A Real-Time Head-Tracking Android Application Using Inertial Sensors

Massimiliano Benedetto, Alessio Gagliardi
Università di Pisa, Pisa, Italy
{m.benedetto2, a.gagliardi3}@studenti.unipi.it

Pasquale Buonocunto, Giorgio Buttazzo
Scuola Superiore Sant'Anna, Pisa, Italy
{p.buonocunto, g.buttazzo}@sssup.it

***Abstract*—Several systems have been proposed in the literature to monitor head movements for different applications, including health care, tele-rehabilitation, sport, virtual reality, and gaming. Most of these systems are either expensive or not precise, and often require complex setup and calibration procedures. This paper presents a head-tracking monitoring system consisting of a 9-axis inertial unit that sends rotation data to an Android application that is in charge of recording and visualizing data in real time. Sensory data are sent to the smartphone via wireless channel, using the Bluetooth Low Energy communication standard. Data are then processed on the mobile device providing a realistic real-time 3D animation of the user head. Experimental results are presented to evaluate the end-to-end latency of the real-time representation.**

Keywords–Head Tracking; IMU; Real-time operating system; Multitasking application;

I. Introduction

A number of sensors using different technologies can be used today to track joint rotations. They include magnetic systems (like the Polhemus [1]), and visual-based systems, which use several cameras to triangulate the 3D position of a set of points detected, with the aid of markers attached to the body. For instance, the Vicon motion capture system [2] uses retro-reflective markers illuminated by infrared LEDs and tracked by infrared cameras. These systems can reach a precision in the range of millimeters, but are quite expensive, not portable, require a complex calibration procedure, and suffer from problems due to occlusions or missing markers.

A cheaper solution is represented by the Microsoft Kinect [3], which uses a camera and an infrared depth sensor to capture full-body 3D motion under any ambient light conditions. This system reaches a precision in the range of centimeters but, unfortunately, it is not portable and does not support a direct connection with tablets and smartphones.

Inertial measurements units (IMUs) represent 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 [4], or a similar product by Intersense [5]. 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 [6] presented a tracking system to measure the head movements relative to a moving simulator platform using differential inertial sensors. Avizzano et al. [7] developed a head tracking device based on accelerometers to provide a user interface for navigation in virtual environments and remote control. Keir et al. [8] proposed a head tracking device for robotics applications based on accelerometers. However, all these devices were developed to be integrated in a larger system and not to work as a stand-alone device for personal usage.

Zhou and Hu [9] developed a real-time monitoring system for human upper limb movements in post-stroke rehabilitation. This system is based on an Xsens MT9 inertial sensor acquired by a PC. Roetenberg at al. [10] presented an Xsense motion capture suit capable of reconstructing full-body human posture based on biomechanical models and sensor fusion algorithms. Although the system does not suffer from the problems typical of optical systems, wearing a suit is not so practical for monitoring everyday activities.

Other head tracking devices were proposed using different technologies. For example, Mulder, Jansen, and van Rhijn [11] 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 [12] using a Wii Remote device, a Bluetooth receiver and IR LEDs. Wii Remote is a motion controller using an IR 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 [13], the OCULUS GEAR VR [14], or the Zeiss VR One [15], which exploit the motion sensors and the display of an Android smartphone mounted as mask visor. However, such solutions are quite invasive for the average user, since they 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 [16]. 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, 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. A preliminary version of this paper has been presented as a poster [17].

Contribution of the work. The main contribution of this work is the development of a head tracking system integrating an 9-axis inertial sensing node with an Android application for storing and visualizing the head movements of the user in a graphic form. One of the major advantages 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.

Organization of the paper. The remainder of this paper is organized as follows: Section II presents an overview of the system architecture and the general system functionality; Section III describes the kinematic model used for the 3D representation; Section IV illustrates the structure of the Android applications; Section V presents some performance measures carried out to evaluate the end-to-end latency of the representation process; and finally, Section VI states our conclusions and future work.

II. 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 [18].

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 and quickly be worn by anyone by hiding it inside a cap or fixing it at an elastic band around the head, as illustrated in Figure 1.

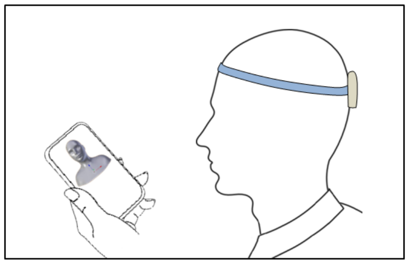


Figure 1. The head tracking system.

The application running on the smartphone has been developed in Java on Eclipse IDE (integrated development environment) using free and open source software components. The Java Development Kit 7 released by Oracle Corporation gives a preliminary class libraries and tools, such as compilers and debuggers, necessary for developing applications [19]. The Android SDK API 18 was used to provide the Bluetooth 4.0 Low Energy (BLE) support [20] for interfacing the smartphone with the inertial sensor.

As far as the graphic is concerned, the 3D design of the head model has been carried out using Blender [21], whereas the real-time manipulation of the model was implemented using Libgdx [22], a game-development application framework which includes some third-party libraries, as OpenGL, for the 3D rending and other functionalities.

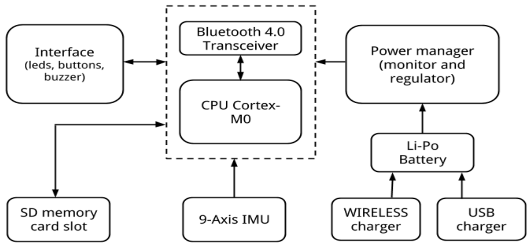
A. The inertial sensor

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 [23]. 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 the following components:

- a Nordic nRF51822 microcontroller with a 2.4 Ghz embedded transceiver [24];
- an InvenSense MPU-9150 9-axis inertial measurement unit (IMU);
- an integrated onboard chip-antenna with 20 m range indoor/80m range outdoor;
- a USB port;
- a microSD card for logging and debugging;
- some I/O devices (3 LEDs, 2 buttons and a buzzer);
- six configurable GPIO pins (that can be used for digital I/O or analog inputs), for expanding the board with other sensors, like ECG, EMG, etc.

