Aplikasi Augmented Reality untuk Katalog Penjualan Rumah

Bregga Tedy Gorbala[1], Mochamad Hariadi[2]
[1]*Bidang Studi Teknik Komputer & Telematika, Jurusan Teknik Elektro ITS Surabaya*
[2]*Bidang Studi Teknik Komputer & Telematika, Jurusan Teknik Elektro ITS Surabaya*

Abstract*—Augmented Reality* (AR) adalah suatu lingkungan yang memasukkan objek virtual 3D kedalam lingkungan nyata secara *real-time*. Penelitian ini akan memasukkan teknologi AR kedalam katalog penjualan rumah sehingga katalog rumah ini menjadi lebih hidup dengan adanya animasi-animasi disekitar rumah. Katalog rumah AR ini memerlukan video *streaming* yang diambil dari kamera sebagai sumber masukan, kemudian aplikasi ini akan melacak dan mendeteksi *marker* (penanda) dengan menggunakan sistem *tracking*, setelah *marker* dideteksi, model rumah 3D digambar diatas *marker* seolah-olah model rumah tersebut nyata. Untuk menggunakan model rumah pada aplikasi katalog rumah AR ini, model harus dibuat terlebih dahulu dengan perangkat lunak desain 3D (3DS Max, Blender, Sketchup) kemudian diubah formatnya menjadi format yang didukung oleh aplikasi ini. Pada proses pengubahan format ini, terjadi berbagai macam masalah yang menyebabkan model yang ditampilkan berbeda dengan model asli atau bahkan gagal ditampilkan. Pengujian dilakukan dengan menggunakan tiga webcam yang berbeda, sepuluh *marker*, dan dua puluh model 3D yang dibuat menggunakan tiga perangkat lunak desain 3D yang berbeda. Dari hasil pengujian, aplikasi ini berjalan baik ketika menggunakan webcam yang berbeda-beda. 90% model rumah yang dibuat dengan 3DS Max berhasil ditampilkan sesuai dengan model aslinya. Sedangkan model yang dibuat menggunakan Blender dan Sketchup hanya 40% saja yang berhasil ditampilkan sesuai dengan aslinya. Hal ini dikarenakan *plugin* yang digunakan untuk mengkonversi model masih belum sempurna sehingga ada sebagian *texture* pada model yang hilang.

***Kata Kunci—Augmented Reality, ARToolKit, House Catalog*.**

I. PENDAHULUAN

DENGAN memanfaatkan teknologi AR, maket/miniatur rumah yang biasa digunakan untuk memberi contoh rumah sebenarnya dapat digantikan dengan model rumah 3D yang ditampilkan secara virtual menggunakan perangkat komputer, sehingga para pengusaha properti dapat menghemat biaya pengeluaran karena mereka tidak perlu lagi membuat miniatur rumah dan menggantinya dengan aplikasi katalog rumah AR ini. Tidak hanya pembeli dapat melihat bagian dalam rumah dengan detil, tetapi lingkungan disekitar rumah juga akan terasa lebih hidup dengan adanya animasi pendukung seperti mobil yang melintas, burung-burung terbang, dan lain sebagainya. Penelitian ini bertujuan untuk menghasilkan sebuah aplikasi yang dapat menampilkan model rumah 3D dalam lingkungan augmented reality sehingga dapat membantu para pembeli untuk mengetahui dengan baik rumah yang akan mereka beli.

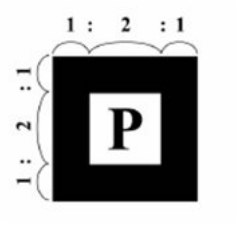
II. DASAR TEORI

Augmented reality (AR) adalah sebuah istilah untuk lingkungan yang menggabungkan dunia nyata dan dunia virtual yang dibuat oleh komputer sehingga batas antara keduanya menjadi sangat tipis. Sistem ini lebih dekat kepada lingkungan nyata (*real*). Karena itu, *reality* lebih diutamakan pada sistem ini. Sistem ini berbeda dengan *virtual reality* (VR), yang sepenuhnya merupakan *virtual environment* . Dengan bantuan teknologi AR (seperti visi komputasi dan pengenalan objek) lingkungan nyata disekitar kita akan dapat berinteraksi dalam bentuk digital (*virtual*). Informasi-informasi tentang objek dan lingkungan disekitar kita dapat ditambahkan kedalam sistem AR yang kemudian informasi tersebut ditampilkan diatas *layer* dunia nyata secara *real-time* seolah-olah informasi tersebut adalah nyata. *Augmented reality* memiliki banyak potensi didalam industri dan penelitian akademis.

