Markerless Augmented Reality Android App For Interior Decoration

Prasad Renukdas#1, Rohit Ghundiyal#2, Harshavardhan Gadgil#3, Vishal Pathare#4

#Department of Computer Engineering, MES College of Engineering, Pune, India

***Abstract*—We propose a markerlessapplication for interior decoration purposes, in which any novice user can easily decorate his/her home. We include our observations and reminisce on potential future improvements.**

This paper also discusses mobile application of Augmented Reality (AR) on the Android platform. We discuss existing SDKs for developing AR applications on mobile platforms and elaborate their functionalities and limitations.

***Keywords*— Augmented Reality, AR, interior decoration, Metaio, Artoolkit.**

I. Overview

Augmented reality (AR) is a live, direct or indirect, view of a physical, real-world environment whose elements are *augmented* by computer-generated sensory input such as sound, video, graphics or GPS data.

This technology is still in its infant stage, particularly in mobile computing, but is quickly maturing with numerous innovative applications already developed for archaeology, art, advertising, education, entertainment, medicine, industrial design, architecture, tourism and more.

Earlier, backpack portable computers and head mounted displays (HMDs) were used for augmented reality. The advancements in mobile hardware and software platforms brought AR into the hands of the common man.

The hardware used for this paper was an Alcatel OT995 Ultra smartphone. It has 512 MB of RAM, a 5 mega-pixel auto focus camera and a Qualcomm Snapdragon 1.4 GHz processor based on the ARM A9 (Cortex) architecture. The Eclipse IDE (ver Juno) was used to develop the applicationand 3D modeling was done in Blender.

II. Introduction to Augmented Reality

A. *Types of Displays Technologies.*

1) *Eye Glasses:*
AR displays can be rendered on devices resembling eyeglasses. Versions include eye wear that employ cameras to intercept the real world view and re-display its augmented view through the eye piecesand devices in which the AR imagery is projected through or reflected off the surfaces of the eye wear lens pieces. A notable example of this is the Google Glass Project, currently under development and expected to release commercially around 2014.

2) *Virtual Retinal Display:*
A virtual retinal display (VRD) is a personal display device under development at the University of Washington's Human Interface Technology Laboratory. With this technology, a display is scanned directly onto the retina of a viewer's eye. The viewer sees what appears to be a conventional display floating in space in front of them.

3) *Handheld:*
Handheld displays employ a small display that fits in a user's hand. All handheld AR solutions to date opt for video see-through. Initially handheld AR employed fiduciary markers, and current and future approaches include GPS and markerless tracking through scene recognition. Handheld display AR promises to be the first commercial success for AR technologies. The two main advantages of handheld AR is the portable nature of handheld devices and ubiquitous nature of camera phones.

B. SDKs currently available for Android development:

1) *ARToolkit:*
It is an open source marker based AR library developed at the University of Washington.
Some of the features of ARToolkit include:

- Single camera position/orientation tracking.
- Tracking code that uses simple black squares.
- The ability to use any square marker patterns.
- Easy camera calibration code.
- Fast enough for real time AR applications.
- SGI IRIX, Linux, MacOS and Windows OS distributions.
- Distributed with complete source code.

It is designed for personal computers and not for embedded devices. Hence porting it directly to mobile platforms is difficult and impractical because it uses a lot of FPU calculations.

2) *ARToolkitPlus:*

This was an enhanced version in the ARToolkit. The library was ported from C to C++. It is available for download under general public license.

It does not read camera images or render geometry of any kind nor is it an out-of-the box solution.

Some of the features of ARToolKitPlus include:

- Class-based API (compile-time parametrization using templates)
- Up to 4096 id-based markers (no speed penalty due to large number of markers)
- New camera pixel-formats (RGB565, Gray)
- Many speed-ups for low-end devices (fixed-point arithmetic, look-up tables, etc.)
- New Pose estimation algorithm gives more stable tracking (less jitter)
- Loads MATLAB camera calibration toolbox files
- Automatic thresholding
- Tools: pattern-file to TGA converter, pattern mirroring, off-line camera calibration, id-marker generator.

