Proceedings of the 31st ACM Symposium on Applied Computing (SAC 2016), Pisa, Italy, April 4-8, 2016.

An Android Application for Head Tracking

Massimiliano Benedetto
Univeristy of Pisa
Pisa, Italy
massimiliano.benedetto.22@gmail.com

Alessio Gagliardi
Univeristy of Pisa
Pisa, Italy
alegaglia@hotmail.it

Pasquale Buonocunto
Scuola Superiore Sant'Anna
Pisa, Italy
p.buonocunto@sssup.it

Giorgio Buttazzo
Scuola Superiore Sant'Anna
Pisa, Italy
g.buttazzo@sssup.it

ABSTRACT

This paper presents a head-tracking system based on an inertial sensor that sends rotation data to an Android application in charge of recording and visualizing data in real time. Data are sent via wireless channel using Bluetooth Low Energy communication and are processed to provide a realistic real-time 3D animation of the user head.

CCS Concepts

•**Human-centered computing** → *Gestural input;* Pointing devices; Pointing; •**Applied computing** → *Consumer health;*

Keywords

Head Tracking, IMU, Real-time operating system, Multitasking application

1. INTRODUCTION

Inertial measurements units (IMUs) represents an effective solution for tracking human joint trajectories in a compact and portable system. Several commercial tracking systems are available today, such as the Moven motion capture suite by Xsens [18], or a similar product by Intersense [8]. The main disadvantages of these solutions are their cost (in the range of 10K euros and beyond), and the use of proprietary wireless communication protocols that requires special hardware.

Several IMU-based tracking systems have been proposed in the literature for different purposes. For instance, Foxlin [5] presented a tracking system to measure the head movements relative to a moving simulator platform using differential inertial sensors. Avizzano et al. [2] developed a head tracking device based on accelerometers to provide a user interface for navigation in virtual environments and remote control. Keir et al. [9] proposed a head tracking device for robotics applications based on accelerometers. All these devices however, were developed to be integrated in a larger system and not to work as a stand-alone device for personal usage.

Other head tracking devices were proposed using different technologies. For example, Mulder, Jansen, and van Rhijn [14] developed an optical head tracking system, using two fixed FireWire iBOT cameras that recognize a dot pattern mounted onto shutter glasses. A similar system was developed by Lee [11] using a Wii Remote device, a Bluetooth receiver and IR LEDs. Wii Remote is a motion controller using an IR filtered camera to track a number of IR LEDs placed on the user's head. Although the cost of these systems is more affordable, they are not portable and suffers from occlusion problems.

Other approaches developed for virtual reality and gaming are represented by the Google Cardboard [7], the OCULUS GEAR VR [13], or the Zeiss VR One [17], which exploit the motion sensors and the display of an Android smartphone mounted as mask visor. Such solutions, however, are quite invasive for the average user, since require wearing a visor that prevents the view of the real world.

A body motion tracker for Android platforms having characteristics similar to the system presented in this paper is the Run-n-Read developed by Weartrons Labs [10]. The system, however, is specifically designed to support text reading on tablets or Ebook readers during dynamic activities and works by moving the image on the screen in sync with the motion detected by the sensor.

In summary, all the solutions existing today are either too expensive, too complex to be used, not portable, or not easily connectable with mobile devices. To fill this gap, the system presented in this paper has been specifically designed for personal purposes (sport, tele-rehabilitation, training, or gaming), using low-cost components, and associated with an Android application developed for non expert users.

This paper presents an Android application for storing and visualizing the user head movements using data coming from an inertial sensor mounted on the user head. A major advantage of the proposed system is the easy calibration phase, which enables the system to be used by non expert people for a range of different applications domains.

SAC 2016, April 4-8, 2016, Pisa, Italy
©2016 ACM. ISBN 978-1-4503-3739-7/16/04...$15.00
DOI: http://dx.doi.org/10.1145/2851613.2851930

2. SYSTEM DESCRIPTION

The head tracking system consists of a head-mounted inertial sensor communicating via wireless channel to a smartphone running the Android operating system [1].

