

Intuitive User Interface for Mobile Devices Based on Visual Motion Detection

Stefan Winkler, Karthik Rangaswamy, ZhiYing Zhou

Department of Electrical and Computer Engineering
National University of Singapore
Singapore 117576

ABSTRACT

The small form factor and unergonomic keys of mobile phones call for new and more natural approaches in user interface (UI) design. In this paper, we propose intuitive motion-based UI controls for mobile devices with built-in cameras based on the visual detection of the device's self-motion. We developed a car-racing game to test our new interface, and we conducted a user study to evaluate the accuracy, sensitivity, responsiveness and usability of our proposed system. Results show that our motion-based interface is well received by the users and clearly preferred over traditional button-based controls.

1. INTRODUCTION

Mobile phones and PDA's are becoming increasingly powerful and provide more and more functionality to the user. However, their user interface is severely limited due to the small form factor of the devices. Using the small, unergonomic keys for control in applications other than calling is tedious and not always intuitive.

In order to overcome these problems, we have developed novel, motion-based controls as an interface for mobile devices. The controls are based on the detection of the 3D self-motion of the device using the integrated camera. Possible movements include translation and rotation with a total of 6 degrees of freedom. These movements can then be used to control applications on the device. The advantage of using a camera-based method is that – contrary to other specialized motion sensors – an integrated camera with good resolution is readily available on today's mobile devices.

A central part of this research is the 3D motion detection using the phone's built-in camera. There are various methods for motion detection and object tracking.¹ However, here we are interested more specifically in computing the pose of the phone in 3D space from the motion registered by the integrated camera.² The detection has to be performed in real-time under the constraints of the limited memory and computing resources available on a hand-held device.^{3,4} Work on this subject is very recent, and only few studies have been published on device/camera-based motion controls and their use for mobile gaming.⁵

For the work described in this paper, we initially use fiducial markers to simplify the detection of the phone's motion. The position and orientation of the phone in 3D space relative to the marker can be estimated by analyzing the frames returned by the camera for any detected markers and calculating the transformation matrix from marker coordinates to camera coordinates. The transformation matrix is then mapped to various controls in our motion-based interface to drive applications on the mobile device.

The application we developed to test our motion-based user interface is a car-racing game, which involves standard controls for steering and speed. We define certain phone movements that translate into these basic controls on our motion-based user interface in order to maneuver the car.

The technical issues mentioned are only one aspect of the problem; the other is usability. Does the proposed interaction technique feel natural and intuitive to the user? Is it robust and responsive enough? Does the user prefer it over traditional button-based input for a given application? Therefore, we conduct a user study in order to compare the performance and acceptance of our motion-based interface with that of a button-based interface

E-mail of corresponding author: winkler@nus.edu.sg

and also evaluate certain additional features, which we added to the system to enhance the immersive feel for the user.

The paper is organized as follows: Section 2 explains the tools and method for detecting the 3D self-motion of the phone. Section 3 discusses the specifications of the hardware used and evaluates the performance of our system. Section 4 describes the features of our car-racing game and the motion-based interface we designed for this application. Section 5 discusses the user-study conducted and the results of the survey. Concluding remarks are given in Section 6.

2. CAMERA-BASED SELF-MOTION

Our implementation is based on tracking fiducial markers using the phone's camera to determine the device's position and orientation in 3D-space with respect to the marker. The marker tracking is implemented with ARToolkitPlus,⁶ a modified version of ARToolkit that can be compiled for the mobile platform. ARToolkitPlus includes many new features over the original ARToolkit, namely new pixel formats for images returned by the mobile phone's camera (like RGB565), fixed point arithmetic to replace some of the computationally intensive floating-point operations, automatic thresholding to compensate for varying lighting, and ID-encoded markers instead of template-based recognition that require no prior training and at the same time do not affect the speed of operation. Figure 1 shows an ID-encoded ARToolkitPlus marker and our mobile device in the foreground.



Figure 1. ID-encoded ARToolkitPlus marker used for computing the pose of the mobile device in 3D space.

