawali dengan pengertian transformasi

objek 3D. Proses di OpenGL diawali dengan pengertian transformasi objek 3D ke sistem koordinat layar dalam OpenGL. Tujuan utama dari sistem *OpenGL* adalah menterjemahkan koordinat 3D dari suatu objek ke citra 2D untuk dapat ditampilkan ke layar. Proses menterjemahkan ini dapat dilakukan jika nilai sistem koordinatnya telah sama (homogen).

Koordinat homogeny,diperkenalkan oleh August Ferdinand Mobius, untuk membuat grafik dan perhitungan geometri proyektif yang mungkin dalam ruang. Koordinat Homogen adalah suatu cara untuk mewakili ruang N berkoordinasi dengan angka dimensi N+1. Untuk membuat homogeny 2D koordinat, diperlukan menambahkan vairabel tambahan, w, ke koordinat yang ada. Oleh karena itu, sebuah titik dalam koordinat Cartesian, (X, Y) menjadi (x , y , w) di koordinat homogeny. Dan X dan Y di Cartesian adlaah dinyatakan kembali dengan x,y dan w sebagai suatu bentuk persamaan dalam membetuk koordinat homogeny sebagai berikut :

$$X = x / w \tag{8}$$

$$Y = y / w \tag{9}$$

Sebagai contoh, sebuah titik dalam Cartesian (1 , 2) menjadi (1 , 2 , 1) di koordinat homogeny. Jika titik (1 , 2), bergerak ke arah tak terbatas, hal tesebut akan menjadi (∞ , ∞) dalam koordinat Cartesian. Dan menjadi (1, 2 , 0) di koordinat homogeny, karena (1/0,2/0) =(∞,∞).

Proses pengubahan dari homogeny koordinat (x,y,w) ke koordinat Cartesian, maka dapat dilakukan dengan cara melakukan proses pembagian antara nilai x dan y oleh w;

$$(x,y,w) \Leftrightarrow \left(\frac{x}{w}, \frac{y}{w} \right) \tag{10}$$

Sebagai penjelasan sederhana dari proses tersebut adalah sebagai berikut :

$$(1,2,3) \quad \Rightarrow \left(\frac{1}{3}, \frac{2}{3} \right) \tag{11}$$

$$(2,4,6) \quad \Rightarrow \left(\frac{2}{6}, \frac{4}{6} \right) = \left(\frac{1}{3}, \frac{2}{3} \right) \tag{12}$$

$$(4,8,12) \Rightarrow \left(\frac{4}{12}, \frac{8}{12}\right) = \left(\frac{1}{3}, \frac{2}{3}\right) \quad (13)$$

$$(1a,2a,3a) \Rightarrow \left(\frac{1a}{3a}, \frac{2a}{3a}\right) = \left(\frac{1}{3}, \frac{2}{3}\right) \quad (14)$$

Hasil pengamatan dari proses tersebut diatas adalah titik – titik (1,2,3),(2,4,6) dan (4,8,12) sesuai dengan titik *Euclidean* yang memiliki nilai yang sama (1 / 3 , 2 / 3). Dan setiap produk sklar, (1a,2a,3a) adalah titik yang sama seperti (1/3 ,2/3) di ruang *Euclides*. Oleh karena itu, titik – titik ini disebut dengan homogeny, karena mereka mewakili titik yang sama pada ruang *Euclides (*atau Cartesian ruang). Dengan kata lain, koordinat homogeny adalah invariant skalanya. Koordinat homogeny sangat berguna dan konsep fundamental dalam komputer grafis, seperti memproyeksikan sebuah *scene* 3D (tiga dimensi) ke bidang permukaan 2D (dua dimensi).(Bowo Dwi Ariyanto,2012)

2.9.1. Matrix Transformasi *OpenGL*

OpenGL menggunakan 4 x 4 matrix untuk transformasi. Dapat dilihat pada gambar dibawah ini, 16 elemen dalam matrix disimpan sebagai array pada satu kolom matrik. Matrik transpose digunakan untuk mengubah menjadi konvensi standar format baris utamanya.

$$\begin{bmatrix} m0 & m4 & m8 & m12 \\ m1 & m5 & m9 & m13 \\ m2 & m6 & m10 & m14 \\ m3 & m7 & m11 & m15 \end{bmatrix}$$

Gambar 2.14. Matrix transformasi

Berdasarkan Gambar 2.14, 3 elemen matrix kolom paling kanan (m12, m13,14) adalah untuk terjemahan transformasi proyektif. 3 elemen set, (m0,m1,m2), (m4,m5,m6) dan (m8,m9,m10) adalah untuk *Euclid* dan tranformasi *affine*, seperti rotasi *glRotatef()* atau skala *glScalef()*.

2.10. Open source Computer Vision (OpenCV)

OpenCV adalah singkatan dari Intel® *Open Source Computer Vision Library*. Merupakan kumpulan dari struktur data dan fungsi-fungsi dalam bahasa pemrograman C serta beberapa *class* C++, yang menggunakan prinsip pemrograman berorientasi objek atau lebih dikenal dengan sebutan OOP (*Object Oriented Programming*). Kumpulan struktur data, fungsi, dan *class* tersebut merupakan implementasi dari beberapa algoritma populer yang sering digunakan pada aplikasi *Computer Vision* dan dikelompokkan dalam beberapa kategori, sesuai dengan penjelasan dari *Intel Open Source Computer Vision Library – Reference Manual*, yaitu sebagai berikut:

Basic Structure and Operations, yang dibagi lagi menjadi beberapa kategori sebagai berikut:

Image Function, seperti: pembuatan *header image* dan alokasi data (cvCreateImage), penghapusan *image* dari memori (cvReleaseImage), menentukan ROI (cvSetImageROI), dan sebagainya. Perlu diperhatikan bahwa terdapat beberapa fungsi yang memiliki kegunaan yang sama antara IPL dan OpenCV.

Drawing Primitives, seperti: menggambar garis (cvLine), mencetak *string* (cvPutText), dan sebagainya.

Utility Function, seperti: memisahkan warna *image* ke dalam masing-masing *plane* yang terpisah (RGB, HSV) dengan fungsi cvCvtPixToPlane, melakukan *fill image* dengan suatu nilai konstan (cvFillImage), dan sebagainya.

Motion Analysis and Object Tracking, seperti: melakukan pelacakan warna (cvCamShift), menghitung *optical flow* antara dua *image* dengan metode *Lukas Kanade* (cvCalcOpticalFlowLK), dan sebagainya.

Image Analysis, seperti: menghitung nilai *mean* pada *image region* (cvMean), menghitung nilai histogram *image* (cvCalcHist), dan sebagainya.

Object Recognition, berisi fungsi-fungsi yang digunakan pada sistem pengenalan objek, seperti: menghitung nilai *likelihood*/kemiripan (cvEViterbi) pada metode EHMM, menghitung koefisien *decomposition* dari input objek (cvCalcDecompCoeff) pada metode PCA, dan sebagainya.

Struktur data utama yang digunakan pada sebagian besar fungsi IPL dan *OpenCV* yaitu struktur iplImage yang menyimpan informasi dari sebuah *image*, karena itu semua format citra (BMP, JPEG, TIFF, PNG) yang digunakan sebagai input harus diubah terlebih dahulu ke dalam struktur iplImage. Tabel 2.1. berikut menjelaskan struktur dari iplImage:

Struktur IplImage

```
typedef struct _IplImage {
int nSize /* size of iplImage struct */
int ID /* image header version */
int nChannels;
int alphaChannel;
int depth; /* pixel depth in bits */
```

Struktur IplImage (Sambungan)

```
char colorModel[4];
char channelSeq[4];
int dataOrder;
int origin;
int align; /* 4- or 8-byte align */
int width;
int height;
struct _IplROI *roi; /* pointer to ROI if any */
struct _IplImage *maskROI; /*pointer to mask ROI if any */
void *imageId; /* use of the application */
struct _IplTileInfo *tileInfo; /* contains information on tiling
*/
int imageSize; /* useful size in bytes */
char *imageData; /* pointer to aligned image */
int widthStep; /* size of aligned line in bytes */
int BorderMode[4]; /* the top, bottom, left,
and right border mode */
int BorderConst[4]; /* constants for the top, bottom,
left, and right border */
char *imageDataOrigin; /* ptr to full, nonaligned image */
} IplImage;
```

Field width dan height menunjukkan panjang dan lebar citra dalam piksel. Sedangkan *field* depth berisi informasi tipe dari nilai pixel, yaitu:

IPL_DEPTH_8U - *unsigned 8-bit integer value* (*unsigned char*),

IPL_DEPTH_8S - *signed 8-bit integer value* (*signed char* atau *char*),

IPL_DEPTH_16S - *signed 16-bit integer value* (*short int*),

IPL_DEPTH_32S - *signed 32-bit integer value* (*int*),

IPL_DEPTH_32F - *32-bit floating-point single-precision value* (*float*).

