Design and Implementation of an Immersive Virtual Reality System based on a Smartphone Platform

Anthony Steed[*], Simon Julier[†]
Department of Computer Science, University College London

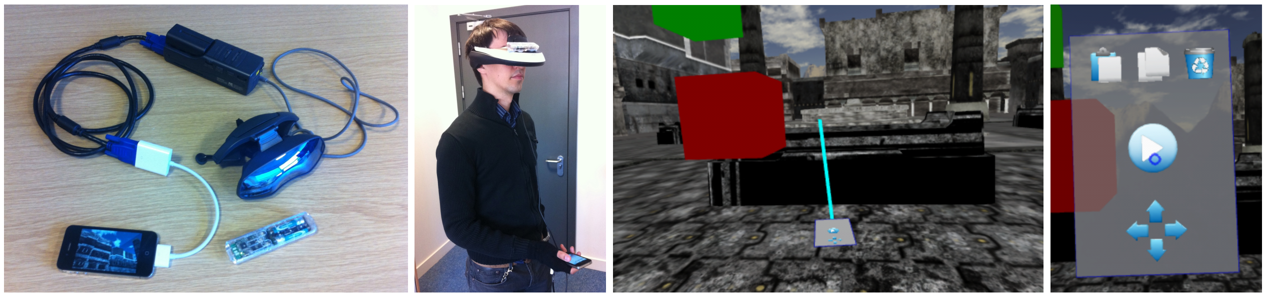


Figure 1: Head-mounted virtual reality system built on a smartphone platform. Left: equipment for a fully mobile configuration. Middle Left: User wearing a tethered version of the display. Middle Right: An example screenshot from our system. Right: Virtual panel showing controls for interacting with objects.

Abstract

With the increasing power of mobile CPUs and GPUs, it is becoming tractable to integrate all the components of an interactive, immersive virtual reality system onto a small mobile device. We present a demonstration of a head-mounted display system integrated onto an iPhone-based platform. In building this demonstration we tackled two main problems. First, how to integrate the user-interface, utilizing the phone itself as an unseen touch interface. Second, how to integrate multiple inertial measuring units to facilitate user interaction. The resulting system indicates the practicality of mobile virtual reality systems based on smartphones.

Keywords: Mobile virtual reality, head-mounted display, 3D user interaction, selection tasks.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information System—Artificial, augmented and virtual realities

1 Introduction

The very rapid increase in the power of mobile CPUs and GPUs has meant that while until recently, immersive virtual reality (VR) systems were powered by dedicated rendering computers, it is now feasible to ask whether we can integrate such systems on mobile devices. Mobility of a VR system has some attractive advantages, such as no tethering to base stations and easy deployment in novel situations. If the VR systems can be built with consumer components, then it reduces the need for specialist hardware.

In this paper, we present an interactive VR system integrated on to a modern smartphone, an iPhone 4S. Such devices come with many of the components that are required for an immersive VR: a mobile GPU capable of rendering high definition imagery, external video and audio output and an inertial measuring unit (IMU) for estimating pose. We needed to add another tracker and a head-mounted display system; both were readily available components.

[*]e-mail: A.Steed@ucl.ac.uk
[†]email: S.Julier@ucl.ac.uk

Our prototype makes two main technical contributions. First, because we are driving an external head-mounted display (HMD), the smartphone itself can also act as an interaction device. We can thus use it as a hand-held (but unseen) controller that can be used to effect interaction within the virtual environment. Second, because the smartphone is hand-held, the HMD is tracked by an external sensor, which is also an IMU. A pair of IMUs will not be registered into the same coordinate frame without some external reference. We take advantage of the fact that both devices can be independently registered against the local gravity vector. Through two simple interaction techniques, raising the hand to the face and a clutch, we can allow the user to realign the two IMUs.