ARToolKit adalah *tracking system library* yang bersifat *open-source* yang memungkinkan *programer* dengan mudah mengembangkan aplikasi *Augmented Reality*[1]. Salah satu bagian paling sulit mengembangkan aplikasi AR justru menghitung sudut pandang pengguna secara *real time* sehingga model virtual selaras dengan lingkungan dan objek dunia nyata. ARToolKit menggunakan teknik visi komputer untuk menghitung posisi kamera nyata dan hubungannya terhadap *marker*, sehingga memungkinkan para programmer untuk menampilkan objek virtual ke *marker* ini. Cepat dan tepat, adalah ciri dari sistem pelacakan (*tracking*) yang disediakan oleh ARToolKit sehingga akan menghasilkan banyak aplikasi AR baru yang menarik. Didalam ARToolKit sudah terdapat sistem pelacak dan *source code* lengkap untuk sehingga memudahkan *programer* untuk melakukan pemrograman pada berbagai *platform* atau menyesuaikannya untuk aplikasi mereka sendiri.

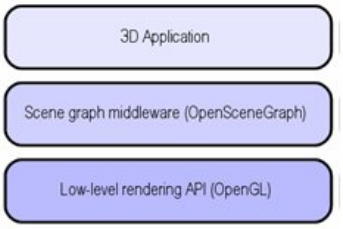
Fiducial marker adalah sebuah penanda yang didalamnya terdiri dari kumpulan titik acuan untuk memudahkan komputasi dari pengukuran parameter-parameter yang dibutuhkan dalam pengolahan citra. *Marker* dapat berupa warna atau dapat berupa gambar. Sudah banyak penelitian tentang penanda untuk keperluan AR. Penanda yang paling sederhana dan bekerja dengan sangat baik adalah penanda *matrix*[2] (lihat gambar 1). Penanda *matrix* menggunakan 2D *barcode* sederhana, yang dipakai untuk mengenali sebuah objek dan untuk mengetahui hubungan antara posisi kamera dengan penanda tersebut.

OpenSceneGraph (OSG) adalah *application programmer interface* (API) yang bersifat *open source* untuk menangani *high performance grafis* 3D yang biasanya digunakan oleh para pengembang aplikasi dalam bidang-bidang tertentu



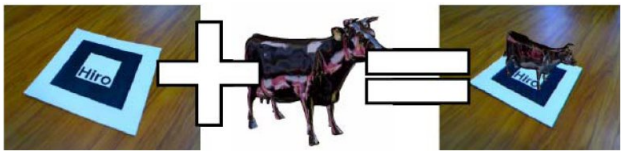
Gambar 1. Contoh *Fiduciary Marker* 2D yang digunakan ARToolKit untuk sistem *tracking*.

seperti *visual simulation*, *computer games*, *virtual reality*, *scientific visualization* dan *modeling*. OSG berperan penting dalam aplikasi 3D karena OSG merupakan perangkat lunak *middleware* yang posisinya berada diatas OpenGL (lihat gambar 2), membuat OSG menyediakan level *rendering* ke arah yang lebih tinggi, I/O, dan mengatur fungsi lainnya kedalam aplikasi 3D[3] seperti diperlihatkan pada gambar 2. Banyak aplikasi 3D membutuhkan fungsi tambahan dari *middleware library* daripada berinterkasi langsung dengan *low-level rendering* API.



Gambar 2. Penempatan OpenSceneGraph pada aplikasi 3D.

OSGART adalah sebuah *library* yang ditulis dalam bahasa pemrograman C++ yang ditujukan untuk mengembangkan aplikasi *augmented reality* atau *mixed reality* dengan menggabungkan *computer vision based tracking libraries* (seperti ARToolKit, ARToolKitPlus, SSTT dan BazAR) dengan 3D *scene graph libary* (OpenSceneGraph) seperti ditunjukkan pada gambar 3.