Field nChannels menunjukkan jumlah *color plane* pada citra, dimana citra *grayscale* memakai satu *channel*, sedangkan citra berwarna biasanya terdiri dari tiga atau empat *channel*. *Field* origin menunjukkan letak data nilai piksel citra di *memory*, apakah dimulai dari baris atas citra/*top row* (*origin* == IPL_ORIGIN_TL)

atau dari baris bawah citra/*bottom row* (*origin* == IPL_ORIGIN_BL). Windows *bitmap* biasanya bertipe *bottom-origin*, sedangkan format citra lain pada umumnya bertipe *top-origin*.

Field dataOrder menunjukkan tipe *color planes* pada citra berwarna antara *interleaved* (*dataOrder* == IPL_DATA_ORDER_PIXEL) atau terpisah/*separate* (*dataOrder* == IPL_DATA_ORDER_PLANE), seperti ditunjukkan pada Tabel 2.2. berikut:

Data Order Nilai Piksel dari Struktur IplImage

Data Ordering	**Description**	**RGB Example (channel ordering = RGB)**
Pixel-oriented	All channels for each pixel are clustered.	RGBRGBRGB (line 1) RGBRGBRGB (line 2) RGBRGBRGB (line 3)
Plane-oriented	All image data for each channel is contiguous followed by the next channel.	RRRRRRRRR (line 1) RRRRRRRRR (line 2) R plane RRRRRRRRR (line 3) GGGGGGGGG (line 1) GGGGGGGGG (line 2) G plane

		GGGGGGGGG (line 3) ...

Field widthStep berisi jumlah *byte* antar dua piksel pada kolom yang sama dengan baris yang berurutan atau dengan kata lain lebar citra dalam jumlah *byte*. *Field* width tidak bisa dipakai untuk menghitung jarak (nilai *byte*) tersebut karena pada tiap baris bisa terdapat tambahan beberapa *byte* informasi untuk mempercepat *image processing*, sehingga terdapat jarak/gap antara akhir dari baris ke-i dan awal dari baris ke i+1. *Field* imageData berisi *pointer* ke baris pertama dari *image data*, jika terdapat beberapa *plane* pada *image* (jika *field* dataOrder == IPL_DATA_ORDER_PLANE), masing-masing akan ditempatkan pada *image* terpisah dengan total baris height*nChannels. Dapat dipilih daerah fokus pada *image* (berbentuk kotak) atau *color plane* tertentu saja, atau gabungan keduanya dan proses pada *image* hanya akan dilakukan pada daerah tersebut. Daerah fokus pada *image* disebut "*Region of Interest*" atau ROI. Pada struktur IplImage terdapat *field* roi yang mendukung hal ini. Jika *pointer* tidak *NULL*, maka akan menunjuk ke struktur IplROI yang berisi parameter informasi daerah ROI (seperti ditunjukkan padaTabel 2.4. berikut), jika *NULL* maka dianggap daerah ROI mencakup keseluruhan *image*.

Struktur IplROI

```
typedef struct _IplROI {
int coi; /* channel of interest or COI */
int xOffset;
int yOffset;
int width;
int height;
} IplROI;
```

Tabel 2.5 berikut akan menjelaskan deskripsi dari tiap-tiap *field* pada struktur IplImage dan atribut nilai yang didukung.

Nilai *Field* pada Struktur IplImage

Description	**Value**
Size of the IplImage header (for internal use)	nSize in bytes
Image Header Revision ID (internal use)	ID number
Number of Channels	1 to N
	(Including alpha channel, if any)
Alpha channel number	N (0 if not present)
Bits per channel	
Gray only	IPL_DEPTH_1U (1 bit)
All images: color, gray, and multi-spectral	IPL_DEPTH_8U (8 bit unsigned)
(The signed data is used only as output for some image output operations)	IPL_DEPTH_8S (8 bit signed)
	IPL_DEPTH_16U (16 bit unsigned)
	IPL_DEPTH_16S (16 bit signed)
	IPL_DEPTH_32S (32 bit signed)
	IPL_DEPTH_32F (32 bit float)
Color model	4 character string: "Gray", "RGB", "RGBA", "CMYK", etc.
Channel sequence	4 character string, can be: "G", "GRAY", "BGR", "BGRA", "RGB", "RGBA", "HSV", "HLS", "XYZ", "YUV", "YCr", "YCC", or

	"LUV"
Data Ordering	IPL_DATA_ORDER_PIXEL
	IPL_DATA_ORDER_PLANE
Origin	IPL_ORIGIN_TL (top left corner)
	IPL_ORIGIN_BL (bottom left)
Scanline alignment	IPL_ALIGN_DWORD
	IPL_ALIGN_QWORD
Image size: height	Integer
Width	Integer
Region of Interest (ROI)	Pointer to structure
Mask	Pointer to another IplImage
Image size (bytes)	Integer
Image data pointer	Pointer to data
Aligned width	Width (row lengths in bytes) of image padded for alignment
Border mode of the top, bottom, left, right sides of the image	BorderMode[4]
Border constant of the top, bottom, left, right sides of the image	BorderConst[4]
Original image	Pointer to original image data
Image ID	For application use only, ignored by the library
Tiling information	Pointer to IplTileInfo structure

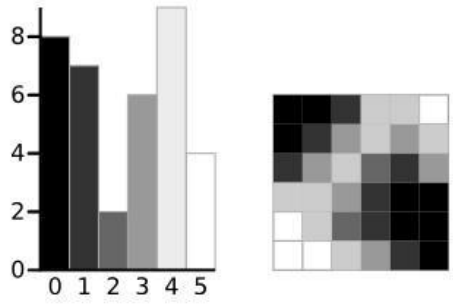
2.11. Otsu Thresholding

Pada lingkungan kondisi yang ideal, pencahayaan yang cukup dan parameter kamera yang konstan, teknik segmentasi sederhana sudah cukup untuk mengatasi nya. Tetapi kebanyakan *image* dari video yang ditangkan sangat rentan

terhadap noise dan perubahan cahaya. Sehingga mengakibatkan level segmentasi yang yang bervariasi. Ketika menghadapai suatu *image*, *static threshold* sudah mencukupi. Pengaturan manual masih memungkinkan. Namun pada video, pengaturan manual merupakan solusi yang kurang baik. Karena pada video, image akan di proses kurang lebih pada 25 frame per second (FPS). Sehingga diperlukan tenik yang secara otomatis mengatur *threshold* tiap-tiap frame nya (Daniel Diggins,2005).

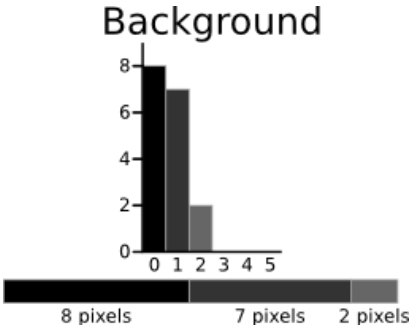
Metode *Otsu Thresholding* melibatkan iterasi melalui semua kemungkinan nilai threshold dan menghitung ukuran penyebaran untuk tingkat pixel setiap sisi ambang batas pixel, baik pixel latar belakang (*background*) ataupun latar depan (*foreground*). Tujuan dari metode ini adalah untuk menentukan nilai minimum *threshold* dari jumlah penyebaran latar belakang (*background*) dan latar depan (*foreground*).