One of the most interesting features of this system is that the application does not require the user to mount the sensor in a very precise position on the head. This is achieved through a self-calibration routine executed at system initialization that allows the application to automatically compensate for different initial sensor positions.

More specifically, when the mobile device establishes the connection with the sensor, the first received quaternion is stored by the application as a *reference quaternion*, representing the zero position of the human head. From this point on, all head rotations are computed as a difference between the current quaternion (received in real-time from the sensor) and the initial reference quaternion. Such a simple feature acts as a very effective self-calibration method that solves all the problems resulting from the irregular morphology of the head and relieves the user from being too precise in positioning the sensor on its head. Thanks to this method, the system can easily be worn by anyone by hiding it inside a cap or fixing it at an elastic band around the head.

2.1 The inertial sensor

Head movements are detected through a wireless inertial sensor developed at the RETIS Lab of the Scuola Superiore Sant'Anna of Pisa for limb tracking in telerehabilitation systems [4]. The sensor was specifically designed to reduce power consumption with respect to similar commercial devices, balancing lifetime with dimensions and incorporating the state-of-the-art IMU device to provide accurate orientation data.

The device dimensions are $4*3*0.8$ cm and it weights about 30 g. The sensor integrates a Nordic nRF51822 microcontroller with a 2.4 Ghz embedded transceiver [15], an InvenSense MPU-9150 9-axis inertial measurement unit (IMU), an integrated onboard chip-antenna, a USB port, a microSD card for data logging, some I/O devices (3 LEDs, 2 buttons and a buzzer), and six configurable GPIO pins that can be used for digital I/O or analog ADC inputs for expanding the board with other sensors. Figure 1 illustrates the block diagram of the main logical components of the sensor node.

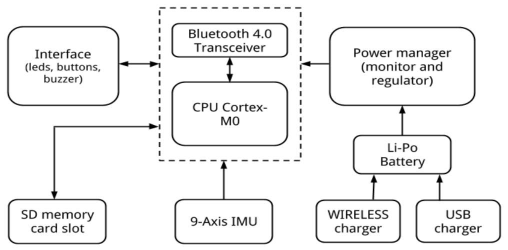
Interface (leds, buttons, buzzer)
Bluetooth 4.0 Transceiver
CPU Cortex-M0
Power manager (monitor and regulator)
Li-Po Battery
SD memory card slot
9-Axis IMU
WIRELESS charger
USB charger

Figure 1: Block diagram of the sensor components.

The node is powered by a single cell LiPo battery that guarantees more than 20 hours of continuous usage at the maximum frequency (100 Hz) and can be charged via USB or through a wireless recharge device. The device communicates with the table through the Bluetooth 4.0 Low Energy (BLE) protocol [6] intended for strongly energy-constrained applications, such as sensors or disposable devices. Although the BLE data transmission is performed at a reduced speed, compared to the standard Bluetooth [3] or to WiFi communication, the available bandwidth (approximately 1 Mbit/s) resulted to be appropriate for the implemented head tracking system, allowing us to exploit the other advantages of the BLE protocol, as lower power consumption, better immunity to interference, and compatibility with widespread mobile devices.

3. 3D MODEL

To recreate the natural movement of the head, a 3D model was imported in Blender to provide the definition of virtual bones, used as a virtual skeleton for expressing rotations around their joints. Each bone affects a specific portion of the mesh, so that its movements will change the geometry of the corresponding area of influence. A careful positioning and dimensioning of the armature is crucial to achieve a realistic head animation, since small position errors of the armature inside the mesh may cause incorrect rotations of the whole model.

The skeleton structure used in this application consists of three bones, one for the bust (*RootBone*), one for the neck (*NeckBone*), and one for the head (*HeadBone*). The rotation detected by the head-mounted sensor is associated with the *HeadBone* segment. The kinematic transformation performed by the Android application on the 3D model is then triggered by the reception of the quaternion, periodically transmitted by the IMU every 20 milliseconds.

