<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Mobile edutainment with interactive augmented reality using adaptive marker tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Lai, C; Wang, CL</td>
</tr>
<tr>
<td><strong>Issued Date</strong></td>
<td>2012</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10722/164936">http://hdl.handle.net/10722/164936</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>International Conference on Parallel and Distributed Systems. Proceedings. Copyright © Institute of Electrical and Electronics Engineers.</td>
</tr>
</tbody>
</table>
Mobile Edutainment with Interactive Augmented Reality using Adaptive Marker Tracking

Chi-Lam Lai, Cho-Li Wang
Department of Computer Science
The University of Hong Kong
Hong Kong
{cllai, clwang}@cs.hku.hk

Abstract — Augmented Reality (AR) is a great partner of Edutainment in motivating students and enriching a class. Students can check out additional digital information of physical items such as books, samples, exhibits or even sites. The information looks just like existing in the reality. In addition to a show of multimedia content, we opine that if the virtual objects augmented to the reality can act and react like real objects, the learning experience can be greatly promoted. We developed an AR system in that the virtual objects can interact with each others. We implemented several applications using the system to demonstrate the use of interactive AR in edutainment. We also attempted to resolve the limitation of image angle coming with vision-based AR by building a multi-marker mechanism using a cube structure with surface area approximation. We also discussed some challenges and issues that researchers or developers should take note of in pursuing AR development.

Keywords — edutainment; augmented reality; marker tracking;

I. INTRODUCTION

Edutainment is a portmanteau of “education” and “entertainment”. The biggest obstacle to any kind of learning or training is bore. Learning by making fun is an ideal way to inspire and attract students, especially kids. It is not a new concept. Going to zoo, playing scrabble and singing songs are all edutainment.

AR is a technology that displays computer-generated information on top of and aligning to some real objects [3]. The user feels like the digital information appears directly in the real world and is stick to physical objects. It is in contrast to another regime Virtual Reality (VR) in which the user is completely immersed into a synthesized digital world [4]. It is a fashion of pervasive computing in that information is integrated with the real world and is accessible anywhere anytime.

If the AR system goes beyond rare display and allows the virtual objects to interact with each other just like physical objects in the real world, the experience will be more realistic and fantastic. We see that interactive AR [14] has great potential for edutainment by its very unique and funny nature. AR can deliver a different education approach from traditional teaching software in that the students are not confined to the computer screen but can engage with the real things. The populace of smartphones and tablets has created new opportunities for AR edutainment. With the mobility an AR application can be embedded everywhere and accessible through a commodity device instead of specialized equipment.

There have been a number of AR applications developed for school class subjects, e.g. chemistry [16], sun-earth relationships [17], mathematics and geometry [18]. In general they had received positive feedbacks from the surveyed students that AR can enhance learning. However, their requirements on special display devices or other fixed environment setups would limit their practicality and readiness. A mobile and portable platform would be a breakthrough. In a story-telling application [19], there was an interactive feature that allows the user to choose the story ending using auxiliary markers. That interactivity was very encouraging. Notwithstanding, the markers there just served as a user interface to provide input values for the program. They functioned as an input device purely and did not role-play a subject themselves under the context of the application. Our goal of interactivity is not to provide an alternative input method to the AR application. What we want to achieve is that each object represent “something” under the “story” of the application, and that “something” have relationship and interaction among each others.

In this paper we design and implement a mobile AR system that allows interactivity between virtual objects resembling the real physical environment. We used ARToolKit [11], a vision-based AR software library, for the implementation. ARToolKit has an inherent limitation that the camera must capture enough surface of a pattern for recognition. If the plane carrying the pattern is out of view or in a marginal view, it cannot be detected. In this regard, we developed an adaptive multi-marker system using a cube with distinct markers on each faces. At least one marker can be detected anytime once the cube is in front of the camera. The detected markers (at most three) have different exposed surface to the camera and thus different reliability. We adopted an area approximation algorithm to select the marker with largest surface for virtual object alignment. We built the interactive AR system on both iPhone4 and iPad2 running on iOS 5 platform. We evaluated the performance of our system with respect to the object complexity, hardware configuration, and video resolution.