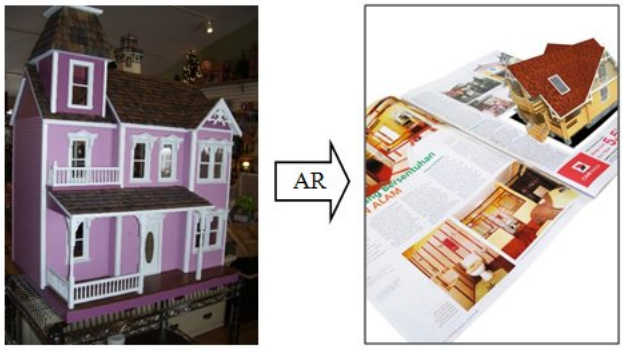
ARToolKit OpenSceneGraph osgART

Gambar 3. OsgART menambahkan fungsi AR pada OpenSceneGraph

Sudah banyak tersedia *toolkit* untuk membuat dan mengembangkan aplikasi AR, mulai dari *low-level programming* (e.g. ARToolKit) sampai *high-level programming*. Keberhasilan ARToolKit untuk membuat aplikasi AR disebabkan karena kesederhanaan tingkat pemrogramannya. Oleh karena itu OSGART dibuat dengan tujuan untuk mempertahankan kesederhanaan yang ada pada ARToolKit dan membawa ARToolKit ke generasi yang lebih tinggi. OSGART beralih dari GLUT (*The OpenGL Utility Toolkit*) ke OSG untuk urusan *rendering* model, sehingga kualitas grafis akan menjadi lebih baik ketika dipakai pada sebuah aplikasi[4]. Selain itu, versi dasar dari OSGART ini bersifat bebas dan dapat digunakan untuk keperluan akademik. Siswa dapat dengan bebas melakukan penelitian tentang AR menggunakan *toolkit* ini. *Toolkit* ini tetap mengacu pada prinsip kesederhanaan untuk mengembangkan aplikasi yang akan kita buat (membuat video, menggunakan *tracker*, dan lain sebagainya). Pada dasarnya, *programer* mengembangkan aplikasi mereka dalam bahasa C++, tetapi *programer* juga dapat mengembangkan aplikasinya dalam bahasa pemrograman lain seperti Python, Lua, atau Ruby (dengan menggunakan osgBindings atau osgIntrospection module). OSGART mempunyai berbagai macam fitur yang telah didesain sedemikian rupa untuk dapat lebih memaksimalkan pengembangan aplikasi AR

III. Disain dan Implementasi Sistem

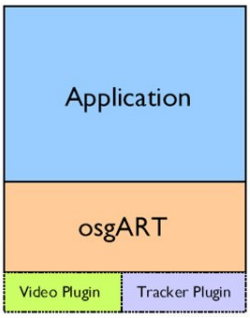
Penelitian ini dibuat berdasarkan kebutuhan seperti yang digambarkan pada gambar 4. Bagian sebelah kiri dari gambar 4 menunjukkan cara tradisional yang dipakai oleh para penjual rumah, yaitu dengan meperlihatkan maket/miniatur rumah yang akan dijual. Maket rumah dibuat semirip mungkin dengan rumah aslinya dengan perbandingan tertentu.



Gambar 4. Aplikasi AR pada katalog rumah

Untuk mendapatkan hasil seperti pada 4, maka teknologi *augmented reality* harus ditambahkan pada sebuah katalog rumah sederhana. Fungsi kamera dalam penelitian ini adalah sebagai media visi bagi aplikasi AR untuk mendapatkan video masukan. Kamera mengambil *frame-frame* video untuk dapat diterima oleh komputer. Komputer digunakan untuk memproses citra digital yang diakuisisi oleh kamera, *frame* demi *frame*. Sebuah *tracking system library* untuk aplikasi AR seperti ARToolKit diperlukan untuk dapat mendeteksi *marker* yang ada pada *frame-frame* video tersebut. Tetapi ARToolKit memiliki kelemahan dalam hal *rendering* model. Sehingga sebuah *library* yang dapat me-*render* model rumah 3D dengan kualitas tinggi seperti OSG diperlukan. Untuk dapat menutupi kelemahan yang dimiliki ARToolKit, proses *rendering* model rumah harus ditangani oleh OSG agar hasil yang didapat maksimal. OsgART adalah solusi untuk masalah ini, karena osgART dapat membuat OSG memiliki fungsi AR (gambar 3).

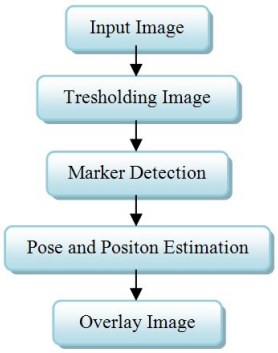
A. OsgART



Gambar 5. Arsitektur osgART

Dari gambar 5, diketahui bahwa osgART memerlukan dua macam *plugin* untuk dapat bekerja, yaitu *video plugin* dan *tracker plugin*. *Video plugin* digunakan sebagai sumber masukan berupa video, sedangkan *tracker plugin* adalah *library tracking sistem* yang digunakan untuk melacak keberadaan *marker* (penanda) didalam aplikasi AR ini.

B. ARToolKit

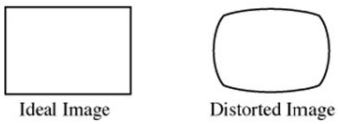


Gambar 6. *Pipeline* ARToolKit

Seperti ditunjukkan pada gambar 6, langkah awal yang harus dilakukan adalah mendapatkan masukan video dari sebuah kamera. Video yang di-*streaming* secara *real-time* ini akan diolah oleh sistem untuk dianalisa *frame* per *frame*.

Sebelum kamera digunakan, kamera harus dikaliberasi terlebih dahulu. Kaliberasi kamera merupakan bagian yang sangat penting dalam proses pengambilan masukan video. Hal ini disebabkan oleh distorsi pada lensa kamera yang tiap-tiap kamera berbeda karakteristiknya (gambar 7). Tujuan dari kalibrasi kamera adalah untuk menghitung tingkat distorsi dari sebuah lensa kamera yang digunakan agar *image* yang dihasilkan mendekati *image* ideal. Parameter ini nantinya digunakan dalam perhitungan pada proses *Pose and Position Estimation* agar model rumah dapat ditampilkan tepat diatas *marker*.