Berkut merupakan contoh *image* 6x6 dengan 6 level *grayscale.*



Gambar 2.15. contoh *image* 6x6
Sumber :
http://www.labbookpages.co.uk/software/imgProc/files/otsuOrig.png

Cara perhitungan varian *background* dan *foreground* dengan nilai *threshold* 3 dapat dilihat pada Gambar 2.16. dan Gambar 2.17



$$\text{Weight} \quad W_b = \frac{8+7+2}{36} = 0.4722$$

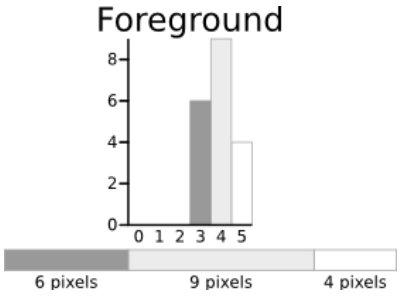
$$\text{Mean} \quad \mu_b = \frac{(0 \times 8) + (1 \times 7) + (2 \times 2)}{17} = 0.6471$$

$$\text{Variance} \quad \sigma_b^2 = \frac{((0 - 0.6471)^2 \times 8) + ((1 - 0.6471)^2 \times 7) + ((2 - 0.6471)^2 \times 2)}{17}$$

$$= \frac{(0.4187 \times 8) + (0.1246 \times 7) + (1.8304 \times 2)}{17}$$

$$= 0.4637$$

Gambar 2.16. Perhitungan *background*



$$\text{Weight} \quad W_f = \frac{6+9+4}{36} = 0.5278$$

$$\text{Mean} \quad \mu_f = \frac{(3 \times 6) + (4 \times 9) + (5 \times 4)}{19} = 3.8947$$

$$\text{Variance} \quad \sigma_f^2 = \frac{((3 - 3.8947)^2 \times 6) + ((4 - 3.8947)^2 \times 9) + ((5 - 3.8947)^2 \times 4)}{19}$$

$$= \frac{(4.8033 \times 6) + (0.0997 \times 9) + (4.8864 \times 4)}{19}$$

$$= 0.5152$$

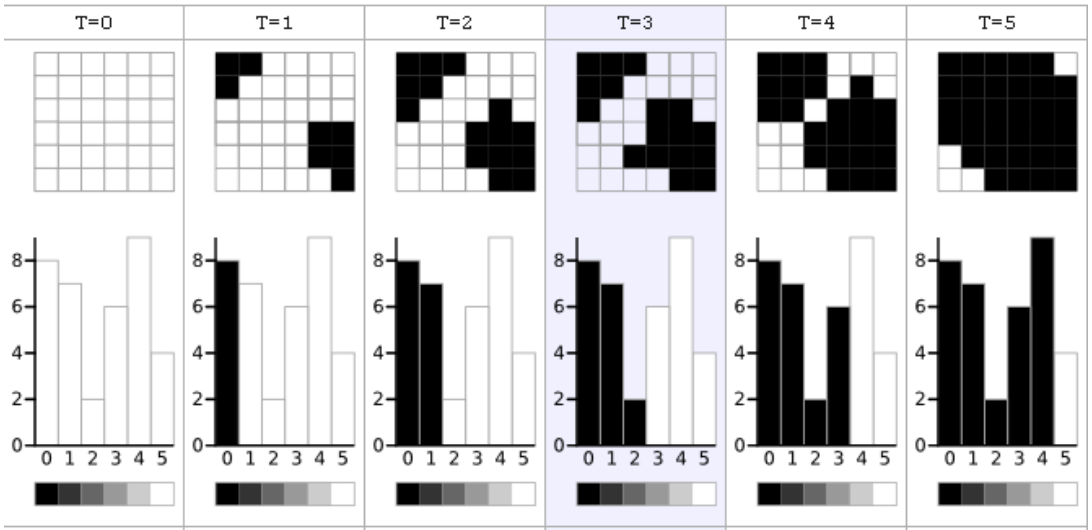
Gambar 2.17. Perhitungan *foreground*

Langkah berikutnya adalah menghitung *class variance* nya

$$\text{Within Class Variance} \quad \sigma_W^2 = W_b\,\sigma_b^2 + W_f\,\sigma_f^2 = 0.4722 * 0.4637 + 0.5278 * 0.5152 = 0.4909$$

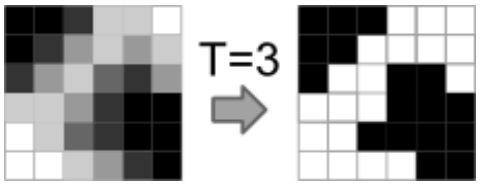
Gambar 2.18. Perhitungan Class variance

Di bawah ini merupakan nilai perhitungan *threshold otsu*.



T=0	T=1	T=2	T=3	T=4	T=5
W_b = 0	W_b = 0.222	W_b = 0.4167	W_b = 0.4722	W_b = 0.6389	W_b = 0.8889
M_b = 0	M_b = 0	M_b = 0.4667	M_b = 0.6471	M_b = 1.2609	M_b = 2.0313
σ^2_b = 0	σ^2_b = 0	σ^2_b = 0.2489	σ^2_b = 0.4637	σ^2_b = 1.4102	σ^2_b = 2.5303
W_f = 1	W_f = 0.7778	W_f = 0.5833	W_f = 0.5278	W_f = 0.3611	W_f = 0.1111
M_f = 2.3611	M_f = 3.0357	M_f = 3.7143	M_f = 3.8947	M_f = 4.3077	M_f = 5.000
σ^2_f = 3.1196	σ^2_f = 1.9639	σ^2_f = 0.7755	σ^2_f = 0.5152	σ^2_f = 0.2130	σ^2_f = 0
σ^2_W = 3.1196	σ^2_W = 1.5268	σ^2_W = 0.5561	**σ^2_W = 0.4909**	σ^2_W = 0.9779	σ^2_W = 2.2491

Gambar 2.19. Gambar perhitungan Otsu Thresholding dengan *within class variance*



Gambar 2.20. perbandingan sebelum dan sesudah *threshold*

Dapat dilihat bahwa nilai minimum class variance nya berada pada posisi 3. Sehingga dapat disimpulkan bahwa semua pixel pada level dibawah 3 adalah *background*, sedangkan sisanya adalah *foreground*.

Dengan sedikit manipulasi formula, maka dapat dirumuskan cara yang lebih cepat, yang disebut dengan *between class variance.*

$$\begin{aligned}\text{Between Class Variance} \quad \sigma_B^2 &= \sigma^2 - \sigma_W^2 \\ &= W_b(\mu_b - \mu)^2 + W_f(\mu_f - \mu)^2 \quad (\text{where } \mu = W_b\,\mu_b + W_f\,\mu_f) \\ &= W_b\,W_f\,(\mu_b - \mu_f)^2\end{aligned}$$

Gambar 2.21. Formula Between Class Variance

Berdasarkan perhitungan yang dilakukan, perhitungan nilai maksimum *Between Class Variance* juga merupakan nilai minimum *Within Class Variance.* Sehingga untuk mempercepat kinerja aplikasi dalam pencarian nilai threshold yang optimal, dapat digunakan formula tersebut. (Dr.Andrew Greensted,2010)

3. METODE PENELITIAN

3.1. Metodologi Penelitian

Sistem perangkat lunak yang dikembangkan untuk pembuatan aplikasi perangkat lunak ini terdiri dari beberapa tahap. Adapun tahapan-tahapan tersebut adalah:

Studi Literatur

Mempelajari buku-buku literatur mengenai *Augmented Reality* dan bahasa pemrograman yang digunakan.

Pembuatan Program

Pembuatan desain class dan program

Melakukan pengujian keakuratan dan kemampuan program dalam mendeteksi *marker*

Menarik Kesimpulan

Merangkum hasil dari pengujian dan memeberikan saran untuk tahap kedepannya.

3.2. Desain Sistem

Pada bab ini akan dibahas perencanaan dan sistem kerja dari perangkat lunak pendeteksi gerak untuk *Augmented reality*. Pembuatan perangkat lunak ini menggunakan *Integrated Development Environtment* (IDE) Visual Studio 2008 dengan bahasa pemrograman C++.

Dalam tugas akhir ini, kamera *webcam* akan digunakan sebagai alat untuk mendapatkan input *image*. Kemudian akan dilakukan pendeteksian *marker* pada gambar yang didapat dari kamera. Setelah *marker* terdeteksi maka dilakukan pencarian fitur *marker*. Fitur tersebut akan digunakan sebagai referensi untuk menentukan posisi titik 0 / titik pusat. Yang nantinya titik tersebut akan digunakan untuk menentukan orientasi yang terjadi.

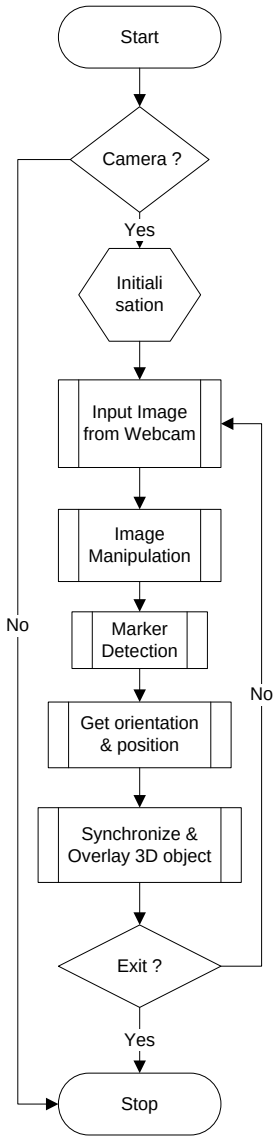
Kemudian akan dilakukan perhitungan estimasi rotasi sudut, yang nantinya akan digunakan untuk proses transformasi pada objek 3D yang ada.