The rest of the paper is organized as follows. Section 2 presents the background of AR and edutainment, and also
discusses the merits and challenges of utilizing AR technology in the field of edutainment. Section 3 presents our proposed system with description of the platform, architecture and designs. Section 4 investigates the system with discussion on some observations and evaluation of the system performance. We conclude the paper and discuss the future work in Section 5.

II. BACKGROUND OF AR AND EDUTAINMENT

The development of AR can be dated back to 1968 [1], when Sutherland [2] built a head-mounted display (HMD) with miniature CRT and head position sensors to “surround the user with displayed 3D information”. Its main contribution was that the user’s head position was tracked and used to transform the displayed graphics, so that the user perceived a virtual world aligned with the real one. Without that, the overlaid graphics would be disconnected with the real world. The challenge of aligning the virtual and real objects according to the viewer’s physical position had become the so-called “registration” problem paramount to all AR systems.

The term “Augmented Reality” was coined by Caudell and Mizell [3] in 1992, as superimposing digital information on the visual field of a user with real-world registration. They appreciated AR for it required less computation resource than Virtual Reality (VR) because it only needed to render the overlaid objects instead of every pixel on the screen. In 1994, Milgram and Kishino [4] introduced the concept of “Mixed Reality”, with AR closer to the real side of the “Virtuality Continuum” (Fig. 1).

AR systems in early days typically used mechanical, ultrasonic or magnetic sensors for tracking user’s movement for registration. As processor performance increased and software technology improved, vision-based methods had been growing. In 1996, Rekimoto [5] proposed a tracking mechanism by recognizing the pattern of a printed 2D matrix marker and estimating its position and orientation relative to the camera. This marker-based registration had become widely used afterwards.

In 1997, Azuma [6] conducted the first survey on AR, in which he defined the following three characteristics which were widely acknowledged as the essential components of AR: (1) Combine real and virtual, (2) Interactive in real time, (3) Registered in 3-D. This technology-neutral definition had excluded some domains by principle, e.g. films with embedded computer graphics were not regarded as AR because they are not interactive media.

In recent years, a wide variety of AR technologies and applications have been developed [7][8]. Display system varies from optical see-through, video see-through to direct projection. Devices can be head-worn, handheld or spatial. Besides visual, augmentation can also work on aural or haptic sense. Various techniques, e.g. GPS, RFID, Wi-Fi, inertial, vision, are used in single or hybrid for tracking and registration. Researchers and developers have applied AR in numerous fields like advertising, navigation, industrial, military, medical, collaboration, education and entertainment.

The term “Edutainment” is a portmanteau coined by Robert Heyman in 1973. Technology has contributed to edutainment by delivering easily-accessed, multimedia and interactive presentation through computing devices. But a traditional computer-based learning environment has limitations. The teaching and learning occurs in the pure virtual world inside the computers. The students are bound to the computers in front of them. They become too attracted to the machine and are isolated from other students, teachers and the classroom. They sacrificed the chance of face-to-face collaboration with humans and see-and-touch of real objects, both are very important process of gaining knowledge and experience.

In contrast, AR works on physical objects. This distinction grants the following advantages [9]:

1) Seamless interaction: AR does not isolate the students from reality. It is only a virtual add-on to the real world. The students can still engage with humans and physical objects in the classroom. They can learn by collaborating and interacting with each other face-to-face.

2) Tangible interface: Manipulating physical objects is more intuitive than operating keyboard and mouse. And the message can also be conveyed more effectively.

3) Transitional interface: The real and virtual worlds are no longer separated and disjoint. The students can easily go to and fro between them.

Therefore we see a great potential in AR for edutainment. It is especially valuable to spatially-related class content that naturally lends itself to 3D space, e.g. science and engineering [10].