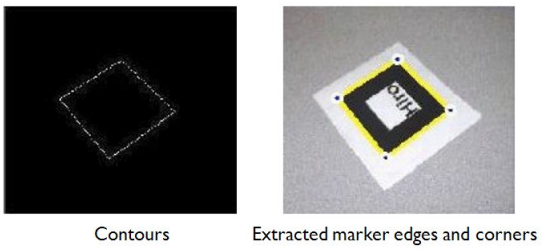
Video yang diterima selanjutnya akan mengalami proses binarisasi (*gray-scale*), kemudian nilai *threshold* ditentukan sehingga mengasilkan gambar hitam-putih. Nilai *threshold* berada pada angka 0 – 255 dan secara *default*, *threshold*



Gambar 7. Perbandingan antara *image* yang ideal dengan *image* yang disebabkan oleh faktor distorsi

bernilai 100. Fungsi dari proses ini adalah untuk membantu sistem agar dapat mengenali bentuk segi empat dan pola di *marker* pada video yang diterima. Nilai *threshold* dapat dirubah dan disesuaikan dengan kondisi cahaya disekitar *marker* untuk tetap membuat *marker* terlihat sebagai segi empat, karena ketika cahaya disekitar *marker* berkurang ataupun berlebih pada saat proses *thresholding*, sistem tidak dapat mendeteksi *marker*. Hal ini penting mengingat aplikasi ini bekerja dengan cara mengenali *marker*.

Setelah video mengalami proses *thresholding*, langkah selanjutnya adalah mendeteksi *marker*, dimana sistem akan mengenali bentuk dan pola yang ada pada *marker*. Sistem akan mencari bagian yang memiliki bentuk segi empat dan menandainya. Sistem juga akan menghilangkan area yang tidak berbentuk segi empat sehingga yang akan ditampilkan pada layar hanyalah area yang memiliki bentuk segi empat.



Gambar 8. Hasil dari *contour extraction* dan *corner detection*

Contour extraction dan *corner detection* digunakan untuk mendapatkan koordinat dari empat sisi dan empat titik sudut pada segi empat yang tersisa setelah proses *image labeling* (gambar 8). Setelah proses ini selesai dilakukan, dua garis paralel pada *marker* diproyeksikan sehingga persamaan garisnya pada koordinat layar kamera adalah seperti berikut ini :

$$a_1x + b_1y + c_1 = 0 \qquad a_2x + b_2y + c_2 = 0 \tag{1}$$

Parameter pada persamaan 1 akan disimpan dan dipakai pada proses selanjutnya.

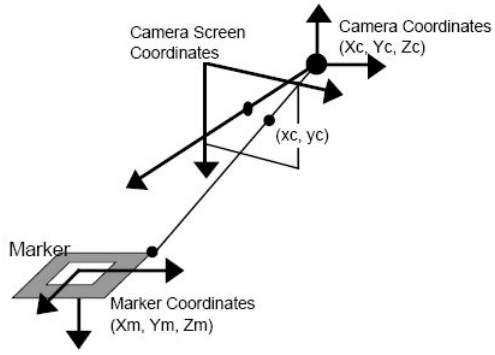
Gambar 9. *Pattern normalization* dan *template matching*

Karena sudut dari lensa kamera tidak tegaklurus terhadap *marker* ketika mengambil video, sudut-sudut *marker* yang dibentuk oleh sisi-sisi segi empat tidak 90° (9). Hal ini

membuat pola yang ada didalam *marker* tidak dapat dikenali dengan baik. *Pattern normalization* berperan untuk mengubah sudut *marker* yang tidak 90° menjadi 90° agar pola dapat dikenali dan dicocokkan menggunakan *template matching* dengan pola (*template*) yang telah ada pada sistem untuk memperoleh positif ID dari *marker* tersebut. Sebuah gambar, foto, maupun nama dapat dijadikan pola pada sebuah *marker* agar sistem dapat mengenali pola itu.

Untuk menaruh objek 3D tepat diatas *marker*, sistem perlu mengetahui koordinat dari *marker* dan kamera.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} V_{11} & V_{12} & V_{13} & W_z \\ V_{21} & V_{22} & V_{23} & W_y \\ V_{31} & V_{32} & V_{33} & W_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix}$$
$$= \begin{bmatrix} & \mathbf{V}_{3\times3} & & \mathbf{W}_{3\times1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = \mathbf{T_{cm}} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \quad (2)$$



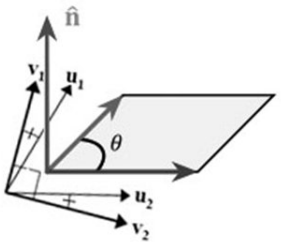
Gambar 10. Hubungan antara koordinat *marker* dengan koordinat kamera.

Matrix transformasi ($\mathbf{T_{cm}}$) dari koordinat *marker* ke koordinat kamera seperti pada gambar 10 diberikan pada persamaan 2[5]. Untuk *marker* yang sudah dikenali, nilai dari parameter a_1,b_1 , c_1 dan a_2,b_2 , c_2 didapatkan ketika proses *contour extration*. Matrix proyeksi **P** pada persamaan 3 diperoleh ketika proses kalibrasi kamera. Dengan mengganti x_c dan x_y pada persamaan 3 untuk x dan y pada persamaan 1 didapat persamaan garis seperti persamaan 4.