3) *Vuforia:*

Vuforia is an Augmented Reality SDK provided by Qualcomm, AG. It supports a variety of 2D and 3D target types including Image Targets, 3D Multi-Target configurations, and a form of addressable Fiduciary Marker known as a Frame Marker. Additional features of the SDK include localized Occlusion Detection using 'Virtual Buttons', runtime image target selection, and the ability to create and reconfigure target sets programmatically at runtime.

The latest release of the SDK, Vuforia 2.0, also supports cloud recognition, where a user points his smartphone at a book, packaging or publication and information about the item like the price, reviews, etc are shown.

4) *Metaio Mobile SDK:*

The Metaio SDK includes a powerful 3-D rendering engine in addition to plug-ins for Unity. It has advanced tracking features, such as Markerless 2D and 3D Tracking, client-based visual search and SLAM. Some of its features are:

- Native App development for iOS, Android and PC (Windows)
- Includes powerful 3D Rendering engine.
- Highly advanced tracking and minimal artefacting.
- FREE commercial usage (with watermark)
- Huge developer support community.

The latest release, Metaio SDK 4.1 includes AREL (Augmented Reality Experience Language) which is a javascript binding of the metaio SDK's API in combination with a static XML content definition. AREL allows scripting of powerful, highly interactive Augmented Reality experiences based on common web technologies such as XML, HTML5 and JavaScript. AREL allows creation of platform independent applications instead of using platform specific programming languages - Java for the Android SDK, Objective C for iOS and C++ for Windows 7 and up.

III. RESTRICTIONS ON MOBILE DEVICES

Smartphones with RAM as low as 64 MB won't support this app. Android system will force close the app if it won't respond within stipulated time (3 to 5 sec). The CPUs on the embedded processors mostly don't allow parallel processing. Even then AR can make use of multithreading which speeds up certain marking and detection algorithms. Many of the phones don't have a separate FPU. Hence the app which runs on the phone will work 3 to 4 times slower than that of the average PC and will consume the battery more than other apps due to more computations required. Another hurdle is the OpenGL. This app requires OpenGL ES version 2.0 to run and will not run on lower versions.

IV. CHOICE OF SDK

Each SDK has its own merits and limitations. We chose Metaio mobile SDK 4.1 to develop our application for the following reasons:

1) *High level of abstraction:*
Metaio implements complex computer vision algorithms in native C++ libraries and ports them to Android. This abstraction enables a developer to assume low level implementations and design and develop apps with greater ease, efficiency and enhances productivity.

2) *Advanced Tracking:*
Metaio's tracking algorithms are highly advanced, allowing us to develop apps free of visual artifacts. The algorithms are also greatly efficient, and can be executed in reasonable time on sufficiently low cost devices.

3) *Support for a wide range of formats:*
Modeling in 3D is an arduous and time consuming task, and Metaio's support for obj, fbx and md2 model formats gave us greater flexibility in our choice of 3D modeling tool and made switching between and comparing the benefits of different formats easier.

4) *Direct loading of 3D models with abstraction of OpenGL calls:*
Unlike Vuforia and other SDKs which need 3D models in obj format to be converted to C++ header files (.h), Metaio directly reads the respective formats without conversion. This allows faster and more efficient use of smartphone resources like device memory and minimizes I/O load times.

V. Design and Implementation

We have chosen a hybrid tracking system[18] that uses integrated sensors embedded in the mobile device like accelerometer and 2D visual tracking to estimate pose and create an AR environment.The main objective is to calculate pose which consists of six degrees of freedom. Three of the six degrees of freedom are represented in terms of a translation vector $\bar{t}$, which relates the position of an observer relative to a plane and the remaining three in terms of a rotation matrix R that relates the orientation of an observer relative to a plane.
Accelerometer measures at least two of the three degrees of freedom, R is determined using samples from the accelerometer and $\bar{t}$ must be determined using 2-D visual tracking.