The user moves the phone over a set of square markers of known size, which are tracked by the phone's camera. Figure 2 provides a geometric perspective to this problem. In their work on ARToolkit, Billinghurst and Kato⁷ give a detailed mathematical treatment to the problem of marker detection, recognition and pose estimation. The frames returned by the phone's camera are thresholded, and regions whose outline contour can be fitted by four line segments are extracted. Parameters of these four line segments and coordinates of the four vertices of the regions found from the intersections of the line segments are stored for calculating the transformation matrices from marker coordinates to camera coordinates. The regions bounded by the line segments are normalized, and the sub-image within the region is analyzed to identify specific user-defined markers. Once the transformation matrices are determined, the translation and orientation components are extracted, which define the phone's pose with respect to the marker in 3D space. This process is carried out for every frame returned by the camera.

The translation and orientation components of the transformation matrix as well as their changes over time are then mapped to various application-specific controls within the motion-based user interface. This is explained in detail in the subsequent sections.

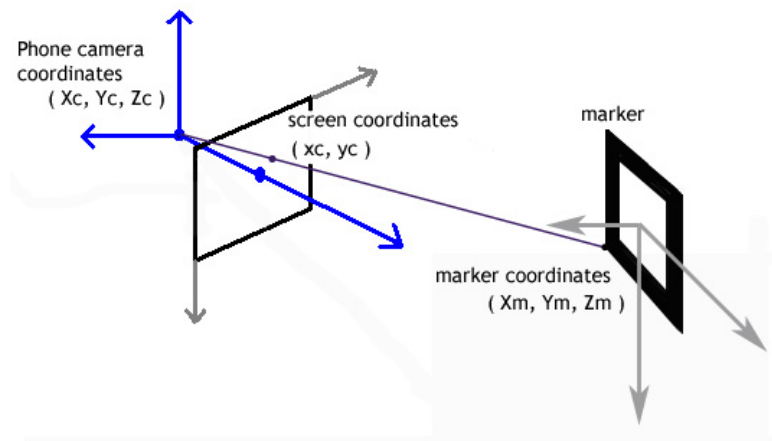


Figure 2. Transformation from marker to camera coordinates – a geometric perspective.

3. SYSTEM DESIGN & PERFORMANCE

Our application runs on a Hewlett-Packard rw6828 PocketPC equipped with a 416 MHz Intel PXA272 processor and 64MB RAM, running Windows Mobile 5.0 OS. The phone has a built-in 2-megapixel camera that is used to track the markers at a video resolution of 176x144 pixels. We developed a DirectShow FrameGrabber filter to capture incoming frames from the device's camera. ARToolkitPlus accepts these frames and returns the transformation matrix describing the 3D pose of the phone if a marker is detected in the frame. The obtained translation and orientation components are then mapped to different controls on the user interface to drive the application. The complete flow of control in the system is summarized in Figure 3.

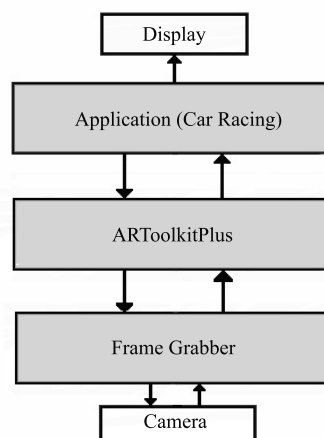


Figure 3. Control flow in the system.

As our UI targets mobile devices, people expect to use them anywhere, and thus robustness in different environments becomes a major issue. Accurate detection and recognition of the markers at different distances from the phone and under varying lighting conditions are two major factors that affect the robustness of the system. This is further aggravated by the poor video resolution of the built-in camera.

ARToolkitPlus provides two ways of dealing with the lighting issue. If the lighting is going to be relatively constant, the threshold can be fixed at a pre-defined value. For varying illumination conditions, ARToolKitPlus can do dynamic thresholding by looking at the marker content (pattern) in the frame and taking the average

between the darkest and brightest pixels. If no marker is found the threshold value is randomized. This process is not computationally intensive and hence does not affect the application’s speed of operation. Given the intended use of our system, the second method is clearly the better choice.

We evaluated the accuracy of marker detection by comparing the position of the phone as computed with ARToolkitPlus to the actual physical distance between the marker and the phone. This was done using a square marker with a side-length of 8 cm that was moved towards and away from the phone in the depth direction. The measurements were repeated at various slant angles (0, 30, 45, 60 and 75 degrees). Figure 4 shows the displacement errors measured using this procedure. From the graph, it can be inferred that as long as the phone is held approximately parallel to the marker (no slant), the displacement error is minimal and will not affect the accuracy of the application. Yet accuracy decreases if the phone is slanted with respect to the marker plane.