With the populace of smartphones and tablets nowadays, the mobile devices are a desirable platform for developing AR edutainment application. They are flexible and widespread, allowing more pervasive use of AR. They are powerful enough to run AR application. Moreover, they do not require an initial fixed equipment setup which in other words shifted the cost to the device owners.

There are a various edutainment AR app developed in the mobile platform. Common examples are location-based apps for sightseeing or museum guidance annotating an exhibit when the user is at certain locations. Vision-based apps are often used for magazines, books or other publications to pop out interesting 3D objects, animations or video. In these examples, the users’ main option is to watch. They can only interact with the virtual object by moving around themselves or the object to watch it at different angle. We think that if an application can show multiple virtual objects, and these objects have relationship and can interact with each other, then the application will be much more entertaining and have more potential to deliver educational message.

If interactivity is encouraged, the application shall most likely require the user to move and rotate the physical object in all directions, i.e. 6-degree-of-freedom (6DOF): three variables (x, y, and z) for position and three angles (yaw, pitch, and roll) for orientation [7]. Here vision-based tracking is
often the technology of choice. Being a closed-loop system, it can achieve pixel-perfect registration because the virtual objects are overlaid on the same image from which the pattern is recognized. It is more promising than open-loop system which only relies on sensor data (e.g. GPS) unrelated to the output image. Vision-based tracking also comes in handy for "unprepared" environment where no specific infrastructure (e.g. transmitters and sensors) is setup.

Marker is commonly used in vision-based AR [19]. There are restrictions in the marker’s design so that the algorithm has assumptions to recognize it, e.g., with a square border, black and white, and rotational asymmetric. On the other hand, a more sophisticated marker-less technology, i.e. Natural Feature Tracking (NFT) [15], is fast-growing. Instead of tracking a marker with simple and restricted pattern, NFT can track the natural pattern (or feature) of a physical object, such as the whole magazine cover. With NFT, the experience will be more seamless without the artificial markers. However, natural patterns are normally much more complex and irregular than marker patterns. It implies a more complicated algorithm and thus more expensive computation, which is unfriendly to resource-restricted mobile devices.

Either marker-based or marker-less, the vision-based technology has a fundamental limitation. It relies on a plane pattern to estimate the position and orientation of an object. That means the pattern should face to the camera at a reasonable angle showing sufficient surface area to allow the recognition. The best angle is definitely orthogonal to the camera direction (90 degree). If the angle is too small, the shape of the pattern will be deformed too much for the system to recognize. Also, different patterns may look similar in a small angle, mixing up the objects in the application. In extreme case, if the pattern is parallel to (or nearly) the camera direction, the tracking can just be impossible.

In this paper we discuss two enhancements for an AR system that will be beneficial to an edutainment application. First, the application will allow interactivity between virtual objects. There can be some rules or knowledge in the virtual world and certain kind of relationship between the virtual objects therein. Second, the marker in the physical world is adaptive to the camera angle to allow robust tracking and registration.

III. PROPOSED SYSTEM

A. Platform

We built the system for iPhone4 and iPad2 running on iOS 5 platform. Smartphones/tablets are very ideal mobile devices for AR application because they are popular, all-in-one and powerful. They are equipped with: built-in camera essential to vision-based tracking; GPS, accelerometer and 3-axis gyro needed by location-based registration; powerful processor to handle the heavy AR computation; high quality display for the rendered video; and networking capability for online services. Smartphones and tablets nowadays have got plenty of useful hardware bundled in a single device that can be used to develop interesting AR applications without the need to setup the environment.

We used ARToolKit [11] to build the system. ARToolKit is a software library widely used for building AR application. It has both marker-based and markerless (NFT) versions. In this paper, we adopted marker-based technology considering the reliability and battery consumption. ARToolKit works by first feeding in the video stream from the device’s camera. It then preprocesses the frames (e.g. binarization) to highlight the important features, and check if there is any predefined marker pattern in the frames. If one is found, it calculates its position and orientation (6DOF) relative to the camera, and stores the information in a 4x4 transformation matrix.