Sehingga objek 3D yang tertera dilayar akan sesuai dengan sudut pandang atau perspektif posisi *marker* yang tampak di kamera.

Sistem perangkat lunak yang dikembangkan untuk proses pendeteksian gerak untuk *Augmented reality* ini terdiri dari beberapa tahap. Mulai dari *preprocessing* yaitu *camera calibration* untuk memperbaiki citra gambar, proses pencarian *marker* pada papan acuan, pencarian rotasi *marker* dan terakhir menampilkan suatu objek 3D pada *marker* sesuai dengan orientasi nya.

3.2.1. Desain Flochart

Dalam sistem perangkat lunak yang dikembangkan untuk proses pendeteksian, terdapat beberapa tahapan yang harus dilakukan. Adapun garis besar rancangan sistem yang akan digunakan dapat dilihat pada Gambar 3.22



Gambar 3.22. *Flowchart* garis besar perangkat lunak

Perangkat lunak ini menggabungkan dunia nyata dengan dunia virtual, maka dibutuhkan lah inputan *image*. Inputan *image* berasal dari kamera *webcam*. Maka dari itu, hal pertama yang dilakukan adalah mengecek apakah terdapat kamera *webcam*. Apabila tidak ada, maka perangkat lunak tidak akan dapat dijalankan dan akan keluar. Apabila ada, maka perangkat lunak akan melanjutkan ke tahap berikutnya.

Initialization adalah proses inisialiasi variabel-variabel yang akan digunakan dalam perangkat lunak. Seperti men-set nilai flag, mengambil nilai intrinsic camera, menyiapkan matrix dan storage contour.

Input image adalah gambar bergerak yang berasal dari *webcam* yang terhubung dengan sambungan USB. Perangkat lunak akan mendeteksi kamera yang terhubung, lalu mengambil data *image* untuk dipersiapkan untuk tahap selanjutnya

Camera calibration merupakan suatu tahap pilihan untuk melakukan pembetulan output *image* dari *webcam*. Apabila *webcam* yang digunakan dirasa memiliki output yang kurang bagus, seperti terjadi perubahan bentuk yang dikarenakan oleh titik fokus lensa yang kurang tepat, *skew factor*, *scaling factor* maupun *lens distortion.* Maka diperlukan proses *Camera Calibration.* Pada tahap ini akan dilakukan pengambilan 10 *image checkerboard* dengan sudut dan lokasi yang berbeda. Kumpulan data *image* tersebut akan dilakukan perhitungan untuk mendapatkan parameter kamera. Yang kemudian parameter tersebut akan digunakan untuk membetulkan hasil output kamera.

Image Manipulation adalah tahap dimana dilakukan proses manipulasi image sebelum tahap *marker detection*. Pada tahap ini dilakukan perubahan image ke grayscale. Lalu di lakukan proses smoothing untuk menghilangkan noise. Kemudian dilakukan proses binerisasi (image hanya berupa hitam atau putih). Proses berikutnya adalah dilakukan thresholding. Thresholding ini dilakukan untuk memperkecil *range* kontur yang akan diproses oleh *marker detection*. *Thresholding* yang digunakan tidak menggunakan *static thresholding*, melainkan *adaptive thresholding*. Dengan pertimbangan bahwa apabila menggunakan *static*

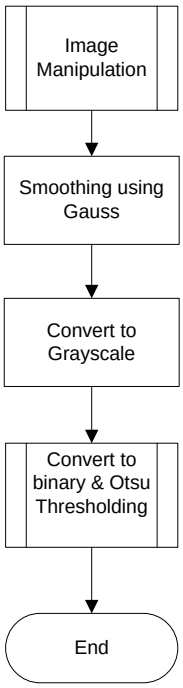
thresholding, maka besar kemungkinan nya bahwa pada suatu kondisi tertentu berhasil, tetapi pada kondisi yang lain tidak dapat terpenuhi.

Marker Detection adalah proses dimana perangkat lunak akan mencari lokasi *marker* dari input *image*. Pendeteksian dilakukan dengan cara mencari setiap bentuk yang sudut tepinya adalah 4. Apabila sudah terdeteksi maka akan dicatat posisi tiap tepinya.

Get orientation & posision of marker, setelah perangkat lunak barhasil mendeteksi *marker* yang ada, maka dapat diketahui posisi *marker* di dunia nyata. Kemudian untuk mencari orientasi *marker*, yang menjadi pusat perhatian (*point of interest*) kita berikutnya adalah fitur bagian dalam (*inner marker*). Dari *inner marker* tersebut aplikasi akan dicari orientasi suatu fitur yang nanti nya akan dipakai sebagai titik pusat / acuan dari *marker* tersebut. Sehingga dari sana dapat dilakukan estimasi rotasi yang terjadi. Cara perhitungan estimasi rotasi adalah dengan menggunakan rumus perhitungan matematis antara garis pusat dengan garis setelahnya terhadap garis *horizontal* dari titik pusat tersebut.

Synchronize & overlay 3D object, merupakan tahap dimana dilakukan pengiriman data *image* dari *webcam* ke *window OpenGL*. Kemudian model objek 3D akan digambar pada *window OpenGL* sesuai dengan rotasi *marker* yang terjadi.

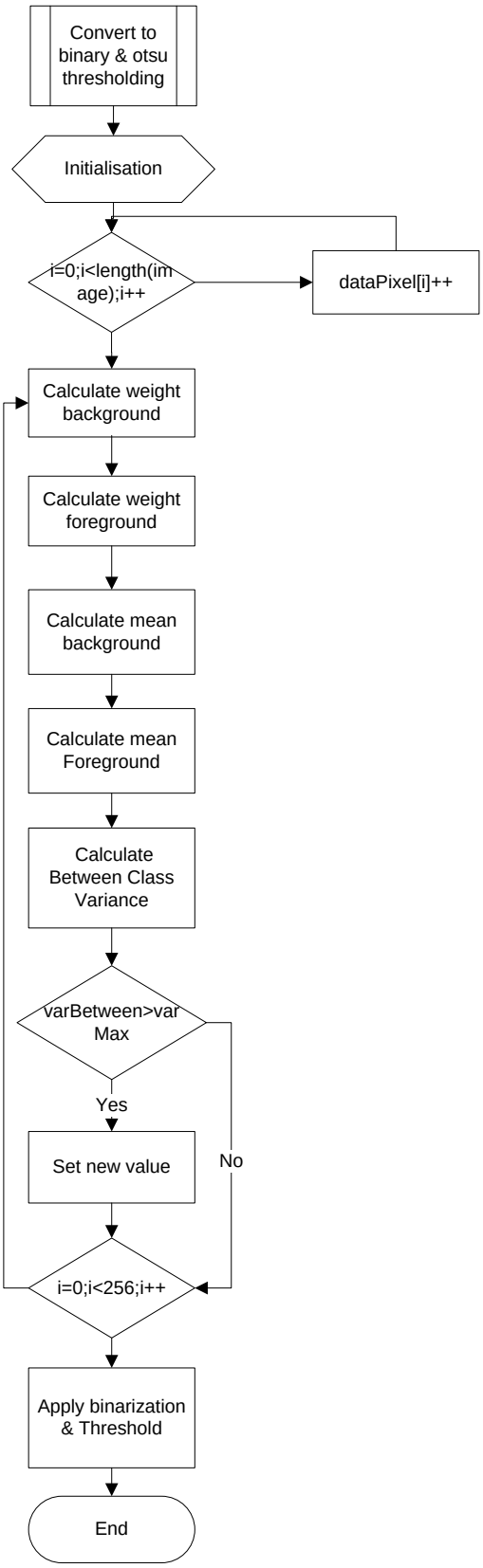
3.2.1.2. Image Manipulation



Gambar 3.23. Flowchart Image Manipulation

Pada tahap ini dilakukan manipulasi *image* yang didapat dari kamera *webcam*. Pertama kali yang dilakukan adalah *smoothing* atau *blurring* dengan metode *gaussian* untuk mengurangi noise yang ada. Tahap berikutnya merubah *channel image*, yang pada mula nya adalah RGB menjadi 1 channel saja. Sehingga didapat gambar grayscale hitam putih. Berikutnya dilakukan proses binerisasi (hanya berwarna hitam atau putih) dan dilakukan prosedur *thresholding*. Proses binerisasi dan *thresholding* ini dilakukan guna mengurangi range area yang harus dicek. Sehingga apabila lingkup area *image* yang dicek semakin sedikit, maka proses pendeteksian pun semakin pendek dan aplikasi pun akan berjalan semakin cepat.