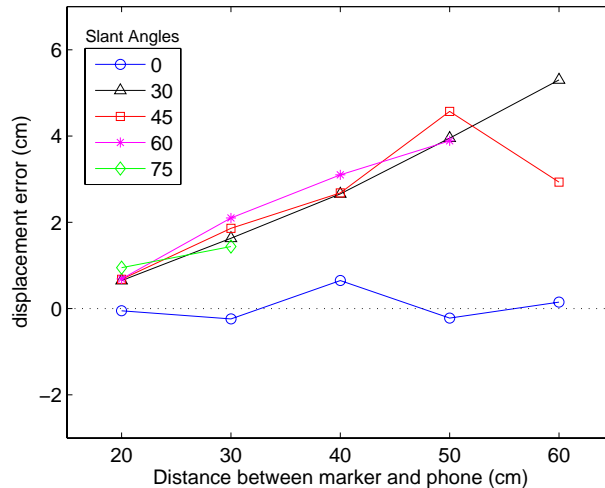


Figure 4. Error of marker-based position estimates as a function of distance and slant.

The measurements were limited to a range of 20-60 cm, as the marker was either too large or too small to be detected reliably within the camera’s field-of-view beyond this range for the above-mentioned marker size of 8 cm. Based on the graph, we can define a range of use for the phone, where the errors of the computed pose are minimal and do not affect the application much.

4. APPLICATION

The car racing game was developed in C++, making use of the Direct3D Mobile libraries for graphics design. The main controls of the game are the steering wheel and the accelerator. In our interface the user can handle the phone directly as both a steering wheel and an accelerator in order to maneuver the car. To steer the car, the user tilts the phone to the left or right about the depth direction. For acceleration and deceleration, respectively, the user moves the phone towards or away from the marker in the depth direction, as shown in Figure 5.

We use the distance between the phone and the marker along the depth direction as an indicator of the car’s speed, and the rotation about the depth direction to determine the direction and extent of turning.

In addition to the basic controls, we introduced two different modes of display for the game, namely a true-horizon mode and a relative-horizon mode, which are illustrated in Figure 6. In true-horizon mode, we design the interface so that the in-game horizon remains parallel to the ground irrespective of the tilt of the phone. This assumes the marker orientation with respect to the ground is fixed and known. In relative-horizon mode, the in-game horizon is parallel to the base of the phone’s display. Finally, we implemented an engine sound whose pitch is proportional to the car’s speed as an additional speed indicator.

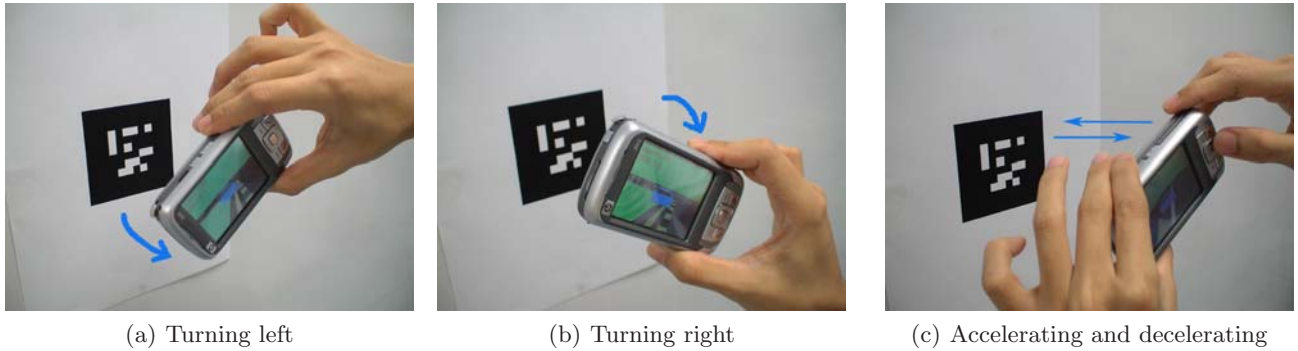


Figure 5. Motion-based controls: (a,b) The car is steered left or right by tilting the phone in the corresponding direction. (c) The car is accelerated or decelerated by moving the phone towards or away from the marker respectively.