Next, the matrix is passed to the 3D rendering system of the platform to draw the virtual objects on top of the video aligning with the markers according to the position and orientation information stored in the matrix. iOS platform adopts OpenGL for Embedded System (OpenGL ES) [12] as the 3D rendering system. OpenGL ES is a subset of desktop OpenGL specialized for embedded systems e.g. consoles, mobile phones, appliance and vehicles. ARToolKit can use low-level OpenGL primitives for drawing. It has also incorporated the libraries for loading and drawing 3D model files of Wavefront OBJ or OpenSceneGraph (OSG) format. In our program, some 3D objects are constructed by open source tool Blender [13].

B. System Overview

The system is a middle layer between ARToolKit and OpenGL ES as shown in Fig. 2. In brief, it retrieves the matrices from ARToolKit when the latter has detected some markers in the video. The layer analyses the matrices and generate the appropriate 3D environment according to the required program logic. It is done by change the properties of the virtual objects. Finally the 3D environment is rendered by OpenGL ES. The layer consists of two components: i) Interactivity Engine, and ii) Cube Tracking Module. The Interactivity Engine aims to build up a virtual world with rules to govern the relationship between the virtual objects according to their geometric coordinate. The Cube Tracking Module aims to provide an adaptive and robust marker tracking with the help of the physical form of a cube object.

![Fig. 2 – System Architecture](image)
1) Interactivity Engine

In order to achieve interactivity between two virtual objects, we need to find out the relative distance, position and angle between two markers. Given such intelligence, we can implement interactive program logic. For example, a virtual “bulb” will light up with a “battery” is inserted into a circuit, a “mirror” will reflect a “laser beam”, an “object” will slide on a “ramp” according to “gravity” or the magnetic field of a “magnet” will attract the pointer of a “compass”. To achieve these effects, we need a common ground on which the geometry of different virtual objects in the system can be understood and compared on the same basis. The concept is to make use of the coordinate system.

In the system, each marker has its own local coordinate system with the origin (0, 0, 0) at the marker’s center (Fig. 3). The associated virtual object is initially based on this local coordinate. The coordinate of each vertex in the object, in this local sense, will keep unchanged no matter how the marker is moved before the camera. In other words, the virtual object will always “think” of its position relative to the marker disregard of the actual placement of the marker before the camera.

![Fig. 3 – Local coordinate of the marker](image)

On the other hand, the display system has its own coordinate system with the origin at the screen’s center (Fig. 4). This world coordinate is maintained internally by OpenGL ES.

![Fig. 4 – World coordinate of the display](image)

If we have two virtual objects in the system, we cannot relate them geometrically if we only consider their local coordinates. Both objects may have a vertex with the same (x, y, z) but the two points are actually at different locations as displayed on the screen. To bridge between the two virtual objects, we need the transformation matrix generated by ARToolKit. The matrix is stored programmatically as a linear 16-elements array and is retrievable from each marker object. The matrix defines the translation, rotation and scaling factors of the object. OpenGL ES requires such factors to generate the object in the global environment.

When we multiply a vertex by a transformation matrix, we “move” the vertex by translating, rotating and scaling it according to the values in the matrix. This process can be reversed by multiplying by the inverse of the matrix. Suppose a virtual object (VO1, with matrix M1) is to interact with another virtual object (VO2, with matrix M2). We can:

1. Take appropriate vertexes (Vref) from VO1 as reference points;
2. Compute Vref x M1 to get the world coordinates of the reference points;
3. Create an inverse matrix of M2 (M2-1);
4. Compute Vref (in world coordinate) x M2-1 so that the reference vertexes are transformed into the local coordinate of VO2;
5. VO2 and Vref are now in the same coordinate system (of VO2). We can compare and analyze their geometric to implement the interactivity logic.