3.2.1.3. *Convert to binary & Otsu Thresholding*



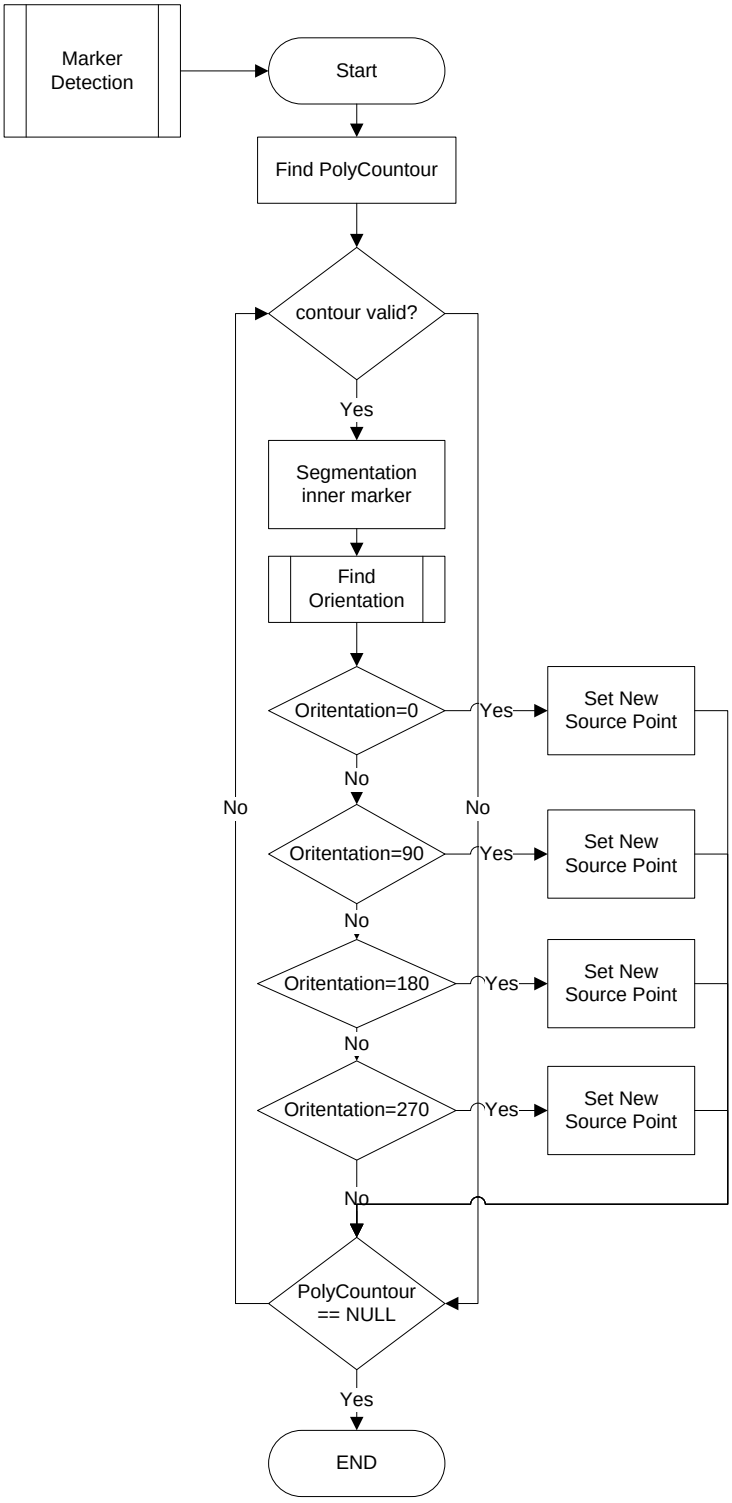
Gambar 3.24. Flowchart convert to binary & otsu thresholding

Pada proses ini, pertama-tama dilakukan pengambilan data pixel *image*. Pengambilan data pixel dengan cara melakukan pengaksesan secara langsung pointer data *image*. kemudian dilakukan *looping* sebanyak panjang *image* untuk menambahkan jumlah nilai setiap pixel dari 0-255.

Kemudian dilakukan *looping tiap*-tiap nilai pixel, untuk menghitung *weight background*, *weight foreground*, rata-rata *background*, rata-rata *foreground*. Yang nantinya akan digunakan untuk mencari nilai *class variance*. Kemudian apabila nilai varian lebih besar dari variabel, maka akan dilakukan update nilai variabel dan nilai threshold. Setelah nilai optimal *threshold* didapat, maka nilai tersebut akan dijadikan patokan nilai *threshold* ke data *image*.

Kemudian untuk menghindari *memory leak*, diakhir proses pencarian *threshold* akan dilakukan releasing variabel yang tidak terpakai.

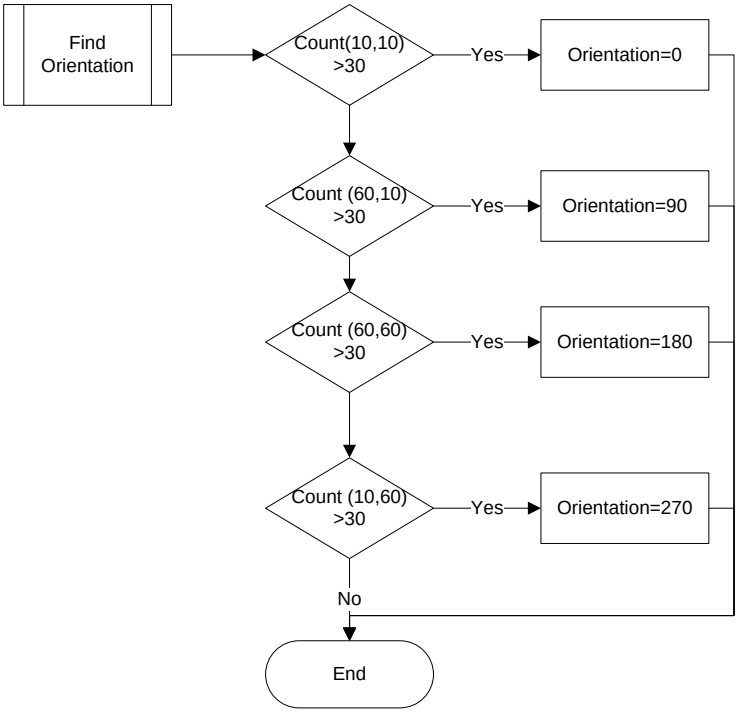
3.2.2. *Marker Detection*



Gambar 3.25. Flowchart marker detection

Pada tahap ini akan dilakukan pencarian *marker*. *Marker* yang dideteksi diharuskan memiliki spesifikasi minimal seperti yang sudah dijelaskan pada bab 2, dengan penambahan suatu fitur berbentuk kotak kecil. Fitur tersebut akan digunakan sebagai referensi untuk menentukan posisi titik 0 / titik pusat. Sehingga kita dapat mengetahui posisi acuan yang nantinya akan digunakan untuk proses penentuan orientasi.

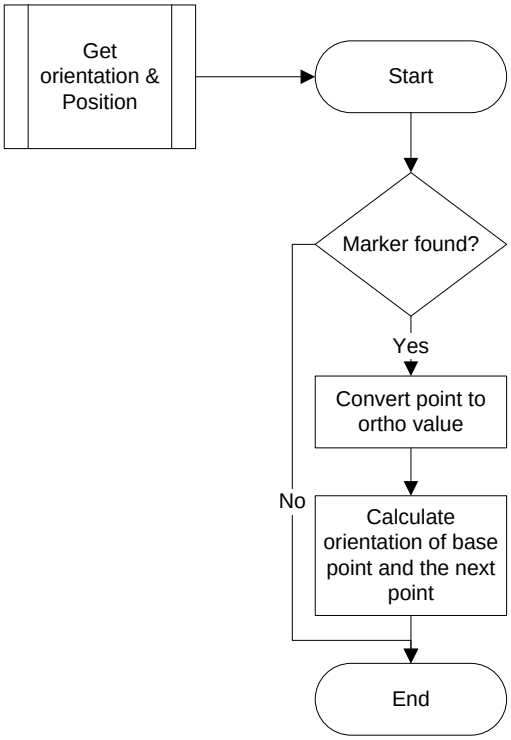
3.2.3. *Find Orientation*



Gambar 3.26. Flowchart Find orientation

Pada tahap ini dilakukan pendeteksian sederhana untuk mencari fitur kotak kecil. Dari pendeteksian fitur tersebut, kita dapat mengetahui bahwa titik itu adalah posisi titik 0 / titik pusat nya. Apabila kita merotasi *marker*, dapat kita ketahui dimana titik pusat yang baru berada. Sehingga dari sana dapat kita cari orientasi *marker* tersebut. Cara pendeteksian dilakukan dengan menghitung jumlah pixel berwarna putih yang ada pada beberapa sudut perputaran. Yaitu apabila sudut putar 0^o ,maka koordinat (10,10) atas kiri dihitung. Untuk perputaran 90^o akan dicari pada koordinat atas kanan (60,10). Untuk 180^o bawah kanan (60,60). Untuk 270^o pada koordinat (10,60) bawah kiri. Apabila jumlah tersebut melebihi nilai yang sudah ditentukan, maka disimpulkan bahwa orientasi *marker* berada pada sudut tersebut.