A. *Pose calculation Steps:*

1) *Accelerometer values:*

An accelerometer measures the acceleration of the sensor relative to an object in free-fall. The accelerometer measures the force of gravity along each of its axes x, y and z. It is calibrated to return an acceleration reading of 1g upwards along its z-axis when it lies flat on a table since it is accelerating upwards at 1g relative to an object in free-fall. Rotations are represented as a combination of the device's rotated axes readings. This combination forms the acceleration vector, a:

$$\vec{a} = [\overrightarrow{a_x a_y a_z}]^T \qquad (1)$$

Where $\overline{a_x}$, $\overline{a_y}$, $\overline{a_z}$ arethereadings from the accelerometer's x-, y- and z-axes respectively.
An accelerometer can only determine two of the three degrees of freedom since rotationaroundthe device's z-axis when it's lying ona flat surface will not result in a change in the force of gravity along any of its axes. We will "guess" the third rotation component: direction. An accelerometer inside a moving object returnsacceleration vectorsinstead of orientation vectors.We find the orientationvectors of the device by using a low-pass filter on consecutive readings of $\overline{a_i}$:

$$\overline{a_i} = 0.1 \times \overrightarrow{a_i} + 0.9 \times \overline{a_i - 1}$$

Where $\vec{a} = [\overrightarrow{a_x a_y a_z}]^T$ and $\overline{a_i}$ is theith orientation vector estimation.

2) *Calculation of rotation matrix:*

The rotation matrix, R, is a 3-by-3 matrix in the form:

$$R = [\overline{r_1}\overline{r_2}\overline{r_3}] \qquad (2)$$

Where$\overline{r_1}$, $\overline{r_2}$, $\overline{r_3}$ are the three 3-by-1 column vectors of R. $\overline{r_1}$ represents the device's orientation vector as a unit vector:

$$\overline{r_1} = \hat{a} = \frac{\overline{a}}{\|\bar{a}\|} \qquad (3)$$

Where $\|\bar{a}\|$ is the length of $\bar{a}$.

$\overline{r_2}$ influences the direction rendered models will face on the plane and shouldbe an arbitrary unit vector perpendicular to $\bar{a}$ definedby:

$$\overline{a_x}x + \overline{a_y}y + \overline{a_z}z = 0 \qquad (4)$$

Choosing x=0 and y=1 so that z=$-\frac{\overline{a_y}}{\overline{a_z}}$. We define this new vector, $\vec{b}$ as:

$$\vec{b} = [0 \quad 1 \quad -\frac{\overline{a_y}}{\overline{a_z}}]^T \qquad (5)$$

$\overline{r_2}$is then

$$\overline{r_2} = \hat{b} = \frac{\overline{b}}{\|\bar{b}\|} \qquad (6)$$

Where $\|\bar{b}\|$ is the length of$\vec{b}$
$\overline{r_3}$ must be the cross product of $\overline{r_1}$and $\overline{r_2}$so that it is orthogonal:

$$\overline{r_3} = \overline{r_1} \times \overline{r_2} = \hat{a} \times \hat{b} \qquad (7)$$

Substituting (3), (6) and (7) into (2) gives:

$$R = [\hat{a}\hat{b}\hat{a} \times \hat{b}] \qquad (8)$$

The OpenGL Model View matrix, $M_{modelview}$ is a 4-by-4 matrix that determines the rendering system's translation, rotation and scale. For multiplication with the 4-by-4 Model View matrix,$M_{modelview}$, the R matrix must be represented in homogeneous coordinates:

$$R = \begin{bmatrix} \overline{r_1} & \overline{r_2} & \overline{r_3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \hat{a} & \bar{b} & \overline{\hat{a} \times \hat{b}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying R with the model view matrix, $M_{modelview}$ correctly rotates the OpenGL view to reflect the orientation of the real world in the scene. We multiply these two matrixes by first initializing the Model View matrix and multiplying it by R:

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glMultMatrixf(R);
```

3) Calculating Relative Translation Between Two Frames:

We define $\overline{t_x}$ (and $\overline{t_y}$) to be the relative change in translation of acamera betweenthe current frame and the previous frame along the camera's xand y-axis in image coordinates.

When tracking points between consecutive frames using optical flow, we obtainanarrayrepresenting the relative movement of all matched points between frames. The array is given as a set of (dx,dy) movement values,one for each tracked point. To find $\overline{t_x}$and$\overline{t_y}$ we average all the dxand dy values for two consecutive frames. For n points tracked:

$$(\overline{t_x}, \overline{t_y}) = \frac{1}{n}\sum_{k=1}^{n}(dx_k, dy_k)$$

To find the change in scale between two consecutive frames ($\overline{t_z}$)is slightly trickier. Whenthe camera is translated along its z-axis that the movement of points in the top and left segments of an image move in the opposite direction relative to pointsin the bottom and right segments. therefore thearray of (dx, dy)values is split up into four quadrants:

$$Q1 = (dx_1[1], dy_1[1]), (dx_1[2], dy_1[2]), \ldots$$

$$Q2 = (dx_2[1], dy_2[1]), (dx_2[2], dy_2[2]), \ldots$$

$$Q3 = (dx_3[1], dy_3[1]), (dx_3[2], dy_3[2]), \ldots$$

$$Q4 = (dx_4[1], dy_4[1]), (dx_4[2], dy_4[2]), \ldots$$

With Q1 being the top left quadrant of the image, Q2 top right, Q3bottom left and Q4 bottom right. We then find$\overline{t_x}$and$\overline{t_y}$ for eachquadrant:

$$(\overline{t_{x,1}}, \overline{t_{y,1}}) = \langle Q1 \rangle$$
$$(\overline{t_{x,2}}, \overline{t_{y,2}}) = \langle Q2 \rangle$$
$$(\overline{t_{x,3}}, \overline{t_{y,3}}) = \langle Q3 \rangle$$
$$(\overline{t_{x,4}}, \overline{t_{y,4}}) = \langle Q4 \rangle$$

where the angle brackets represent averaging over the arrays Q1,Q2, Q3 and Q4.We then find$\overline{t_z}$ as follows:

$$\overline{t_z} = (\overline{t_{x,1}} + \overline{t_{x,4}}) - (\overline{t_{x,2}} + \overline{t_{x,3}})$$

Translation and scale is represented by the translation vector, $\vec{t}$ .We use an iterative approach to estimate $\vec{t}$, updating it accordingto the tracking results between each two frames. We use a poseestimate buffer to store three valuesrelated to translation that arethe total translations along each axis of the camera relative to aninitial starting position:

- Total translation along the camera's x-axis:$\overline{t_{x,total}}$.
- Total translation along the camera's y-axis:$\overline{t_{y,total}}$.
- Scale. Scale is equivalent to translation along the camera's z-axis:$\overline{t_{z,total}}$.

Overall translation, $\overline{t_{Total}}$, is the compound effect of all iterative measurements of change in translation between two frames since the start of tracking.

$$\overline{t_{total}} = [\overline{t_{x,total} t_{y,total} t_{z,total}}]^T$$

Every time a new frame is processed and a relative change in translationfrom theprevious frame is calculated, $\overline{t_i}$,we update$\overline{t_{Total}}$ by adding $\overline{t_i}$:

$$\overline{t_{Total,i}} = \overline{t_{Total,i-1}} + \overline{t_i}$$

Where$\overline{t_{Total,i}}$ is the ith estimate of$\overline{t_{Total}}$ and$\overline{t_i}$ is the relative changein translation between the current frame i and i-1.