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} & 0 \\ 0 & P_{22} & P_{23} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \begin{bmatrix} hx_c \\ hy_c \\ h \\ 1 \end{bmatrix} = P \begin{bmatrix} Z_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \quad (3)$$

$$a_1P_{11}X_c + (a_1P_{12} + b_1P_{22})Y_c + (a_1P_{13} + b_1P_{23} + c_1)Z_c = 0$$
$$a_2P_{11}X_c + (a_2P_{12} + b_2P_{22})Y_c + (a_2P_{13} + b_2P_{23} + c_2)Z_c = 0 \quad (4)$$

Marker segi empat yang digunakan mempunyai empat sisi dimana dua sisi adalah garis yang paralel. Vektor normal dari *marker* adalah $\tilde{\mathbf{n}}$ yang dihasilkan dari perkalian *cross* vektor $\mathbf{u}_1$ dan $\mathbf{u}_2$, seperti ditunjukkan pada gambar 11. Pada kenyataanya, vektor $\mathbf{u}_1$ dan $\mathbf{u}_2$ seharusnya tegak lurus, hal ini disebabkan oleh sudut kamera ketika pengambilan gambar yang tidak tegak lurus terhadap marker. Vektor $\mathbf{v}_1$ dan $\mathbf{v}_2$



Gambar 11. Dua buah vektor yang tegak lurus : $\mathbf{v}_1$ dan $\mathbf{v}_2$ didapat dari $\mathbf{u}_1$ dan $\mathbf{u}_2$.

dibuat agar memiliki sudut 90° dengan menggunakan nilai dari vektor$\mathbf{u}_1$ dan $\mathbf{u}_2$untuk memperkecil kesalahan. Setelah $\mathbf{v}_1$ dan $\mathbf{v}_2$ tegak lurus, $\mathbf{v}_3$ dihasilkan dari perkalian *cross* $\mathbf{v}_1 \times \mathbf{v}_2$. Nilai $\mathbf{v}_1$, $\mathbf{v}_2$, dan $\mathbf{v}_3$ adalah komponen rotasi pada matrix transformasi $\mathbf{T_{cm}}$ dari koordinat *marker* ke koordinat kamera seperti yang disampaikan pada persamaan 2.

Setelah komponen rotasi $\mathbf{V}_{3\times3}$ pada matrix transformasi diketahui, komponen translasi $\mathbf{W}_1$, $\mathbf{W}_2$, dan $\mathbf{W}_3$ dapat diperloleh dengan menggunakan persamaan 2 dan 3.

Setelah transformasi matrix didapat, langkah terakhir yang dilakukan adalah menggambar objek virtual 3D pada frame video tepat diatas permukaan *marker* dan hasilnya dapat dilihat pada keluaran videonya. Dengan demikian model rumah virtual seolah-olah ada diatas *marker.*

C. Implementasi Sistem

Implementasi dari katalog rumah AR ini dibagi menjadi beberapa tahap, yaitu:

- Inisialisasi
  osgViewer adalah kelas untuk membuat *window* yang digunakan untuk memuat hasil *render* dari model 3D. Langkah pertama yang dilakukan adalah membuat *window viewer* karena hampir semua bagian dari aplikasi ini membutuhkan window ini untuk proses visualisasi.
- Menambahkan Video
  Setelah *window viewer* terbentuk, langkah selanjutnya adalah menambahkan *video plugin* untuk mendapatkan sumber masukan berupa video. Aplikasi katalog AR ini tidak dapat berjalan ketika tidak ada *video plugin* dan akan menampilkan pesan *error.* Video akan mulai direkam dan terus di-*streaming* sebagai video masukan dan dijadikan sebagai *video background* dengan menggunakan algoritma.
- Sistem Tracking
  Untuk dapat membaca posisi *marker* maka sistem harus dilengkapi dengan sistem *trakcing*. ARToolKit telah menyediakan kelas tersendiri yang disebut *tracker* untuk mengatasi masalah *trakcing* sistem ini. Setelah tracker dari ARToolKit di-*load*, *tracker* akan dihubungkan dengan *video plugin* untuk dapat menerima *input* dan mendeteksi *marker*. Telah dijelaskan diawal bahwa proses *thresholding* sangat diperlukan ketika keadaan cahaya disekitar marker berkurang atau berlebih. Untuk itu kita perlu merubah nilai *threshold* tergantung pada keadaan cahaya disekitar marker. Kelas *tracker* juga perlu

disesuaikan dengan video yang diterima dari kamera, hal ini dikarenakan adanya kesalahan yang disebabkan oleh distorsi dari lensa kamera. Oleh karena itu *file* kalibrasi kamera juga digunakan dalam proses *tracking* ini.