4. APPLICATION STRUCTURE

The application has been developed using the LibGDX library [12], which is an open source Java framework for game development. The main advantage if this library is that it provides APIs for multiple platforms, such as Linux, Mac, Windows, Android, iOS, BlackBerry, and HTML5. This is possible thanks to the presence of multiple project folders in its layout. In particular,

- Core project: it contains all the application code, except the so called starter classes. All the other projects link to this project.
- Android project: it contains the starter class and other necessary files to run the application on Android. The assets/ folder stores the assets of the application for all platforms.
- Desktop project: it contains the starter class to run the application on the desktop. It links to the Android project's assets/ folder and to the core project.
- HTML5 project: it contains the starter class and other necessary files to run the application as a native HTML5 application. It links to the Android project's assets/ folder (see gwt.xml file) and to the core project.
- iOS RoboVM project: it contains the starter classes and other necessary files to run the application on iOS through RoboVM. It links to the Android project's assets folder (see robovm.xml) and to the core project.

For the head tracking application, only the Core and Android project folders have been used. The Android project contains the functions responsible for the Bluetooth communication and the user interface, whereas the ”Core project” contains the functions responsible for rendering the model and change its parameters. The data received from the sensor are sent to the classes of the Android project and are processed by the classes of the Core project. For this reason, both folders contain specific code for exchanging data between them. The structure of the application is shown in Figure 2.

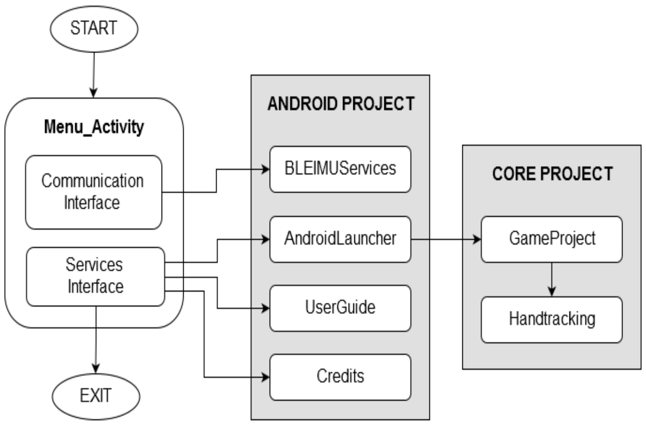

Figure 2: Application Structure.

The main menu of the application provides a *Communication Interface* responsible for managing the Bluetooth protocol (BLEIMUServices) and a *Services Interface* for selecting of the offered services.

4.1 Communication interface

Using the Communication Interface the user can start or stop the Bluetooth communication between the sensor and the application, set the baud rate, scan the available devices and even monitor the arrival of incoming packets. The *StartService* and *StopService* buttons are used to start or stop the communication service, respectively. The *setHz* button allows setting the transmission frequency, whereas *ScanDevices* is used to start searching for available sensors. The *Connect* and *Disconnect* buttons allow starting and stopping the connection and, finally, the *ON/OFF* button can start/stop data acquisition from the connected devices.

In particular, when the ON button is pressed, the first received quaternion is used as a reference for all the subsequent data coming from the sensor, in the sense that all head rotation angles are computed as a difference between the current quaternion and the initial reference. In this way, the preliminary phase of positioning the sensor on the user head is not crucial, since any initial misalignment with respect to the ideal fixed frame is compensated by the differential computation with the initial quaternion. Such a calibration phase can be triggered by the user at any time by interacting with the control panel.

4.2 Services interface

The Services Interface allows the user to start other activities, such as testing the system, checking for the credentials, accessing the app manual, and turning the system off by pressing the *Exit* button.