The above describes the basic working mechanism of the Interactivity Engine. The exact logic could be more complicated and depends on what we want to do in the interactive activity, e.g. measurement of distance/angle between two objects, collision detection, gravity simulation, etc. The effect of the interaction is done by tampering the OpenGL context, e.g. changing the geometric, texture, lighting properties of an object. Animation can be done by updating the object’s property perpetually per each frame. We can also add sound effects or touch-screen events to enrich the application design.

2) Cube Tracking Module

As discussed above, the fundamental problem of vision-based AR with plane pattern is that the camera must capture the image of the marker in a sufficient angle. If the angle is too small, the marker image will be distorted and may result in misrecognition. To solve this problem, we attempt to make the system more reliable by using multiple markers to identify a single object. With at least one marker detected, the object can be rendered in the right position. That would increase the chance of detection when the user or the marker is moving around, and hence improved the robustness of the system.

A raw way to realize multiple marker tracking is to simply adding more markers on a surface, but that would add little value because the whole surface still faces the same angle problem. The markers would better come in 3D space so that some markers can still be captured by the camera even though some others are completely out of view. A cube is an ideal structure to pursue. No matter how it is oriented, it can show 1, 2 or 3 faces before the camera at the same time (Fig. 5). Its faces are rectangular which fit well with the square markers. It is simple and easy to make. Moreover, it can stand steady on a surface (e.g. table) that would grant convenience in designing applications.

![Fig. 5 – A cube can show 1 to 3 faces at the same time](image)
This module associates a virtual object with multiple distinct markers affixed on different faces of a cube. There will be at most three and at least one marker seen by the camera, making the detection more robust. The virtual object needs to be aligned with the cube. Therefore, this module needs to adjust the geometry of the virtual object according to which marker is detected so that object will always follow the movement of the cube rather than just appear on top of the detected marker.

When multiple faces are detected simultaneously, the question is how to choose the most appropriate one to provide the basis for calculating the geometry of the virtual object. Say in Fig. 5-C, there are three possible faces to choose. A simple implementation is by sequence of detection. For example, if the markers are detected in sequence 3, 1 and 2, then at last marker 2 is selected to render the object. However, it is obvious in this example that marker 2 is not a good candidate because it has the least surface area exposed to the camera. Its reliability is lower than the other two markers. Therefore, a better implementation should select the marker with largest exposed surface area.

Calculating the exact surface area of the detected markers would be very expensive given that this computation has to be done per frame at runtime. Instead we implemented an algorithm that approximates the “remaining ratio” of the surface area, i.e. the percentage of surface area that remained viewable. The concept is that if we rotate a surface along a particular axis, the remaining surface area in front of the viewer is simply approximated to be inversely and linearly proportional to the angle rotated (Fig. 6). If two axes are rotated, then we approximate the combined effect by multiplying their corresponding ratio together.

First the algorithm calculates the rotated angle along x-axis (x-angle) and y-axis (y-angle) respectively. It omits the z-axis rotation because it has no effect on surface area. It then calculates the remaining ratio for x-axis (x-ratio) and y-axis (y-ratio) by:

\[
x\text{-ratio} = \frac{90^\circ - \text{x-angle}}{90^\circ}
\]
\[
y\text{-ratio} = \frac{90^\circ - \text{y-angle}}{90^\circ}
\]

For example, if the surface is rotated 30°, the remaining ratio would be 60 / 90 = 0.667. For 45° rotation, the ratio is 45 / 90 = 0.5. For 60° rotation, the ratio is 30 / 90 = 0.333. The ratio for x-axis and y-axis are then multiplied together to combine the effect (e.g. for x-ratio = 0.333 and y-ratio = 0.667, the overall remaining ratio is 0.333 x 0.667 = 0.222). Consequently, the ratios of all detected markers are compared and the one with largest ratio is used for rendering the object.

IV. IMPLEMENTATION AND EVALUATION

First, we demonstrate the possible uses of mobile and interactive AR for edutainment by implementing several programs based on the system described in Part III. These programs target at young kids and show some basic rules of physics in optics, electricity, magnetism, gravity and collision. In these programs, there are two or three objects to manipulate that can interact with each others dynamically.