B. Algorithm Steps[2]:

Initialization
1. Initialize the video captureand camera parameters and acquire device position information from the magnetic compass and accelerometer.

Main Loop
2. Freeze video input frame.
3. Getdevice position and orientation from the magnetic compass.
4. Calculate the camera transformation relative to the device position.
5. Draw the virtual objects in the center of the screen.

Shutdown
6. Close the video capture down.

C: *The App: The User Interface*

The user interface navigation screens in the app will be designed asnative Android layouts. The `WebView`class in Android enables loading and resizing of web pages at runtime, which will be used for the start screen. A few preliminary designs of the user interface include:

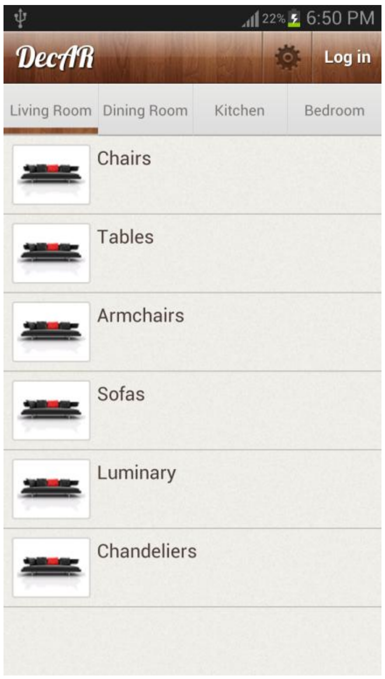
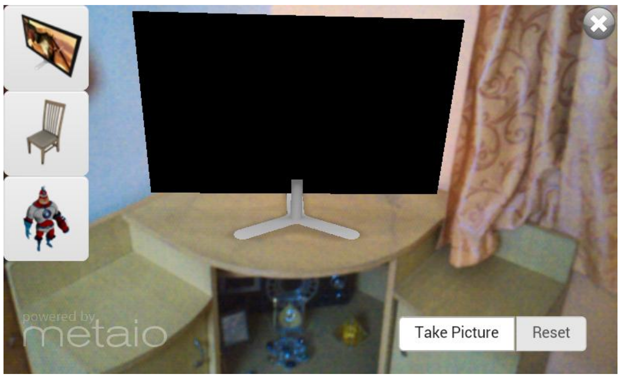


Figure 3:Item selection screen, level 1

Eclipse lets us design powerful and beautiful layout for the GUI, and Android, from 2.3 (Gingerbread) onwards, has significantly improved the `WebView` class to render web pages faster and make browsing and touch events more responsive.

The Rendering interface:

As the camera captures the image and renders the chosen item, items similar to current are shown in the top left corner. We believe this will improve accessibility and improve user satisfaction. A few screenshots of the app in action are as follows:



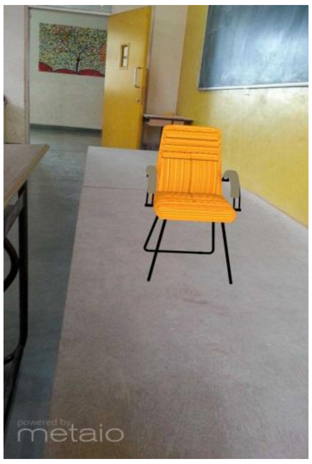


Figure 5:The app in action, rendering 3D objects without Fiduciary markers

D. *3D Modeling:*

We choose the open source tool, Blender for creating our 3D models because it is free and easy to use, and has extensive documentation. It also runs well on a moderately powered PC, like the ones the authors possessed.

The following is a screenshot of a model of a sofa under development:

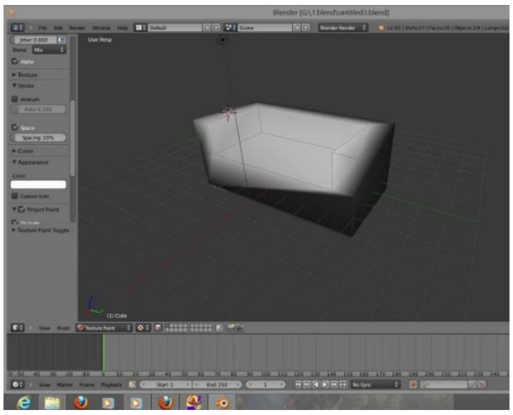


Figure 6: A sofa 3d model under development

The model requirements of the Metaio SDK were sufficiently flexible and resistant to tracking artefacts, without being too restrictive, unlike those of the Vuforia SDK from Qualcomm which is another reason we chose Metaio over Vuforia.

3D modeling was done on a desktop PC with an AMD Phenom II X4 Quad Core Processor, 3.5 GB RAM and on a Dell Insperon 15R laptop with an Intel i3 dual core processor and 3 GB RAM.

VI. OBSERVATIONS

1) Application works perfectly on high end phones but performance degrades on low end phones.
2) Very high quality models give some minimal artifacting.
3) Models are right now locally stored, resulting in high size of application and initial loading times.
4) Application can run in realtime video mode as well as static photo mode.
5) All models in a particular room are loaded immediately loaded when activity for that room starts, resulting in high memory usage and loading times.
6) Models possess low shadow and reflection effects, resulting in a not-so-realistic feel. In the future, models can be improved for cutting edge hardware.

VII. FUTURE SCOPE

1) The models can be stored on file hosting sites and loaded on demand. Although this will result in increased performance and low memory consumption and increased reliability, it will require the user to have a high speed internet connection.