2 Background

The ability to use a computer that is untethered to power and services has many potential applications. While wearable computers have existed for several decades (for example, Mann's WearComp devices [9]), it is in the mid-1990s that the ability to drive HMDs from mobile computers received broader attention. A seminal system is the Touring Machine [5]. This augmented reality system combined a backpack-mounted computer and HMD with a hand-held display and stylus. The HMD was a see-through type. The display could show labels over buildings based on interaction of the user with the untracked handheld display.

The desire to run a full VR system on a completely mobile platform is partly driven by the desire to avoid the restriction of tethering the user to fixed equipment. The Virtsim system supports a team of users each with a backpack and HMD [10]. However, it uses a motion capture system for tracking and so users are constrained to operate within specially instrumented environments. An alternative is to develop a system which does not use any fixed infrastructure. Hodgson presented a fully integrated mobile VR system using a backpack computer [8]. It supported wide area tracking based on GPS positioning. To our knowledge the system did not provide user interaction through a tracked hand-held device.

The continued development of mobile phone platforms now means that devices such as the iPhone 4S have graphics capabilities that surpass desktop systems from only a few years ago. Projects such as the FOV2GO project from the MxR lab at University of Southern California [1][11], demonstrate their potential for use in VR systems. Recently Basu et al. [3], have demonstrated a mobile

VR system using a phone as a component. Compared to their work, the hardware of our system is simpler and we exploit the phone itself as an interaction device and control panel.

In related work, a number of demonstrations have been made of using a mobile device as an interaction device. For example, interacting with a remote display [4].

3 Platform Overview

3.1 Hardware

Our mobile VR system is based on an Apple iPhone 4S. The choice of this rather than an Android-based platform was governed by the availability of specific peripherals for the iPhone at the time of construction. In particular, we required VGA and HMDI video dongles and support for continuous full-screen output. We have demonstrated support for two HMDs. The first is a Sony Glasstron LDI-D100BE (stereo), with 800×600 full pixel resolution in each eye and a horizontal field of view of approximately 28°. This HMD is designed to be power by battery or mains. We use a standard iPhone VGA adapter. This configuration is fully portable. The complete equipment for this setup is shown in Fig. 1, Left. The block at the top middle of this picture is the HMD control box. The second HMD is a Sony HMZ–T1. This is a stereo display with 1280×720 full pixels in each eye and a horizontal field of view of 45°. It is designed to be mains powered, though a portable version could be constructed. We use a standard iPhone HDMI adapter. We were not able to drive either display in stereo, though it should be possible with the Sony HMZ–T1. To track the HMDs, we used the commercially available Hillcrest Labs, Freespace Reference Kit, FSRK–BT–1.

3.2 Software

The software was written in a mixture of Objective-C and C++ for iOS 5.0.1 using XCode 4.3.1 and iOS SDK 5.1. We used the ofxAssimpModelLoader module from OpenFrameworks version 007[1] for 3D model loading. Other rendering software was written in OpenGL ES2.0.

The Freespace FSRK–BT–1 communicates using BlueTooth. We investigated several other options for integrating an external tracker on to the iPhone, including making our own break-out cables, modem-based communication through the microphone, wireless networks, etc. However all presented problems with the closed nature of the hardware and software of the iPhone platform. BlueTooth was chosen because various BlueTooth connections had been demonstrated on jailbroken phones for devices such as wireless keyboards and mice. In particular, an open source, portable user-space Bluetooth Stack, btstack, was available [2]. With the aid of the Hillcrest Lab documentation on the device's BlueTooth packets we were able to write a BlueTooth driver that would read the Freespace device [7]. Thus to make this prototype the iPhone, running iOS 5.0.1, needed to be jailbroken. This may not be necessary in future versions of iOS that may allow users more freedom in selecting external peripherals or if a 3D tracker peripheral manufacturer is certified by Apple for use with iOS.