![Fig. 6 – Approximation of remaining ratio](image)

![Fig. 7 – Demonstrations of Interactivity](image)
plugged into the circuit in right direction (Fig. 7-B2). If the battery’s direction is reversed, the bulb will not light up. If a battery with higher voltage is inserted, the bulb will be brighter. In magnetism, the pointer of the compass will follow the magnet if the latter comes close enough (Fig. 7-C1, C2). In gravity, a bowl of noodle marks the “ground” plane for gravity simulation (Fig. 7-D1). When the pan tilts, the egg on it will slip along the pan (Fig. 7-D2) with acceleration related to the angle ($\alpha$) between the pan and the “ground” with the rule: acceleration = gravity x sin $\alpha$. In collision, a revolver fires a bullet when the user touches the screen (Fig. 7-E1). When the bullet hits the target, the target will collapse (Fig. 7-E2).

Second, we demonstrate the cube tracking algorithm for multi-marker registration. A cube is made with distinct markers on five different faces and the remaining face attached to a handle for easy manipulation (Fig. 8-A). A cone with brick texture is aligned on top of the cube. In our trial, the cone can be rendered at the right position regardless of the cube’s orientation as long as the cube appears before the camera (Fig 8-B,C). The marker selection algorithm works as it is expected. The marker with the largest exposure (and hence more reliable) is selected to provide geometry of the object, which makes the rendering steadier and prevents any false detection due to marginally exposed marker.

![Fig. 8 – Demonstration of Cube Tracking](image)

In the course of implementation and testing, we encounter two major problems that greatly affect the marker detection accuracy and reliability: reflected light and background noise. Besides, we also have a number of observations in relation to the marker design and performance issues.

### A. Reflected Light

When the environment is under a direct light source, the marker will likely reflect the light too sharply resulting in a significant bright spot on the marker pattern. ARToolKit will interpret the bright spot as a white area, breaking the marker's pattern and making it unrecognizable. This phenomenon is frequent when we are near a window or a lamp. A lesson learnt is that if we want to run a marker-based AR application successfully, we need soft and ambient lighting in the environment. This implies that we may face considerable difficulties when we want to apply marker-based AR at outdoor environment.

### B. Background Noise

The second problem is background noise. If the background of the scene is very clean and plain, it will not cause any trouble to the system. However, if it is messy, i.e. scrambled with different objects and structures constituting a lot of lines and shapes, such “noise” may be misinterpreted as marker patterns by ARToolKit. Unexpected detection will occur even though no marker is actually there (Fig. 9). This problem gets worse if the AR application is pre-set to recognize many markers because it increases the chance that the noises hit the expecting patterns. In most cases the noises are just marginally resemble a pattern and so the false detections are sporadic. The effect is that the virtual object flickers across the screen.

![Fig. 9 – False detection without a marker](image)

In view of this, we tried to design a policy to remove the noise based on our observations. For a correctly detected marker $A$, the system will generate a notification per frame resulting in a stream of consecutive detection events:

<table>
<thead>
<tr>
<th>Events: $AAAAA ...$</th>
<th>Time:</th>
</tr>
</thead>
</table>

If a marker B is added to the scene at time $t$, the event stream will be like:

<table>
<thead>
<tr>
<th>Events: $AAAAAAAABABABA ...$</th>
<th>Time: $t$</th>
</tr>
</thead>
</table>

A detected marker will generate at least several notification events even though it is just presented to the camera for a short instance, say 1 second. Therefore, in normal cases we should not observe a single and sporadic occurrence of marker detection within the stream like the $C$ in:

<table>
<thead>
<tr>
<th>Events: $AAAAAABCABABABA ...$</th>
<th>Time:</th>
</tr>
</thead>
</table>