- Menghitung transformasi kamera terhadap *marker*
  Matriks proyeksi **P** digunakan oleh ARToolKit perlu diterapkan dalam OSG. Seperti telah dijelaskan sebelumnya bahwa matriks proyeksi didapatkan ketika proses kalibrasi. Transformasi pada marker yang telah dihitung oleh ARToolKit, harus dipetakan juga kedalam transformasi OSG. ARToolKit menggunakan matriks transformasi yang berbeda untuk setiap penanda yang dikenali pada tiap *frame* kamera. OSG menggunakan transformasi untuk memposisikan model 3D. Kemudian OSGART memetakan transformasi ARToolKit ke dalam transformasi OSG.
- Menambahkan Model 3D
  langkah selanjutanya adalah me-*load* model OSG dan menggambarnya diatas *marker* sesuai dengan posisi dan pose koordinat masing-masing model.
- Menutup *window*

Langkah 2 sampai 5 akan diulang terus menerus (*looping*) sampai aplikasi berhenti, sedangkan langkah 1 dan 6 dilakukan pada masing-masing inisialisasi dan *shutdown*. Selain langkah-langkah tersebut, aplikasi ini mungkin memerlukan respon dari mouse, keyboard atau aplikasi lain.

Metode LOD (Level-of-Detail) juga digunakan dalam penelitian ini. Dalam metodel ini, dua model di-*load* secara bersamaan, yaitu model office.ive dan office2.ive. LOD menggunakan dua buah *node*, yaitu *closer node* (jarak dekat) dan *far node* (jarak jauh). Pada saat jarak antara kamera dan *marker* antara 0 cm sampai 25 cm (dekat), model yang digunakan adalah office.ive sedangkan ketika jarak melebihi 25 cm (jauh) maka sistem akan mengganti model office.ive dengan office2.ive.

IV. Pengujian Sistem

Dalam pengujian ini, *marker* yang digunakan berjumlah 10 buah. Model 3D yang digunakan didesain dengan menggunakan perangkat lunak yang berbeda-beda, yakni 3DS Max 2010, Google Sketchup v7, dan Blender v2.6.2. *File format* yang dipakai adalah *file format* OSG (*.osg, *.ive). Oleh karena itu dibutuhkan perangkat lunak tambahan untuk membantu merubah format asli menjadi format yang didukung oleh OSG seperti OSGExp v0.9 (untuk 3DS Max) dan osgexporter v2.42b (untuk Blender). Plugin tersebut akan mengubah *file format* menjadi *.osg. Format *.osg tidak dapat mengandung *texture*, sehinga untuk model-model yang memiliki *texture* harus diubah menjadi *.ive yang juga merupakan *file format* pada OSG dan memiliki ukuran file yang relatif kecil. Untuk itu OSG telah menyediakan *utility* yang disebut *Osg Converter* (osgconv). *Osg Converter* adalah *utility* untuk membaca database 3D dan dapat mengubah format *.osg menjadi *.ive melalui perintah di *command prompt*. Berikut ini adalah contoh perintah yang ditulis pada *command prompt*

```
osgconv cow.osg cow.ive
```

Perintah tersebut akan mengkonversi *file* cow.osg menjadi cow.ive. Ukuran file *.ive dapat diperkecil tanpa terlalu mempengaruhi kualitas model. Hal ini dapat dilakukan dengan menambahkan pilihan *compressed* pada perintah osgconv seperti dibawah ini.

```
osgconv --compressed cow.osg cow.ive
```

Terdapat langkah yang berbeda untuk mengubah *file format* sketchup (*.skp) agar model tersebut dapat di-*render* pada aplikasi ini. Google Sketchup memiliki *file exporter* kedalam format COLADA (*.dae) yang juga didukung oleh Blender dan Google Earth. Setelah *file* *.skp dieksport ke *.dae, Blender akan mengubah file *.dae menjadi *.osg kemudian dengan menggunakan *utility* osgconv, format *.osg akan dikonversi menjadi *.ive.

Dari tujuh model 3D yang didesain menggunakan Sketchup, hanya satu model yang dapat ditampilkan sempurna seperti desainnya. Juga terdapat dua model yang menyebabkan aplikasi katalog ini menjadi *crash* ketika me-*load* model. Hal ini disebabkan oleh besarnya ukuran *file* (Audi.osg : 23,8 Mb dan jazz.osg : 6,13 Mb) dan juga terdapat kesalahan pada *plugin* osgexporter untuk Blender yang kurang sempurna dalam mengeksport *file* OSG (*.osg). Pada 7 model Sketchup tersebut tidak ada model yang dilengkapi dengan fitur animasi. Bentuk model terlihat lebih kasar dan ada *texture* yang hilang dikarenakan proses konversi yang kurang sempurna (pada *file* dengan hasil kurang dari 100%).