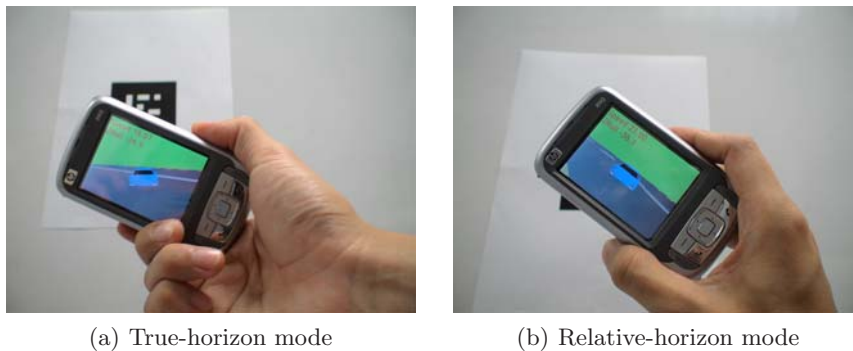


Figure 6. Display modes: (a) In true-horizon mode, the in-game horizon is parallel to the ground. (b) In relative-horizon mode, the in-game horizon is parallel to the phone's base.

The complete application, including marker detection, pose computation, graphics display and sound output, runs at a rate of about 10-12 frames per second on our PocketPC.

5. USER STUDY & DISCUSSION

We conducted a user study to compare the proposed motion-based interface for the car-racing game with traditional button-based controls in terms of accuracy, sensitivity, responsiveness and usability. The user study was designed to evaluate the application controls as well as the other features of our system described in Section 4. For this purpose, we developed a button-based interface for the same car-racing game that we had designed to run with the motion-based controls. In the button-based interface, the four directional buttons are used to maneuver the car. The left and right arrow buttons steer the car in either direction, and the up and down arrow buttons accelerate and decelerate the car, respectively.

15 male participants and 13 female participants between 15 and 16 years of age took part in the study. They were asked to play the game using both interfaces. All the participants had prior experience with mobile phone games, and over 90% of them had played a car-racing game on a mobile phone before. Each question in the survey required the user to give a rating between -3 and 3 comparing various aspects of the game, where the ends of the scale indicate a clear preference for one or the other, and zero represents the neutral point.

The user ratings for the accuracy in steering and speed control with both interfaces are shown in Figure 7. The average ratings are 0.85 and 0.82, with variances of 2.77 and 2.58, respectively, indicating that the motion-based interface was preferred over the the button-based interface. We expected a higher rating for steering, but the location of the camera at the corner on the back of the phone plays a major role here. When the phone is tilted to steer the car, the marker has a high probability of disappearing from the camera's field-of-view. Despite the fact that this required some getting used to for the participants and was pointed out by them during the

informal feedback session, most participants agreed that our new design was a better alternative to steering with buttons on the phone.

Having a camera at the center of the phone’s back would certainly be better for our interface. However, there are two alternative solutions for this issue. One would be to have a 2D array of coplanar markers, so that at least one marker would remain in the field-of-view of the camera at all times. Another solution would be to determine a transformation matrix from the actual camera’s position to the center of the phone and multiply it with the transformation matrix obtained from ARToolkitPlus.

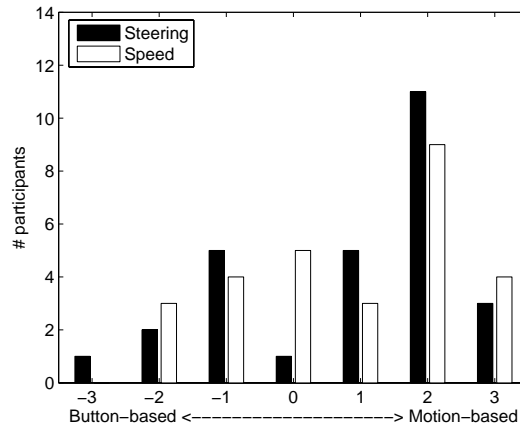
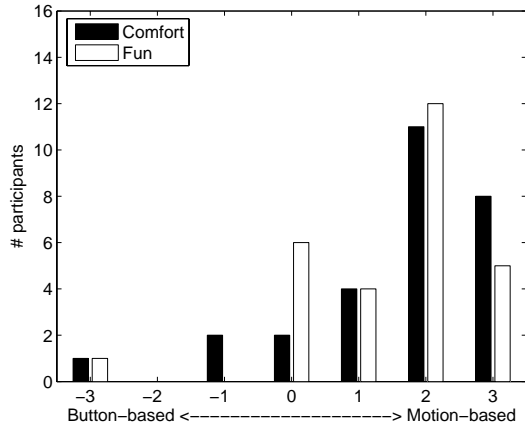


Figure 7. Comparison of button-based and motion-based interfaces in terms of accuracy for steering and speed control.