3.2.4. *Get orientation & position of marker*



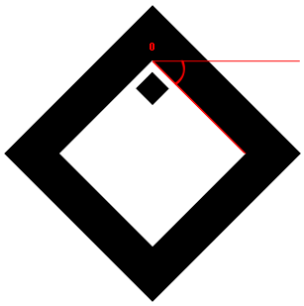
Gambar 3.27. *Flowchart* Get orientation & position

Apabila *marker* tidak ditemukan, maka proses akan berhenti. Jika ditemukan maka proses akan berlanjut.

Oleh karena *OpenGl* menggunakan system koordinat *Cartesian* (koordinat pusat 0,0 di tengah) sedangkan *OpenCV* menggunakan koordinat layar (posisi 0,0 di kiri atas), maka pada tahap ini dilakukan convert nilai dari nilai koordinat layar menjadi koordinat cartesian. Kemudian dilakukan pula estimasi perhitungan rotasi dari perubahan orientasi yang terjadi. Perhitungan sudut dilakukan dengan metode perhitungan sudut antara garis 1 dengan garis 2. Garis 1 adalah garis yang dibentuk oleh titik 0 / titik pusat (yang didapat dari fitur kotak proses sebelumnya) dengan titik 3. Garis 2 adalah garis acuan yang dibuat secara horizontal dari titik pusat seperti pada Gambar 3.28.

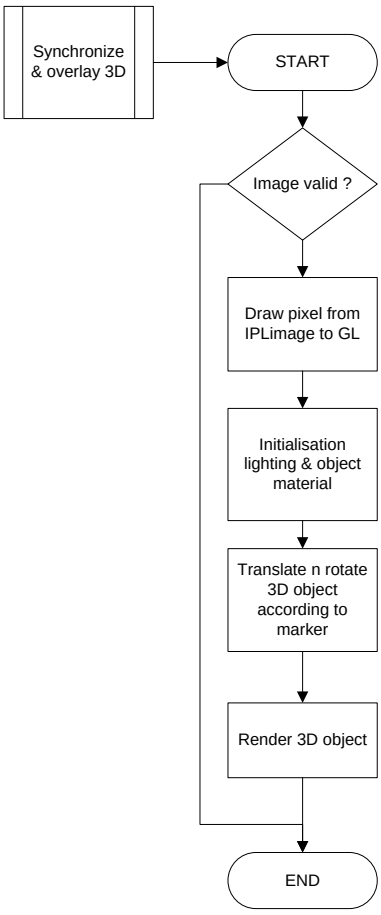
Rumus yang digunakan untuk menghitung 2 garis adalah

$$\theta = arccos\left(\frac{a.b}{|a|.|b|}\right) \quad (1)$$



Gambar 3.28. Garis acuan perhitungan orientasi *marker*

3.2.5. *Synchronize & overlay 3D*



Gambar 3.29. Flowchart synchronize & overlay 3D

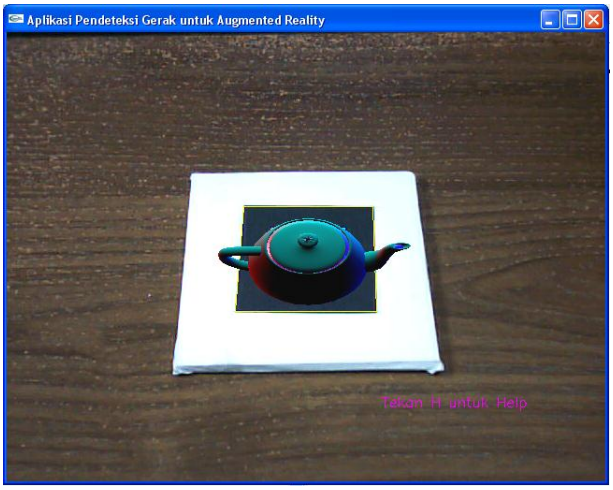
Karena untuk mengambil image dari *webcam* dan pemrosean image menggunakan library *OpenCV*, sedangkan untuk penggambaran objek 3D menggunakan library *OpenGL*, maka diperlukan tahap untuk memindah data dari *OpenCV* ke *OpenGL*. Hal pertama yang dilakukan adalah mengecek validitas *image*, apakah *image* tersebut memiliki *property* yang sesuai atau tidak. Apabila tidak valid, maka tahap ini akan dilewati. Apabila valid, maka akan dilakukan penggambaran *pixel* ke *Window OpenGL*. Yang perlu diperhatikan dalam pemindahan *pixel* ke *OpenGL* adalah *property channel* di *OpenCV* adalah GRB bukan RGB, oleh karena itu perlu sedikit penyesuaian sewaktu pengiriman pixel ke *OpenGL*.

Setelah pemindahan pixel selesai, maka tahap berikutnya yang akan dilakukan adalah menyiapkan pencahayaan dan material objek 3D yang akan digambar agar objek lebih mudah dilihat. Berikutnya dilakukan proses translasi dan rotasi objek 3D terhadap parameter yang dihasilkan oleh *marker*. Kemudian objek 3D siap direnderer ke *window*.

4. HASIL PENELITIAN DAN PEMBAHASAN

4.1. Pengujian Program

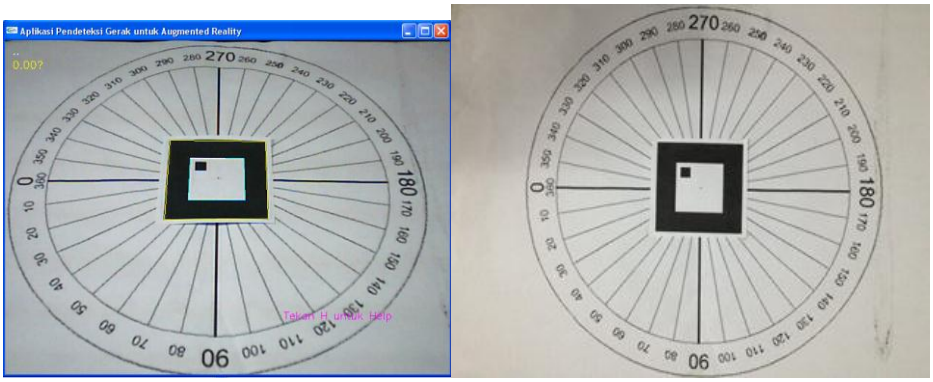
Aplikasi Pendeteksi Gerak untuk *Augmented reality* merupakan aplikasi perangkat lunak utama dalam Tugas Akhir ini. Tampilan dapat dilihat pada Gambar 4.30. Pada saat aplikasi ini berjalan, prosedur pendeteksian *marker* akan berjalan secara otomatis. Apabila suatu objek sudah teridentifikasi sebagai *marker*, maka pada layar akan muncul suatu objek. Objek yang ditampilkan dapat diubah-ubah sesuai dengan keinginan user melalui penekanan tombol keyboard.



Gambar 4.30. Tampilan aplikasi utama

4.2. Pengujian tingkat akurasi rotasi marker pada sumbu pusat

Proses pengujian tingkat akurasi rotasi *marker* pada sumbu pusat dilakukan dengan kondisi jarak dari camera ke *marker* 31 cm dengan ketinggian 37 cm. Dengan rumus phytagoras, maka ditemukan teta sudut kamera adalah 48.27^{o} kearah bawah. Untuk menjaga agar *marker* selalu tepat ditengah ketika percobaan ini dilakukan, maka titik pusat *marker* ditancapkan ke titik pusat kertas derajat dengan menggunakan jarum.



Gambar 4.31. Sudut 0^o gambar dari webcam (kiri) dengan gambar tampak atas (kanan)

Tabel pengujian tingkat akurasi rotasi marker pada sumbu pusat

Sudut real	Sudut estimasi	Selisih
0^o	0^o	0^o
45^o	36.38^o	8.62^o
60^o	69.23^o	-9.23^o
90^o	89.05^o	0.95^o
145^o	140.31^o	4.69
225^o	217.63^o	7.37

Dari Tabel 4.2 dapat dilihat bahwa pada sudut real 0^o dan 90^o, Ketika *marker* selaras dengan axis horizontal maupun vertikal, perhitungan estimasi rotasi yang dihasilkan relative lebih akurat.