4 Smartphone as Unseen Touch Panel

4.1 Input

The overall patten of data flow within the software is shown in Fig. 2, Left. Recall that the iPhone acts as the hand tracker, and the Freespace device as the head tracker. Thus the following raw information is available:

- Accelerometer data from iPhone: 3 degrees of freedom (each returned as a 32bit float, meters / second^2)
- Gyroscope data from the iPhone: 3 degrees of freedom (each returned as a 32bit float, radians / second)
- Estimated orientation data from the iPhone: 3 degrees of freedom (each returned as a 32bit float)
- Accelerometer data from the Freespace: 3 degrees of freedom (each returned as 16 bit integer, millimeters / second^2)
- Gyroscope data from the Freespace: 3 degrees of freedom (each returned as 16 bit integer, milliradians / second)
- Touch input from the iPhone screen: multiple times 2 degrees of freedom (each returned as 32bit float value, in range 0–320 by 0–480)

From these raw data we can exploit various types of interaction. First, we can use the head tracking data to create a coordinate frame for the head, and then we can track the head gaze direction of the user. Second, we can use the hand tracking data to create a coordinate frame for the hand, and track the hand pointing direction. We can orient both of these coordinate frame in to the same sense (Y is up, X to the right, -Z in to the screen or away from the user and roll is rotation around Z, pitch around X, yaw around Y, see Fig. 2, Right). The hand tracker already estimates an absolute rotation through the iOS CoreMotion API.

4.2 Processing

4.2.1 Orientation Fusion

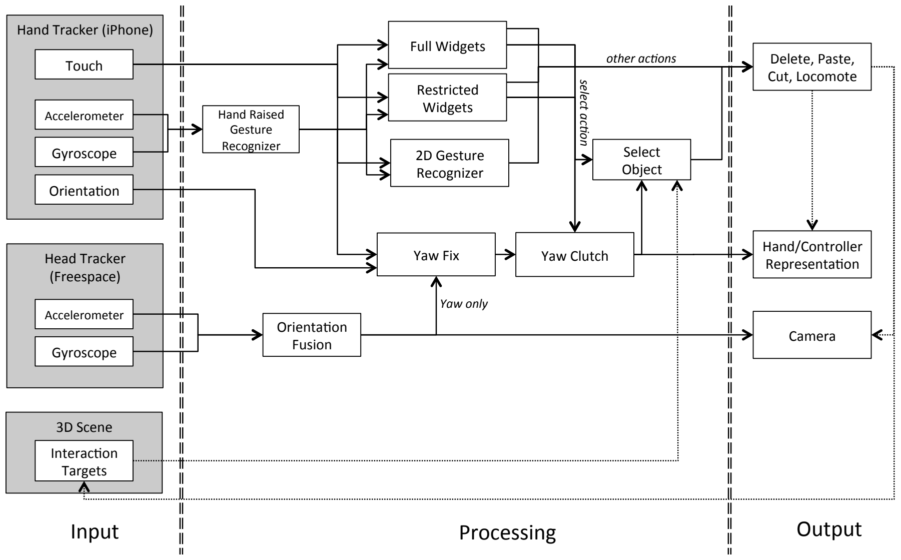
The problem of estimating 3D attitude from a self-contained inertial measurement system is a well-known and widely-studied problem [6, 2, 12]. In most systems, attitude is computed by integrating rate information from gyroscopes over time. However, because these measurements are corrupted by noise and the fact that the sampling rate is finite, this estimate will drift over time. To counteract this drift, supplementary information from other sensors can be used. The InterSense [6] and XSens [12] devices, for example, exploit accelerometer information to measure the gravity vector when the platform is stationary We use the same integration technique as the Intersense and XSens devices. This module has as input the accelerometer and gyroscope data from the Freespace devices, and outputs an orientation for the head tracker.