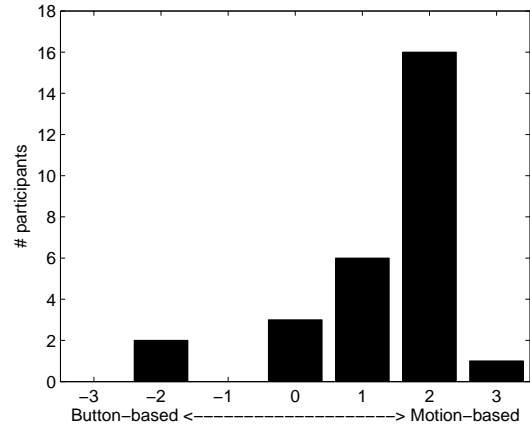
For acceleration and deceleration, people whose ratings were more supportive of the button-based interface stated that although our method was intuitive and better than accelerating the car with the phone’s keys, moving the phone towards or away from the marker was a bit strenuous. Furthermore, the game played by the participants was only a simple simulation of a car on a road without much environmental detail, as we only wanted to test the controls and not have a full-fledged game. Yet the users opined that additional graphical features like textures and objects by the side of the road would give them a better notion of the current speed. On the other hand, if the game is to be made more graphics-intensive, the frame-rate is sure to decrease further, unless better graphics hardware becomes available for mobile devices.

The user ratings for the level of comfort using the phone, the fun generated during game-play and overall enjoyment are shown in Figure 8. The average ratings are 1.61, 1.43 and 1.32 with variances of 2.09, 1.75 and 1.36 respectively, indicating a strong preference for our interface over playing with buttons. In the feedback session, almost all participants appreciated the innovative approach to game play and wanted such a technology to drive games and other applications on the mobile platform. They also wanted the car-game to be developed into a complete product and released commercially. From this feedback, we believe that our way of controlling the game certainly has an edge over the traditional method and can be th basis for the design of intuitive interfaces customized for various applications.

The user ratings for the two display modes (true-horizon vs. relative-horizon) are shown in Figure 9(a). The average rating is 0.07, which indicates that people do not have any specific preference for one or the other. A problem we encountered here is that in our implementation the rotation of the in-game horizon to keep it parallel to the ground is limited by the frame rate of the application, which made the game world jitter whenever the phone was tilted. This was also reported by the participants when they were asked about their rating for this question. However, the problem can be solved relatively easily with simple interpolation. Other participants stated that the horizon rotation creates a dizzy feeling when steering the car. The rest appreciated it as a unique mode of display that is very suitable for applications such as our car-racing game. Even if opinions over the display mode were divided, a true-horizon display mode would be a necessity for applications involving rotations of more than 60-90 degrees.