Meanwhile, such pattern is observed when noise exists. It is so because the background pattern causing the noise is only marginally resemble the real marker pattern. Therefore such false pattern cannot be steadily detected. Having that said, our policy will consider a marker validly detected only when...
it is detected consecutively for \( n \) time. It is implemented by maintaining a counter to store the number of consecutive detection of a marker. A high level description is in Table I. This mechanism filters out the noises with sporadic appearances within a stream of legitimate marker detection events. We tested the policy starting from \( n = 2 \). It can indeed reduce the noises. As we increase \( n \), more noises are removed. However, a larger \( n \) (i.e. \( > 4 \)) induces a drawback that the virtual object aligns with the marker less responsively because some valid detection events are also discarded and so the virtual object’s geometry is not updated as frequent. E.g. for \( n = 5 \), even a valid marker needs to accumulate five detections in order to be endorsed and used to update the 3D object. From our experiment, we find that \( n = 2 \) or \( 3 \) is the optimum balance in a messy background.

### Table I. Noise Removal Policy

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>a marker is detected</td>
<td>if a marker is detected then</td>
</tr>
<tr>
<td></td>
<td>if the marker starts to appear then</td>
</tr>
<tr>
<td></td>
<td>( \text{counter} = 1 )</td>
</tr>
<tr>
<td></td>
<td>else if the same marker is detected recently then</td>
</tr>
<tr>
<td></td>
<td>( \text{counter} ) ( ++ )</td>
</tr>
<tr>
<td></td>
<td>endif</td>
</tr>
<tr>
<td></td>
<td>if counter ( \geq n ) then</td>
</tr>
<tr>
<td></td>
<td>update 3D geometry by the detected marker</td>
</tr>
<tr>
<td></td>
<td>endif</td>
</tr>
</tbody>
</table>

### C. Marker Design

A well-designed marker is also critical to the success of an AR application. Besides fulfilling the basic criteria, a marker should also be easily distinguished by machine, especially if numerous markers are used in the application. Otherwise, the system may be confused by the similarity of marker patterns and is prone to erroneous detection. When a marker is facing straightly to the camera, normally it will be detected safely. Confusion always occurs when the marker rotates away, that effectively compresses the image and makes it easier to mix up with other markers. An example is shown in Table II. The left column shows two markers in their front face views and the right column shows their respective views when they are rotated at extreme angle. We can see that the distinction between the two markers diminishes at such angle compared to their front face views. That will probably cause confusion to the system in practice.

This problem is associated with plane markers and can be relieved by the cube tracking design. In our implementation, we estimate the surface area of the exposed faces of the cube and choose the largest one for reference to draw the 3D object. And we can assure that, out of the three faces (at most) shown up from a cube, the most exposed face will always not be too extremely rotated away from the camera. That will effectively avoid the problem of extreme marker angle.

### Table II. View of Marker at Extreme Rotation

<table>
<thead>
<tr>
<th>View of Marker</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Front Face</td>
<td>![Front Face Image]</td>
</tr>
<tr>
<td>Rotated Away</td>
<td>![Rotated Away Image]</td>
</tr>
</tbody>
</table>

### D. Performance

The performance of the system depends largely on the video resolution. Our applications perform nicely in 640x480 for iPad2 and 480x360 for iPhone4. The augmented video can play smoothly with the implementation of interactivity and cube tracking. We can see that the processing power of iPhone4/iPad2 can sufficiently handle a workload of this kind. However, the frame rate drops significantly when we push the resolution higher. In a vision-based AR application, the major workload is the image processing and pattern recognition, in which the number of pixel determines the processing time. From this, another lesson is learnt that in designing a vision-based AR system we need to strike a balance between the system performance and the required video resolution.

We have evaluated the program performance by using Apple’s development tool “Instruments” and selecting the option “OpenGL ES driver” to measure the frame per second (fps) of the augmented video stream of the applications in runtime. The setup was iPhone4 with 640x480 video. We have benchmarked our applications in different conditions to illustrate the program overhead. First, no marker was presented to the camera, so there was no detection and no 3D rendering. The frame rates here were the result of the per-frame pattern recognition of ARToolKit. Second, we showed the markers, but turned off the middle layer and let the program just render the objects straight. The figures here included the overhead of object registration and OpenGL rendering. Lastly, we run the full programs to see the effect of our middle layer. The result is shown at Table III where the frame rates were obtained by average. For reference, the number of vertexes in each application is also tabulated.