Pressing the *Test* button the system displays a menu that offers two different operational modes. The *Sensor Mode* provides a real-time animation of the 3D model, whose angles are updated every 20 milliseconds based on the data coming from the sensor. The *Manual Mode* displays six screen buttons that can be used to manually change the pitch, roll, and yaw angles of the head, individually. The *Reset* button allows setting the head in the starting position. In both modes, orientation and zoom of the avatar can be controlled by the user via touch screen during the real-time animation.

Finally, pressing the *Instructions* button the system displays the user manual, which reports the application features and explains the user interface.

5. EXPERIMENTAL RESULTS

This section presents some experimental results carried out to evaluate the effectiveness of the developed system in measuring head orientations and providing timely data.

5.1 Orientation accuracy

The orientation errors of the sensor node was evaluated with respect to a reference given by a Polhemus Patriot system [16], which provides the position and orientation of a mobile probe with respect to a fixed reference. In particular, the fixed reference emits a tuned electromagnetic field that is measured by the mobile probe. This procedure avoids performing hybrid data merging via software. The resulting resolution is about 0.03 mm and 0.01 degrees, whereas the static accuracy is of 1.5 mm RMS for the X, Y, and Z position and 0.4 degrees RMS for the orientation.

The orientation accuracy of the sensor node was measured by converting quaternions to Euler angles and directly comparing them to the ones given by the Polhemus. Only the angles around X and Y axes were measured, because rotations around the Z-axis relies on the magnetometer, which is strongly affected by the magnetic field generated by the Polhemus.

Figure 3 shows the IMU measurement error distribution with respect to the Polhemus reference. Note that the error mean is 0.8685 degrees and its RMS is 0.9871 degrees. Considering the lower cost of the head sensor with respect to the Polhemus (i.e., tens of euros compared to thousands of euros) the achieved results are quite satisfactory.

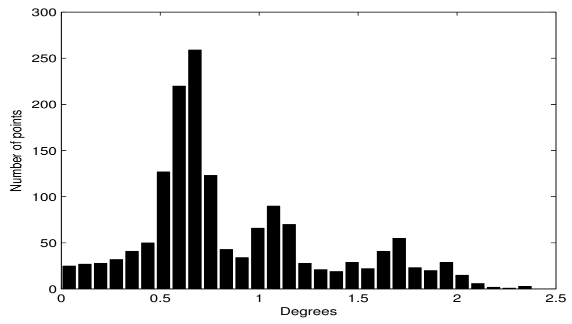

Figure 3: Distribution of the angular error.

5.2 Processing time and communication delay

Processing times on the sensor node were measured by switching a GPIO pin at the beginning and at the end of the computation, and capturing them by a multi-channel logic analyzer. The results of this experiment are shown in Table 1, which reports the execution time statistics for the I^{2}C read routine and the CPU processing time, that is, the time measured from the instant the packet is read from the I^{2}C bus to the time it is sent to the radio transceiver.

Time (ms)	Mean	RMS	Std Dev.	Variance
I2Cread	2.409	2.416	0.1833	0.0335
computation	0.218	0.228	0.0660	0.0043

Table 1: Execution times (ms).

Message delays were estimated by measuring the interval of time from the instant at which the data packet is transmitted by the sensor to the time at which the Android application reacts to it by changing the color of a rectangular area of the screen. The color change (from black to white and viceversa) has been measured by a photoresistor attached to the screen. With the setup described above, data have been acquired at a sampling frequency of 20 Hz by a Nexus 7 tablet for 60 s.

The results of this test are shown in Figure 4, which reports the measured delay distribution. Note that the end-to-end delay of data processing and representation varies from 3 ms to 30 ms, with an average value of 14.85 ms, leading to a smooth and reactive the 3D head animation.

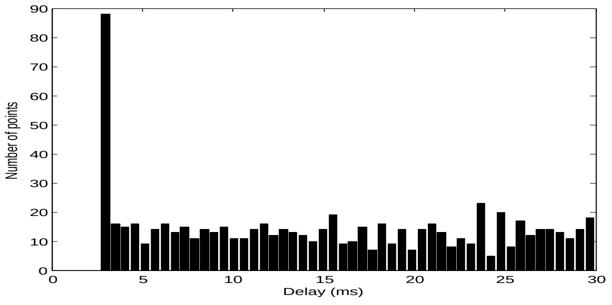

Figure 4: End-to-end delay up to the Android app.