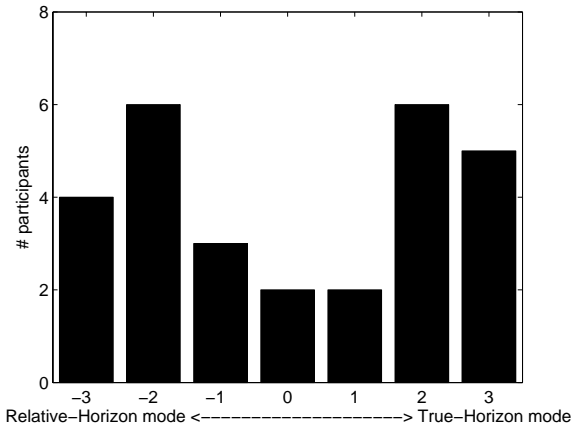
(a) Comfort and fun during gameplay



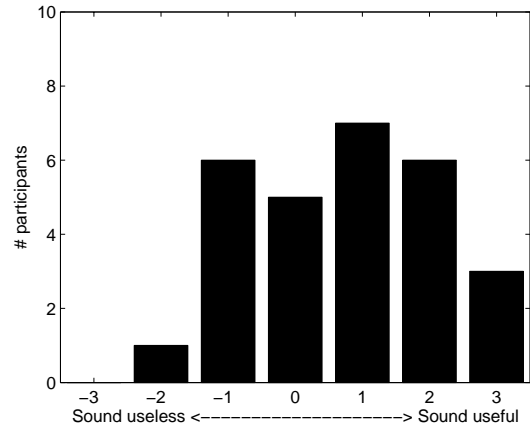
(b) Overall rating

Figure 8. Comparison of button-based and motion-based interfaces in terms of (a) comfort/fun during game-play and (b) overall enjoyment.

Figure 9(b) shows user ratings for the pitch-varying engine sound as a speed indicator, with an average of 0.61. A major shortfall of our system is that the speakers of the phone do not have enough fidelity or power to deliver a realistic engine sound with pitch variation. The user comments confirmed that this feature would be more appreciable with improved sound quality. Additional sound-processing hardware would certainly be a boon to this application. Furthermore, pitch variation is heavily affected by frame-rate, which has to be high enough to give a clear notion of the sound’s continuity even when the pitch varies.



(a) Horizon display modes



(b) Sound as a speed indicator

Figure 9. (a) User ratings comparing the two display modes (true-horizon vs. relative-horizon display). (b) User ratings of the pitch-varying engine sound as a speed indicator.

In all the user ratings discussed above, the average ratings show the acceptance of our proposed interface as a better alternative to using buttons on the phone, and in general as a fun and intuitive way of interacting with mobile applications.

6. CONCLUSIONS

We presented a car-racing game for mobile devices with an intuitive motion-based interface to control the game. This interface is based on pose estimation from the frames captured by the phone's camera using ARToolkitPlus. We conducted a user survey in order to compare our motion-based interface with a traditional button-based interface. The results of the survey show that our proposed interface is received well and clearly preferred over using the buttons on the phone, despite certain shortcomings that can be corrected with minor modifications of the system.

Our future work will focus on enhancing the system with marker-less methods of detecting the phone's pose and movement in 3D space. Techniques for this include feature tracking and optical flow.

The integrated communication capabilities of mobile phones also make them ideally suited for multi-user games or collaborative applications, be it via GSM/3G networks, WiFi or Bluetooth.

Finally, we are developing other applications, in which motion-based controls can help make the user's interaction more intuitive. One example is navigation over a high-resolution image such as a map.

REFERENCES

1. E. Foxlin, "Motion tracking requirements and technologies," in *Handbook of Virtual Environment Technology*, K. M. Stanney, ed., ch. 8, pp. 163–210, Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 2002.
2. E. Foxlin, Y. Altshuler, L. Naimark, and M. Harrington, "Flighttracker: A novel optical/inertial tracker for cockpit enhanced vision," in *Proc. 3rd IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 212–221, (Arlington, VA, USA), November 2004.
3. M. Moehring, C. Lessig, and O. Bimber, "Optical tracking and video see-through AR on consumer cell phones," *Proc. Workshop on Virtual and Augmented Reality of the GI-Fachgruppe AR/VR*, pp. 193–204, 2004.
4. S. A. Drab and N. M. Artner, "Motion detection as interaction technique for games & applications on mobile devices," in *Proc. Pervasive Mobile Interaction Devices (PERMID) Workshop at PERVASIVE*, (Munich, Germany), May 2005.
5. S. Bucolo, M. Billingham, and D. Sickinger, "Mobile maze: a comparison of camera based mobile game human interfaces," in *Proc. 7th International Conference on Human Computer Interaction with Mobile Devices & Services (MobileHCI)*, pp. 329–330, (Salzburg, Austria), 2005.
6. D. Wagner and D. Schmalstieg, "ARToolkit on the PocketPC platform," in *Proc. 2nd IEEE International Augmented Reality Toolkit Workshop (ART032)*, pp. 14–15, (Tokyo, Japan), October 2003.
7. H. Kato and M. Billingham, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," in *Proc. 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR)*, pp. 85–94, (San Francisco, CA, USA), 1999.