Namun pada waktu *marker* diputar diagonal, perhitungan rotasi menjadi tidak akurat. Nilai yang dihasilkan memiliki perbedaan maksimal 8.62^o dan minimal -9.23^o.

4.3. Pengujian jarak optimal pendeteksian *marker*

Pada sub bab ini dilakukan pengujian dengan variabel jarak dan variabel rotasi. Pengujian dengan variabel jarak yaitu, *marker* akan di uji coba pada jarak yang berbeda-beda dimulai dari 20 cm dari kamera, 30 cm, kemudian kelipatan 30

cm hingga 3.6 meter. Sedangkan variabel rotasi adalah *marker* diuji dengan sudut rotasi 0^o, 45^o ,90^o ,135^o ,180^o.

Tabel hasil pengujian jarak dan rotasi marker

		rotasi marker				
		0 °	**45 °**	**90 °**	**135 °**	**180 °**
Jarak (cm)	**20**	359.26	43.64	91.45	133.42	179.68
	30	359.5	42.06	91.53	133.95	180
	60	357.09	42.14	88.06	132.93	177.09
	90	355.71	42.88	86.99	135	175.6
	120	356.05	42.14	87.95	130.82	176.19
	150	3.01	38.66	92.86	131.42	174.56
	180	356.63	39.81	96.71	133.03	183.37
	210	6.34	39.81	96.71	132.51	173.29
	240	3.81	39.81	7.59	42.27	172.41
	270	4.09	45	12.09	39.29	4.09
	300	4.76	45	5.19	34.99	4.76
	330	5.71	36.87	5.71	37.87	5.71
	360	-	-	-	-	-

Tabel 4.3 merupakan tabel hasil pengujian jarak dan rotasi *marker* dengan variabel sumbu-x (horizontal) yaitu rotasi *marker* dalam derajat, variabel sumbu-y (vertical) yaitu jarak *marker* dari kamera dalam satuan cm. sedangkan nilai yang diberikan adalah estimasi sudut yang dihitung oleh aplikasi perangkat lunak. Sedangkan *cell* yang berwarna merah merupakan indicator bahwa estimasi rotasi tidak valid, karena perbedaannya terpaut sangat jauh.

apabila *cell* yang berwarna merah diabaikan, maka dapat ditemukan nilai maximal perbedaan estimasi rotasi oleh perangkat lunak adalah 6.34 ^{o} dan nilai minimum nya adalah -8.13 ^{o}.

Dari data Tabel 4.3 dapat dilihat pada jarak *marker* 20 cm hingga 210 cm dari kamera, *marker* dapat terdeteksi dan dikenali dengan cukup baik. Baik pada

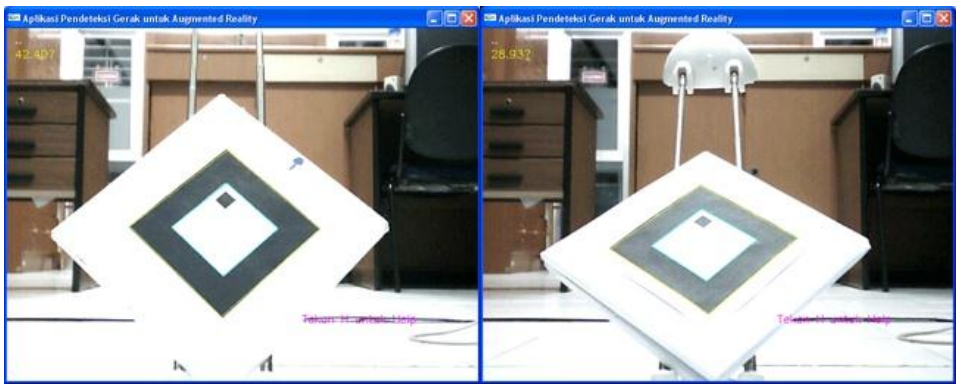
rotasi 0 ^{o}, 45 ^{o}, 90 ^{o},135 ^{o}, dan 180 ^{o}. Nilai perbedaan maksimum yang dihasilkan adalah 6.71 ^{o}, sedangkan nilai perbedaan minimumnya adalah -6.71 ^{o}.

Sedangkan pada jarak 240 cm dengan sudut 90 ^{o} dan 135 ^{o}, terlihat tanda-tanda estimasi rotasi *marker* mulai tidak akurat. Dapat dilihat pada **Error! eference source not found.** kotak kecil yang dipakai sebagai acuan dalam menentukan titik utama mulai tidak terlihat di kamera. Sehingga mengakibatkan proses pengenalan menjadi kacau.

Pada jarak *marker* 270 cm dari kamera, *marker* masih dapat terdeteksi, namun untuk penentuan titik-titik kotak tidak valid. Tingkat error semakin meningkat. Pada sudut 90 ^{o} dan 180 ^{o} dianggap mendekati dengan sudut 0 ^{o} . sedangkan sudut 135 ^{o} dianggap mendekati dengan 45 ^{o}.

4.4. Pengujian tingkat kemiringan *marker* pada jarak dan rotasi tertentu

Pengujian ini merupakan tahap pengujian lebih lanjut dari pengujian sebelumnya. Pada pengujian kali ini faktor rotasi marker juga akan diperhitungkan. Marker akan di miringkan pada sudut 30 ^{o}, 60 ^{o}, 120 ^{o}, 150 ^{o} pada jarak 30 cm hingga 210 cm dan akan dirotasi sebanyak 45 ^{o}



Gambar 4.32. Gambar jarak marker 30 cm dari kamera dengan kemiringan 120^{o} dan 150^{o}

Tabel Perbedaan sudut estimasi rotasi

	Tingkat Kemiringan

		30 ^{o}	60 ^{o}	120 ^{o}	150 ^{o}
Jarak (cm)	30	16.07	2.6	5	9.4
	60	16.93	1.47	3.47	8.37
	90	7.43	5.39	7.85	13.5
	120	15.75	3.99	3.99	9.46
	150	9.78	6.71	9.46	4.09
	180	14.53	3.81	5.19	-
	210	5.19	8.75	9.78	-

Tabel 4.3 merupakan tabel hasil perbedaan sudut estimasi dengan sudut 45^{o}. nilai semakin dekat dengan 0, maka semakin baik. Karena semakin kecil perbedaan nya. Dapat dilihat pada tabel tersebut, pada kemiringan 30^{o}, estimasi rotasi meleset hingga nilai maksimal 16.93^{o}. sedangkan pada kemiringan 60^{o} dengan jarak yang sama (60 cm dari kamera) perbedaanya semakin berkurang. Tetapi pada kemiringan 120^{o} dan 150 ^{o}, estimasi rotasi mengalami peningkatan *error rate*. Hal ini diakibatkan distorsi marker yang membuat perhitungan estimasi tidak presisi.

4.5. Pengujian dengan menggunakan webcam yang berbeda

Pada pengujian ini, akan dilakukan uji coba seperti dengan pengujian 4.2 tetapi dengan menggunakan 3 webcam yang tipe dan resolusinya berbeda. Tujuan dari pengujian ini adalah untuk mengetahui efek dari webcam yang berbeda.

Gambar 4.33. Gambar kamera yang dipakai

Spesifikasi webcam yang digunakan berurutan dari kiri adalah :

1. Webcam intel CS110 dengan resolusi maksimal 320x240
2. logitect quickcam Webcam P/N 861080 dengan resolusi maksimal 320x240
3. Webcam logitech Quickcam Communicate STX P/N 861223-0000. (Webcam yang dipakai pada pengujian-pengujian sebelumnya).

Tabel hasil pengujian dengan 3 kamera

		Orientasi						
		0 ^{o}	45 ^{o}	60 ^{o}	90 ^{o}	145 ^{o}	225 ^{o}	270 ^{o}
kamera	1	0	41.19	70.35	84.09	139.97	214.16	267.95
	2	358.21	40.82	69.44	85.43	137.82/34.82	36.47	358.21
	3	0	36.38	69.23	89.05	140.31	217.63	269.31

Hasil pengujian pada tabel diatas, orientasi 0-90 derajat, dapat dilihat bahwa terdapat perbedaan estimasi rotasi marker. Namun tidak terlalu signifikan.

Akan tetapi pada orientasi 145 ^{o} hingga 270 ^{o}, pada kamera-2 estimasi mulai terlihat kacau. Hal ini dikarenakan spesifikasi kamera yang kurang memadai. Dari hasil pengamatan pada kamera-2, hasil image yang didapat lebih terdapat noise, flicker dan tidak fokus. Hal itu menyebabkan estimasi rotasi pada kamera-2 tidak stabil / gampang berubah.