Given that we have two IMUs they will diverge over time because they have slightly different sampling errors and noise in their measurements. Fortunately the two IMUs we use can both be corrected against the up (gravity) vector. This means that they diverge in yaw (rotation about up) only. To assess the drift which can be caused by movement, we conducted an experiment in which the Freespace was attached to the iPhone, and the two underwent a series of rapid motions. The results are illustrated in Fig. 3, which shows a 45 second log of the yaw values of the head tracker rigidly attached to the hand tracker. Both are moved together, and returned to the starting orientation twice (periods 23–25 seconds and 38–42 seconds). In the second stationary period the difference in yaw value is 18°s, and the two devices have drifted in opposite directions (head +7°, hand -11°). The devices show some drift while stationary. In the period 38–42 seconds, the devices drift slowly: 7.3–7.9° for the head tracker (which is consistent with the predicted steady state covariance of the bias of the gyros), -10.1° – -11.5° for the hand tracker. This may not be a problem for small values of drift, but over time, the offsets could make the system unusable. We thus need a method to reset the relative heading between the head and hand trackers. Note that in this situation, we do not know if the iPhone would calibrate against the magnetometer, but in any case the iPhone yaw estimate has drifted from the known start angle. Thus on this timescale we should expect noticeable yaw drift.

This divergence cannot be controlled as there is no external reference to make a measurement to, and no known relationship between the two head and hand device. Thus the user interface must take into account the divergence between the two yaw values.

[1]http://www.openframeworks.cc/

[2]http://code. google.com/p/btstack/



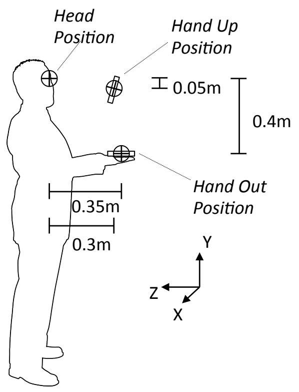


Figure 2: Left: Overall pattern of data flow within the system. On the left we show sources of information for each processing frame. In the middle are the operations on this data. These are explained in the text. On the right are the potential outcomes and scene changes that occur. Solid lines represent transfer of 2D, 3D or flag data. Dotted lines represent interaction with 3D scene objects, including getting geometry or getting or setting transformations. Right: The coordinate frames used to define the user interfaces.

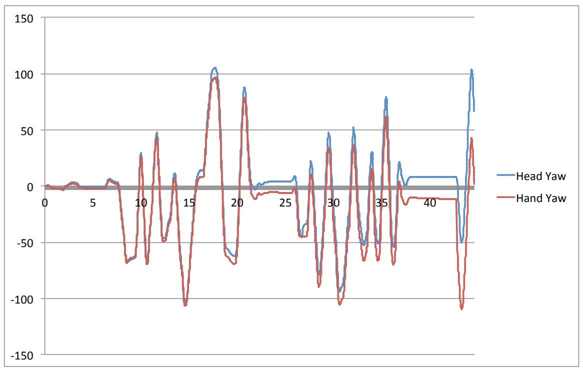


Figure 3: Drift in yaw values for head and hand trackers over a 45 second period.

4.2.2 Virtual Screen

An early design decision was to incorporate some form of virtual phone-like screen in the environment to explore the use of 2D widgets. Such a virtual device could be extended to a full 3D editing system or a complex simulation control (e.g. see [13]). For this demonstration we implemented a few simple editing controls (cut, paste, delete), a selection button and a joystick, see Fig. 1, Right.

In order to make this UI legible, and to accommodate future functionality, it was decided to render it relatively large on the screen, as shown in Fig. 1, Right. Because we do not have an absolute position for the hand, nor a relative distance between the head and hand, the user cannot bring the UI to this position themselves. Thus we utilised a gesture to bring the user interface to this position, see the following section for a discussion of implementation. When the user made this *Hand Raised Gesture* (see Fig. 2, Left), the user interface would be activated (*Full Widgets UI* in Fig. 2. Left) and the hand position and thus the visual position from which the controller is drawn was set to *Hand Up Position*, see Fig. 2, Right.