Kemuian dari lima model yang didesain menggunakan Blender, yang dua diantaranya adalah model dengan animasi. Proses konversi dari *format file* Blender (*.blend) ke OSG (*.osg, *.ive) berjalan sempurna dan animasi juga berjalan dengan baik ketika model di-*render* pada aplikasi katalog rumah AR ini. Tetapi ada juga model seperti Porsche_550.blend ketika dikonversi ke format OSG, model tersebut kehilangan sebagian bentuk dan *texture*-nya. Hal ini sebabkan oleh *plugin* yang digunakan untuk proses konversi masih belum sempurna.

Pengujian format model yang terakhir adalah model yang dibuat dengan menggunakan 3DS Max 2010. Ada sepuluh model yang digunakan untuk proses pengujian ini, dua diantaranya adalah model dengan animasi. Dari sepuluh model yang diuji, hanya satu model yang tidak sempurna. Hal ini disebabkan karena model ini dibuat dengan menggunakan 3DS Max versi lama, sehingga ada beberapa bagian yang tidak cocok dengan versi 3DS Max yang digunakan pada proses pegujian ini. Model yang lainnya berjalan dengan sempurna ketika dijalankan pada aplikasi AR ini sesuai dengan desain aslinya, termasuk juga model dengan animasi.

A. Keterbatasan

Visi komputer berbasis *tracking* sistem sangat memungkinkan untuk membuat begitu banyak aplikasi, tetapi ada keterbatasan yang mempengaruhi ARToolKit dan sistem visi yang lainnya.

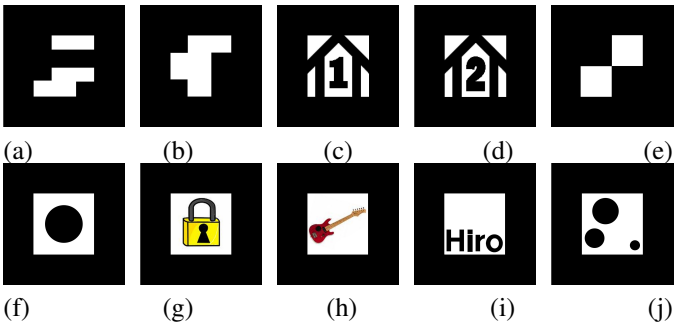
- Oklusi
  Tentu saja benda-benda virtual hanya akan muncul ketika *marker* ditangkap kamera. Hal ini membatasi ukuran

atau gerakan dari objek virtual. Ini juga berarti bahwa jika pengguna menutupi pola yang ada pada *marker* dengan tangan mereka atau benda lain, objek virtual akan menghilang. Juga ketika garis pinggir *marker* berada di luar *frame* kamera, dan memiliki lebih dari empat sudut, maka ARToolKit akan gagal untuk mengenali *marker*.

Marker		Jarak Kamera - Marker		Kemiringan
Ukuran	Kompleksitas	Terpendek	Terjauh	Minimum
8	sederhana	14	98	78
8	sederhana	14	93	80
8	kompleks	13,5	121	75
8	kompleks	14	113	75
12	sederhana	20	168	76
12	kompleks	20	198	80
16	sederhana	27	268	80

Tabel I
BATASAN PADA ARTOOLKIT.

- Jarak
  Jarak juga menjadi masalah dalam pelacakan optik, ketika *marker* bergerak menjauhi kamera, mereka menempati lebih sedikit piksel pada layar kamera, dan mungkin tidak cukup detail untuk dapat dengan benar mengidentifikasi pola pada *marker*. Tabel I memperlihatkan beberapa jenis *marker* persegi dengan ukuran yang berbeda (dalam satuan sentimeter). Hasil ini didapatkan dengan menempatkan *marker* dilantai dan menggerakkan kamera dengan kemiringan (dalam satuan derajat) dan jarak (dalam satuan sentimeter) yang berbeda sampai benda virtual pada marker menghilang. Dari tabel I dapat disimpulkan bahwa semakin besar ukuran pola *marker*, semakin jauh pula pola dapat dideteksi.



Gambar 12. *Marker* yang diujikan.

- Marker
  Dari gambar 12, *marker* (c) dan (d) sering sekali menampilkan model yang bukan modelnya. Hal ini disebabkan oleh pola dari kedua *marker* tersebut sangat mirip sehingga menimbulkan kesalahan ketika proses *pattern normalization* dan *template matching*.
  Kesalan juga terjadi pada *marker* (e) dan (f), ketika model digambar diatas *marker*, model sering berganti posisi dengan sendirinya. Kalau dilihat dengan seksama, *marker* (e) dan (f) adalah *marker* yang simetris. Hal ini menyebabkan sistem kurang dapat menentukan arah dan posisi model pada saat melakukan proses *pose and position estimation*. Kesalahan ini dapat dihindari dengan mengganti pola *marker* dengan bentuk yang asimetris.