### Table III. Frame Rates (fps) Showing Program Overhead

<table>
<thead>
<tr>
<th>Object</th>
<th>No. of Vertexes</th>
<th>No. detection</th>
<th>No. middle layer</th>
<th>Full program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optics</td>
<td>807</td>
<td>26</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>Electricity</td>
<td>17,418</td>
<td>26</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>Magnetism</td>
<td>707</td>
<td>26</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>Collision</td>
<td>6,127</td>
<td>26</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>Gravity</td>
<td>1,814</td>
<td>26</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>Cube Track</td>
<td>34</td>
<td>26</td>
<td>20</td>
<td>16</td>
</tr>
</tbody>
</table>
From the result we observe a gap (~6 to 13 fps) between “no detection” and “no middle layer”. It reflects that the workload of object registration and OpenGL is heavy. It involves calculating the markers’ geometry and all those matrix operations used to align the objects to the markers. The application Electricity demonstrates a more significant drop because it contains much more vertices than the other applications, giving more rendering work to OpenGL. From this we know that OpenGL brings a scalability issue to an AR system in terms of the geometry complexity. Our middle layer also imposes certain overhead (~1 to 4 fps) depending on the complexity of the implemented logic.

We also compared the performance of the system in iPhone4/iPad2 under different video resolutions. Since Electricity contained the most vertices, we used it as the benchmark program. The result is shown at Table IV where the values were obtained by average. We see that iPad2 performed considerably better, yielding a nice 26-fps up to 640x480. iPad2’s superiority over iPhone4 is as expected due to its more powerful Apple A5 dual-core chip, comparing with its predecessor, the A4 single-core chip used in iPhone4 (Apple claims that A5 can push graphics up to seven times faster and is more energy efficient). Nonetheless, iPhone4 can still run the system nicely at 480x320. At present, the A5 chip has already been deployed in the newer iPhone4S and the 3rd generation iPad has used an even more advanced A5X (dual core CPU + quad-core GPU). The increasing computation power and energy efficiency of the mobile devices will surely facilitate the development of mobile AR applications.

**TABLE IV. FRAME RATES (FPS) UNDER DIFFERENT DEVICES AND RESOLUTIONS**

<table>
<thead>
<tr>
<th>Resolution</th>
<th>iPhone4</th>
<th>iPad2</th>
</tr>
</thead>
<tbody>
<tr>
<td>192x144</td>
<td>16</td>
<td>26</td>
</tr>
<tr>
<td>480x320</td>
<td>14</td>
<td>26</td>
</tr>
<tr>
<td>640x480</td>
<td>10</td>
<td>26</td>
</tr>
<tr>
<td>1280x720*</td>
<td>2</td>
<td>11</td>
</tr>
</tbody>
</table>

*The physical display resolution of iPhone4 is 960x640 and iPad2 is 1024x768. At 1280x720, the in-take video will be shrunk to the devices’ limit for display.

V. CONCLUSION AND FUTURE WORK

In this paper, we discussed how to achieve interactivity between virtual objects in a vision-based AR application. We implemented several applications to demonstrate the use of interactive AR in edutainment. We see that the potential of mobile edutainment is quite promising with interactive AR. We also studied the fundamental limitation of vision-based AR and demonstrated a cube tracking algorithm to handle the angle problem of the pattern surface. We evaluated the system and benchmarked its performance for different devices and video resolutions. In future we will investigate the use of NFT instead of markers. NFT can offer more intuitive experience since it does not need any specific marker which always looks abrupt and alien in the scene. For example, a marker on a magazine cover will damage the artistic design. We will also explore the integration of more context information, e.g. GPS, sound, gesture, to build more living applications.

REFERENCES