In practice, when trying to point at an object, the need to use buttons was found to be a little bit difficult. This was because some users would need to shift the iPhone in their hand, or bring their other hand over. Thus the current prototype has a second mode *Limited Widgets UI* where only the select button and joystick buttons are activated and 2D gestures are used. In this case the UI is drawn much smaller, though the buttons activating are still visible if the hand is in view. This smaller UI is shown in Fig. 1, Middle Right. The extra space on the UI is used as the gesture recognition region as discussed below. To facilitate pointing at objects, the hand position is moved to *Hand Out Position*, see Fig. 2, Right.

4.2.3 2D and 3D Gesture

We investigated making 3D gestures by waving the device in 3D paths, but found that this was cumbersome and relatively inaccurate. We also investigated simple 3D gestures combined with 2D gestures. For example picking an object by pulling with the device and pulling with the finger simultaneously. Such gestures were easy to learn and perform but the 3D gesture was considered unnecessary as both the gestures mimicked the same action. The 2D gesture was preferred by users because it was much more reliably recognised somce it had a defined start and end (on press and release of the touch screen). We do have one 3D gesture, the Hand Raised Gesture that is explained below.

The 2D gesture recognition (*2D Gesture Recognizer* in Fig. 2 Left), is based on the shape of the path. We use the “$1 recognizer” [14]. We defined a gesture input region on the top half of the touch screen, above the select button and joystick. We used four gestures for the actions to cut, past and delete objects, and to return to a home position. Users found this region easy to reach.

One 3D gesture was retained: the Hand Raised Gesture (*Hand Raised Gesture Recognizer* in Fig. 2, Left). This is recognised by the pitch of the hand rising above 60° along with a 0.5s vertical motion. The gesture remains active until the hand tips below 50° vertically, in which case the gesture is terminated. The Hand Raised Gesture is used to toggle between Full Widgets UI and Limited

Widgets UI. It is also used to reorient the hand, see below.

4.2.4 Yaw Fix and Yaw Clutch

As noted in Section 4.2.1 the yaw of the head and hand will diverge over time. We provide two mechanisms to re-align the head and hand. The first is provided by the *Yaw Fix* module. Whilst the Hand Raised Gesture is active, the yaw of the head is used to overwrite the yaw of the hand. This has the effect of aligning the Full Widgets UI in front of the head. As the hand is lowered, an offset is calculated between the current hand and head yaw values, and this offset is then applied to the received yaw values from the hand tracker. This has the effect of realigning the hand yaw to the head yaw as the hand is lowered. We have found this to be intuitive for users.

The second mechanism is provided by the *Yaw Clutch*. This is triggered by the select action as described above. When the select action is activated, the yaw of the hand is frozen, by calculating an offset that keeps the hand yaw constant as the hand tracker value continues to vary. This offset is then held constant as select action is deactivated. This acts as a clutch: hold the button to freeze the hand controller and reorient your hand. Users also found this intuitive to do.

4.2.5 Object Selection

In the Limited Widgets UI version, a blue ray is drawn along the pointing direction of the handheld control (-Z), see Fig. 1, Middle Right. When the select button is pressed a ray is cast along this ray to select an object. This uses a single ray cast and thus allows the precise selection of small targets. Because the tracking is stable, we have not seen a need for the user to press select and then "hunt" for the target on release. We have found that the Yaw Clutch and select action can be activated by the same button: a select is a click, whereas a hold is the clutch. Users pick up the dual use quickly and have not reported this to be a problem.

4.3 Output

One module performs edits (cut, paste, delete) on the scene. Locomotion about the environment is effected using the virtual joystick that appears on both the Full Widgets UI and Limited Widgets UI. Because we are using the head direction as the principle direction, and correcting the hand orientation to that as described above, we use the locomotion direction based on travelling forwards and backwards in the direction of gaze. The vertical distance from the initial press event on the virtual joystick is used as velocity of travel. There is a dead zone, and a maximum distance: the range (10,50) pixels distance is mapped to (0, 2) m/s. The horizontal distance is similarly mapped: (10,50) to (0, 90) degree/s rotation in yaw around the head position.