Table 2 reports the delay statistics of radio transmitting times and the overall delay from the time the packet is sent by the sensor to the time it is processed and displayed on the screen by the Android application.

Delay(ms)	Mean	RMS	Std Dev.	Variance
BLEsend	2.627	2.634	0.197	0.039
display	14.8523	17.1974	8.6754	75.2621

Table 2: Delay times (ms).

The high variance observed in the end-to-end delay measurements is due to the transmission variability of the BLE protocol, the lack in the Android framework of a structured support for managing time constraints, and the complexity of the video rendering subsystem, which is optimized to improve the average user experience rather than a predictable timing behavior of data representation.

6. CONCLUSIONS

This paper presented a low-cost, portable head-tracking device that works in combination with an Android application in charge of recording and visualizing head rotation data in real time. Sensory data produced by a 9-axis inertial unit are sent to a smartphone using the BLE communication protocol and then reconstructed to produce a realistic real-time 3D animation of the user head. The tests performed on the device to measure the end-to-end delay show that all the sensory data packets are processed and displayed within 30 ms, with an average delay of 14 ms, making the proposed system very competitive with respect to other (more expensive) commercial solutions, and attractive for many applications, including rehabilitation, sport, gaming, and virtual reality.

7. REFERENCES

[1] Android. Android operating system, 2014.

[2] C. Avizzano, P. Sorace, D. Checcacci, and M. Bergamasco. A navigation interface based on head tracking by accelerometers. In *Proc. of the 13th IEEE Int. Workshop on Robot and Human Interactive Communication (ROMAN 2004)*, Kurashiki, Japan, September 20-22, 2004.

[3] Bluetooth. The low energy tecnology behind bluetooth smart, 2015.

[4] P. Buonocunto and M. Marinoni. Tracking limbs motion using a wireless network of inertial measurement units. In *Proc. of the 9th IEEE Int. Symp. on Industrial Embedded Systems (SIES 2014)*, Pisa, Italy, June 18-20, 2014.

[5] E. Foxlin. Head-tracking relative to a moving vehicle or simulator platform using differential inertial sensors. In *Proceedings of the AeroSense Symposium on Helmet and Head-Mounted Displays V, SPIE Vol. 4021*, Orlando, FL, April 24-25, 2000.

[6] Google. Bluetooth low energy, 2004.

[7] Google. Google cardboard, 2014.

[8] InterSense Inc. *InertiaCube2+ Manual*, 2008.

[9] M. Keir, C. Hann, G. Chase, and X. Chen. Accelerometer-based Orientation Sensing for Head Tracking in AR & Robotics. In *Proc. of the 2nd Int. Conference on Sensing Technology (ICST 2007)*, Palmerston North, New Zealand, Nov. 2007.

[10] W. Labs. Run-n-read, 2013.

[11] J. C. Lee. Head-tracking for desktop vr displays using the wii remote, 2008.

[12] LibGDX. Libgdx, 2013.

[13] O. V. LLC. Oculus gear vr, 2015.

[14] J. D. Mulder, J. Jansen, and A. van Rhijn. An affordable optical head tracking system for desktop vr/ar systems. In *Proc. of the 7th Int. Immersive Projection Technologies Workshop (9th Eurographics Workshop on Virtual Environments)*, 2003.

[15] NORDIC Semiconductor. *nRF51822 Product Spec. v1.3*, May 2013.

[16] Polhemus. *Polhemus Patriot Product Spec.*, Feb. 2010.

[17] Z. VR. Zeiss vr one, 2015.

[18] Xsens Technologies B.V. *MTi and MTx User Manual and Technical Documentation.*, May 2009.