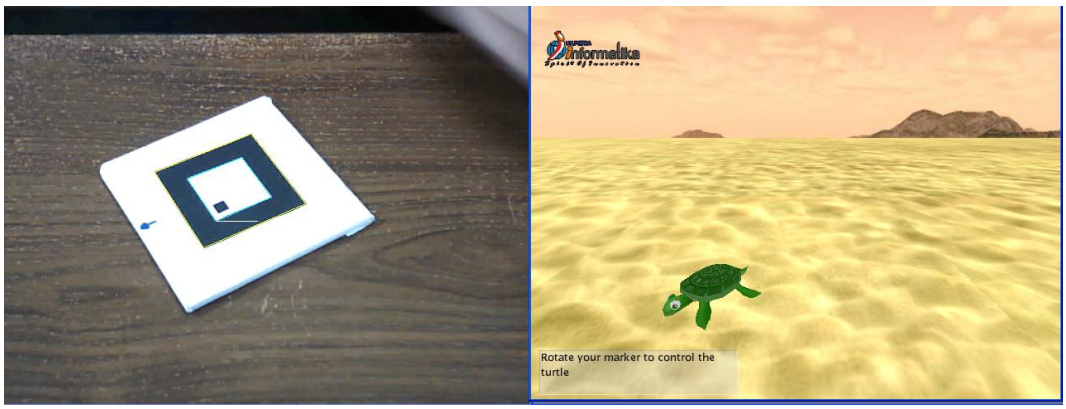
Gambar 4.34. Perbandingan hasil dari kamera-1 dan kamera-2

4.6. Pengujian prototyping game

Pada pengujian ini, core aplikasi marke detection perangkat lunak Tugas Akhir ini akan digabung dengan game engine Irrlicht. *Marker* digunakan untuk

mengontrol objek kura-kura untuk pada suatu *prototype* game sederhana. Input yang digunakan berasal dari kamera *webcam* untuk menangkap image. Kemudian akan dilakukan pendeteksian *marker* dan estimasi rotasi yang terjadi. Nilai estimasi tersebut nantinya akan digunakan untuk mengontrol arah kura-kura.

Gambar 4.35. Gambar dari *webcam (kiri)* dan gambar objek kura-kura yang mengarah ke bawah(kanan)



Gambar 4.36. Gambar dari *webcam (kiri)* dan gambar objek kura-kura yang mengarah diagonal bawah kiri (kanan)

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan sistem yang telah dikembangkan dan hasil pengujian yang telah ditentukan, dapat disimpulkan beberapa hal sebagai berikut:

- Estimasi rotasi pada pengujian rotasi *marker* pada sumbu pusat terdapat perbedaan sudut dengan sudut asli maksimal 8.62 ^{o} dan minimal -9.23 ^{o}.
- Estimasi rotasi pada pengujian rotasi *marker* pada titik tertentu, terdapat perbadaan sudut dengan sudut asli maksimal 9.36 ^{o} dan minimal -6.04 ^{o}.
- Estimasi rotasi dengan cara menghitung 2 garis terkadang memberikan hasil yang relative akurat dengan perbedaan kurang lebih +- 10 derajat. Hal ini dikarenakan *parallax* atau perbedaan posisi sudut pengamatan yang menyebabkan terjadi perbedaan posisi apa yang dilihat kamera dengan yang sebenarnya.
- Faktor external seperti cahaya yang terlalu terang atau bayangan yang menutupi *marker* dapat mempengaruhi kinerja pendeteksian.
- Sistem pendeteksian *marker* tergolong memiliki performa yang cukup bagus. *Marker* dapat dideteksi hingga pada jarak 3.3 meter(330 cm). Namun jarak yang efektif untuk pengenalan rotasi adalah 20 cm hingga 2.1 meter(210 cm) dari kamera.
- Derajat kemiringan *marker* memiliki pengaruh dalam kinerja pendeteksian *marker*. Karakteristik jarak efektif maksimal *marker* dari kamera adalah :
    - Pada derajat kemiringan *marker* 30 ^{o} adalah 210 cm.
    - Pada derajat kemiringan *marker* 60 ^{o} adalah 240 cm.
    - Pada derajat kemiringan *marker* 90 ^{o} adalah 270 cm.
    - Pada derajat kemiringan *marker* 120 ^{o} adalah 180 cm
    - Pada derajat kemiringan *marker* 150 ^{o} adalah 150 cm
- Tingkat kemiringan marker memiliki pengaruh terhadap estimasi rotasi

- System pendeteksian marker dapat berjalan pada webcam dengan spesifikasi resolusi rendah, namun tingkat akurasi estimasi rotasi semakin menurun.
- Sistem inti pendeteksian dan estimasi rotasi *marker* ini dapat dikombinasikan dengan platform yang lain. Misalnya sebagai *controller* suatu objek pada suatu game tertentu.

5.2. Saran

Terdapat beberapa saran yang diharapkan dapat mendukung pengembangan lebih lanjut, yaitu :

- Pengembangan metode estimasi rotasi yang lebih akurat agar lebih mendekati dengan rotasi yang terjadi pada dunia nyata.
- Pengenalan *marker* lebih dari 1 jenis
- Implementasi *augmented reality* pada project lain. Misalnya: *advertising*

DAFTAR PUSTAKA

Andrew, Greensted(2010). *Otsu Thresholding.* Retrieved Februari 23, 2012, from: http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

An introduction to contours (n.d). Retrieved 24 February 2012, from: http://www.aishack.in/2010/01/an-introduction-to-contours/

Bowo Dwi Ariyanto, M. hariadi. (2010). simluasi perilaku pergerakan objek 3d media augmented reality berbasis logika fuzzy, from http://digilib.its.ac.id/ITS-Master-3100012045760/18002

Camera Calibration(n.d.). Retrieved October 11, 2011, from: http://www.umiacs.umd.edu/~ramani/cmsc828d/lecture9.pdf

Documentation: Image Segmentation-Dynamic Thresholding. Retrieved Februari 22, 2012, from: http://www.dandiggins.co.uk/arlib-3.html

Documentation: Geometric Image Transformations. Retrieved Februari 22, 2012, from: http://opencv.willowgarage.com/documentation

Documentation: Camera Calibration and 3D Reconstruction. Retrieved Februari 22, 2012, from: http://opencv.willowgarage.com/documentation/

Gary, B. , Adrian, K. *Learning OpenCV*. Retrieved January 23, 2012, from: http://www.cse.iitk.ac.in/users/vision/dipakmj/papers/OReilly%20Learning%20OpenCV.pdf

Gady, Adam (2006). *Introduction to programming with OpenCV*. Retrieved October 11, 2011, from: http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html

Gopall, D.J, Jayanthi,S(2008). A simple Scheme for Contour Detection. Retrieved 24 February 2012, from: http://cvit.iiit.ac.in/papers/

Hongzhi, John. *generalizing Edge Detection To Contour Detection for Image Segmentation.* Retrieved 24 February 2012, from: *http://www.cs.stevens.edu/~oliensis/contourjournal.pdf*

Kim,M.Shultz. *Detection and Reading of Visual Code Markers.* Retrieved January 22, 2012, from: http://www.stanford.edu/class/ee368/Project_06/Project/ee368_reports/ee368group41.pdf

Konstantinos, G. (2010). *Overview of the ransac algorithm*. Retrieved October 11, 2011, from http://www.cse.yorku.ca/~kosta/CompVis_Notes/ransac.pdf

Martin, Hirzer. (2008). *Marker Detection for Augmented Reality Application.* Retrieved October 11, 2011, from: http://studierstube.icg.tu-graz.ac.at/thesis/marker_detection.pdf

Nick Lowe. *An Introduction to Real-time Rendering Using OpenGL.* Retrieved 24 February 2012s, from: http://60hz.csse.uwa.edu.au/workshop/workshop0/workshop2.html

Parker, Jamer. (*1997*), *Algorithms for Image Processing and Computer Vision (2nd ed.),* New York : John Wiley & Sons,inc

Peter, Michael, Stephen, Erik. (2005). *Fundamental of Computer Graphics(2nd ed.) , India : A.K Peters,LTD*

Richard, Benjamin, Nicholas. (2007). *OpenGL superbible (4th ed.)Michigan : Pearson Education*

Rudy Adipranata,Resmana Lim,Anton setiawan. *Rekonstruksi Obyek 3D dari Gambar 2D dengan Metode Generalized Voxel Coloring, pp. 2-3*

Serge ,B. ,Thomas,L. , Jianbo(2001). *Contour and texture Analysis for Image Segmentation.* Retrieved 24 February 2012, from : *http://www.eng.utah.edu/~bresee/compvision/files/MalikBLS.pdf*

William R, Alan B. (2003). "*Understanding Virtual Realty, Interface, Application and Design".* New York *:* Elsevier