5 Conclusion

In this technote, we have described the development of a HMD-based VR system that is integrated onto an iPhone-based platform. The design of the system is novel in that it exploits the iPhone itself as an unseen touch controller. The main difficulty in implementation was the lack of registration between the two IMUs: one in the iPhone and a separate one on the head. Given that there we no external reference signals to utilize, the user interface had to be adapted as discrepancies in yaw between the two sensor could rapidly grow to the point where pointing behaviour is unacceptably degraded. To overcome these limitations, we introduced two mechanisms: a gesture to automatically realign the coordinate systems crudely, and a clutch to manually realign them precisely. The system can operate at 60Hz for worlds with a few thousand polygons. Latency is acceptable at approximately 100ms.

We have evaluated this system informally with various demonstrations and also in a pilot trial that demonstrated that the clutch technique was easy to learn and useful to the users. Overall users found the system easy to use. None had problems navigating about the environment and interacting with objects.

There are many potential future avenues of work. Aside from integrating the constantly improving hardware, we believe that one of the most fruitful areas of further consideration will be to explore other characteristics of behaviours that can be used to effect registration between different sensors.

Finally, we hope that the availability of an immersive VR system that is based on a widely available consumer platform and other readily available components, may enable new applications of VR in domains where installation and maintenance costs might otherwise be an issue.

Acknowledgements

This work was funded through the European Commission for the European Union project BEAMING (FP7-248620).

References

[1] Fov2go project. http://projects.ict.usc.edu/mxr/diy/fov2go/, Dec. 2012.

[2] R. Azuma. *Predictive Tracking for Augmented Reality*. PhD thesis, UNC Chapel Hill, 1995.

[3] A. Basu, C. Saupe, E. Refour, A. Raij, and K. Johnsen. Immersive 3dui on one dollar a day. In *3D User Interfaces (3DUI), 2012 IEEE Symposium on*, pages 97 –100. IEEE Computer Society, 2012.

[4] R. Dachselt and R. Buchholz. Natural throw and tilt interaction between mobile phones and distant displays. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 3253–3258. ACM, 2009.

[5] S. Feiner, B. MacIntyre, T. Hollerer, and A. Webster. A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. In *Proceedings of the 1st IEEE International Symposium on Wearable Computers*, ISWC '97, pages 74–81. IEEE Computer Society, 1997.

[6] E. Foxlin. Inertial Head-Tracker Sensor Fusion by a Complementary Separate-Bias Kalman Filter. In *Proceedings of IEEE Virtual Reality Annual Symposium VRAIS '96*, pages 185–194. IEEE Computer Society, 1996.

[7] Hillcrest Labs. Freespace hid report definitions. http://libfreespace.hillcrestlabs.com/libfreespace/1000-2045-HIDReportDefinitionsv1.0.pdf, Dec. 2012.

[8] E. Hodgson, E. Bachmann, D. Waller, A. Bair, and A. Oberlin. Virtual reality in the wild: A self-contained and wearable simulation system. *Proceedings of IEEE Virtual Reality Conference*, pages 157–158, 2012.

[9] S. Mann. An historical account of the 'wearcomp' and 'wearcam' inventions developed for applications in 'personal imaging'. In *Proceedings of the 1st IEEE International Symposium on Wearable Computers*, pages 66–73. IEEE Computer Society, 1997.

[10] Motion Reality Inc. Virtsim. http://www.motionrealityinc.com/, Dec. 2012.

[11] L. Olson, D. M. Krum, E. Suma, and M. Bolas. A design for a smartphone-based head mounted display. In *IEEE Virtual Reality*, pages 233–234, Singapore, Mar. 2011. IEEE.

[12] D. Roetenberg, H. Luinge, and P. Veltink. Inertial and magnetic sensing of human movement near ferromagnetic materials. In *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 268–. IEEE Computer Society, 2003.

[13] Z. Szalavári and M. Gervautz. The personal interaction panel: A two-handed interface for augmented reality. *Computer Graphics Forum*, 16(3):335–346, 1997.

[14] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 159–168. ACM, 2007.