2) This app can be further improved and extended for architecture, civil engineering, advertorial and other purposes.

VIII. CONCLUSION

This paper provided a brief introduction to handheld Augmented Reality, discussed the various free SDKs available for developing AR applications and outlined their merits and limitations. We also discussed the hurdles in achieving fast and efficient tracking on mobile devices.

In the context of this paper an Android application for interior decoration purposes is being developed using the Metaio SDK 4.1 described earlier.

IX. ACKNOWLEDGEMENT

The authors wish to thank all referees and all those who contributed to this project in their own small ways. Special thanks to Dr. Ahmad Raza Khan for providing inspiration to use the Android platform for this project and to Prof. RevatiWahul for providing guidance for this paper.

REFERENCES

1] Woohun Lee; Jun Park; , "Augmented foam: a tangible augmented reality for product design," Mixed and Augmented Reality, 2005. Proceedings. Fourth IEEE and ACM International Symposium on , vol., no., pp. 106- 109, 5-8 Oct. 2005
2] Tobias Domhan, “Augmented Reality on Android Smartphones”.
3] Gabriel Takacs, Vijay Chandrasekhar, Bernd Girod, RadekGrzeszczuk, “Feature Tracking for Mobile Augmented Reality Using Video Coder Motion Vectors”.
4] Kenji Oka and Yoichi Sato Hideki Koike, “An Augmented Reality Application for Previewing 3D Décor Changes”.
5] Agusanto, K.; Li Li; Zhu Chuangui; Ng Wan Sing; , "Photorealistic rendering for augmented reality using environment illumination," Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on , vol., no., pp. 208- 216, 7-10 Oct. 2003
6] Ruobing Yang; , "The study and improvement of Augmented reality based on feature matching," Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on , vol., no., pp.586-589, 15-17 July 2011
7] MartinKurze, Axel Roselius; ,” Smart Glasses: An Open Environment for AR Apps”, Deutsche Telekom AG, Laboratories &funijiInc
8] Keil, J.; Zollner, M.; Becker, M.; Wientapper, F.; Engelke, T.; Wuest, H.; , "The House of Olbrich — An Augmented Reality tour through architectural history," Mixed and Augmented Reality - Arts, Media, and Humanities (ISMAR-AMH), 2011 IEEE International Symposium On , vol., no., pp.15-18, 26-29 Oct. 2011Keil, J.; Zollner, M.; Becker, M.; Wientapper, F.; Engelke, T.; Wuest, H.; , "The House of Olbrich — An Augmented Reality tour through architectural history," Mixed and Augmented Reality - Arts, Media, and Humanities (ISMAR-AMH), 2011 IEEE International Symposium On , vol., no., pp.15-18, 26-29 Oct. 2011

9] BorkoFurht, “Handbook of Augmented Reality”, ISBN-10: 1461400635
10] http://dev.metaio.com/arel/overview/
11] http://dev.junaio.com/arel/documentationArelJS/
12] http://handheldar.icg.tugraz.at/artoolkitplus.php
13] https://developer.vuforia.com/
14] http://www.hitl.washington.edu/artoolkit/
15] http://wiki.blender.org/index.php/Doc:2.6/Manual
16] http://www.blender.org/education-help/
17] https://developer.vuforia.com/resources/sample-apps/frame-markers-sample-app
18] Carel P.J. van Wyk, “Markerless Augmented Reality on Mobile Devices with Integrated Sensors”