Merek	Resolusi	Fps	Warna	Jarak Terpendek	Hasil
A4Tech	320x240	30	RGB-24	15 cm	OK
A4Tech	640x480	30	RGB-24	20 cm	tersendat-sendat
A4Tech	800x600	30	RGB-24	26 cm	tersendat-sendat
Ysomc	320x240	30	YUY2	15 cm	OK
Ysomc	640x480	30	YUY2	23 cm	OK
Ysomc	800x600	30	YUY2	-	tidak didukung
Logitech	320x240	30	RGB-24	15 cm	OK
Logitech	640x480	30	RGB-24	19 cm	OK
Logitech	800x600	30	RGB-24	24 cm	tersendat-sendat

Tabel II
SPESIFIKASI KAMERA UNTUK PENGUJIAN DAN HASIL PENGUJIAN

- Kamera
  Proses pengujian aplikasi katalog rumah AR ini juga menggunakan tiga jenis webcam. Spesifikasi kamera dan hasil dari pengujian dapat dilihat pada tabel II. Dari tabel terlihat bahwa faktor kamera (distorsi pada lensa, resolusi video, *frame rate*) juga menentukan keluaran yang dihasilkan oleh aplikasi ini. Hasil keluaran yang baik pada ketiga kamera yaitu ketika konfigurasi yang digunakan adalah resolusi video 320x240 dengan *frame rate* 30fps. Jarak terpendek antar *marker* dengan kamera juga didapat dengan menggunakan resolusi dan *frame rate* tersebut. Semakin tinggi resolusi video yang digunakan, jarak terpendek antar *marker* dengan kamera semakin bertambah jauh. Kesalahan perhitungan jarak dari *marker* ke kamera disebabkan oleh faktor distorsi lensa. Tetapi kesalahan ini sudah diperkecil dengan adanya kalibrasi kamera oleh sistem untuk memasukkan *file* kalibrasi kedalam proses perhitungan transformasi pada model.
- Cahaya
  Faktor cahaya yang dapat ditangkap oleh lensa kamera juga berperan penting karena aplikasi ini menggukan metode *threshold* untuk sistem *trakcing*-nya. Tetapi walaupun demikian, nilai *threshold* dapat diatur sehingga untuk ruangan yang kekurangan cahaya, sistem tetap dapat mendeteksi *marker* dengan baik.

V. PENUTUP

A. Kesimpulan

Setelah melakukan tahapan implementasi dan pengujian sistem, ada beberapa hal yang perlu diperhatikan agar mendapatkan hasil yang maksimal dalam menggunakan aplikasi katalog rumah AR ini. Untuk menampilkan model yang detil dan disertai animasi dengan sangat baik, 3DS Max adalah pilihan yang tepat. Data pengujian menunjukkan hanya satu dari sepuluh model yang tidak dapat ditampilkan dengan sempurna. Hal ini berarti 90% model yang didesain dengan 3DS Max bekerja sangat baik pada aplikasi AR ini dibandingkan dengan Google Sketchup dan Blender. Kamera yang digunakan sebaiknya memiliki nilai distorsi yang rendah agar dapat memperkecil nilai kesalahan perhitungan. Walaupun dengan menggunakan kamera yang berbeda,

aplikasi AR ini akan berjalan baik apabila menggunakan resolusi keluaran 320x240 dengan *frame rate* 30 fps. Marker dengan pola yang sederhana memiliki jarak Dan untuk menghindari kesalahan yang ditimbulkan oleh *marker*, sebaiknya *marker* yang digunakan adalah *marker* dengan pola asimetris karena dapat membantu proses *pose and position estimation*. Sistem *tracking* pada ARToolKit masih belum *robust* karena masih mengandalkan proses *thresholding*, *contour extraction*, dan *corner detection* sehingga marker tidak bisa dikenali ketika berada dilingkungan yang kurang cahaya, atau ketika ada benda yang menutupi bagian kecil dari *marker*.

B. Saran

Untuk pengembangan selanjutnya, diharapkan dapat menggunakan *tracking library* yang berbeda seperti ARToolKitPlus atau ARTag yang memiliki *tracking system* yang lebih baik. Perangkat keras yang digunakan tidak lagi webcam tetapi HMD (Head-Mounted-Display) agar didalam penggunaannya, para pengunjung lebih nyaman dalam berinteraksi dengan model rumah ini. Fitur pendukung seperti AR magiclenses juga dapat ditambahkan pada aplikasi ini.

DAFTAR PUSTAKA

[1] Kato, H., Billinghurst, M., dan Poupyrev, I., 2000, "*ARToolKit version 2.33: A software library for Augmented Reality Applications*", Human Interface Technology Laboratory, University of Washington

[2] Rekimoto J., "Matrix : A Real-Time Object Identification and Registration Method for Augmented Reality", Proceedings of the third Asia Pacific on computer–human interactions, Kangawa Japan, p. 63–98, 1998

[3] Paul M., 2007, "*OpenSceneGraph : Quick Start Guide*", Louisville U.S.A, Skew Matrix Software

[4] Raphaël G., dan Julian L., 2008, "OSGART Introduction, International Symposium on Ubiquitous VR" 10 July. Gwangju, South Korea

[5] Kato, H., dan Billinghurst, M., "Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System", Proceedings of 2nd Int. Workshop on Augmented Reality, 85-94, 1999