Figure 2 illustrates the block diagram of the main logical components of the sensor node, whereas Figures 3(a) and 3(b) show the internal circuitry and the container, respectively.


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 2. Block diagram of the main components of the sensory device.

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 tablet through the BLE protocol [20] intended for strongly energy-constrained applications, such as sensors or disposable devices. Compared to the standard Bluetooth or to WiFi communication, the BLE data transmission is performed

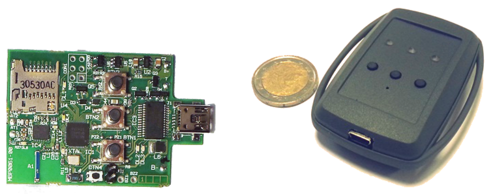

(a) Interior of the node (b) Sensor container

Figure 3. The sensor node.

at a reduced speed [25] (approximately 1 Mbit/s) that was found to be appropriate for the implemented head tracking system, allowing us to exploit the other advantages of the BLE protocol, such as lower power consumption, better immunity to interference, and compatibility with widespread mobile devices.

In summary, the main advantages of this sensor with respect to similar commercial products available on the market are briefly summarized below:

- Low cost (in the range of 50 Euros);
- Precision of joint angle measurements below 1 deg;
- Low-energy consumption (a battery can last for a week in typical conditions and for two days in continuous transmission at the maximum frequency of 100 Hz);
- Standard wireless BLE protocol for connecting to tablet and smartphones;
- light weight (30 g);
- Simple initialization and calibration procedure.

III. 3D MODEL

The 3D model used for the graphics animation of the user's head was built to represent the upper part of a human being, including the head, the neck, the shoulders and a portion of the chest. They are represented by a set of vertices connected with line segments to form a polygonal mesh. Such a 3D model was created in Blender to provide the structure for the animation.

To recreate the natural movement of the head, the model requires the definition of virtual bones, used as a skeleton structure for expressing rotations around their joints. Such a virtual skeleton, referred to as the *armature*, consists of a set of bones associated with the geometry of the mesh. 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*), and is illustrated in Figure 4. Clearly, a higher number of bones allows achieving a better accuracy of the movements, which reflects in a more realistic head motion, but increases the computational complexity, both in terms of memory and processing power. After testing different solutions for the armature, we observed that the selected three-bone structure represents a reasonable trade-off to balance realistic motion with computational complexity.

In general, the bones of the armature form a hierarchical chain of links, where each link is connected to a parent (except for the root link). Then, the rotation of each link can be computed as a kinematic transformation relative to its parent. The whole armature has a rotation, scale, and translation with respect to a fixed reference frame. In this way, the animation requires solving a series of kinematic transformations over a period of time. The complete model with graphic rendering is illustrated in Figure 5.

In this application, 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.

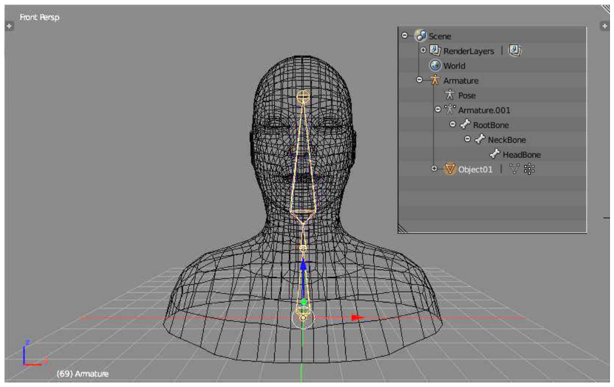


Figure 4. Basic 3D head model with mesh and armature.

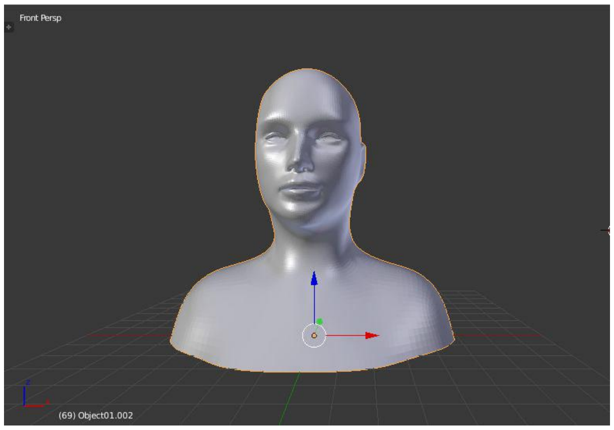


Figure 5. Complete 3D head model with graphic rendering.

Note that the model generated by Blender (exported in a .fbx file format) had to be converted into a format suitable for rendering on LibGDX, which can only import .g3dj and .g3db formats, because they are smaller and faster to load. Since the fbx file does not include texture information, a simple pink texture in .jpg format was created to wrap the 3D model and simulate the human skin. The sequence of operations described above to integrate the model into the Eclipse IDE is illustrated in Figure 6. Note that the Headtracking package containing all the folders and dependencies required by the LibGDX library has been generated by the LibGDX project setup.

IV. APPLICATION STRUCTURE

The application has been developed using the LibGDX library [22], 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.

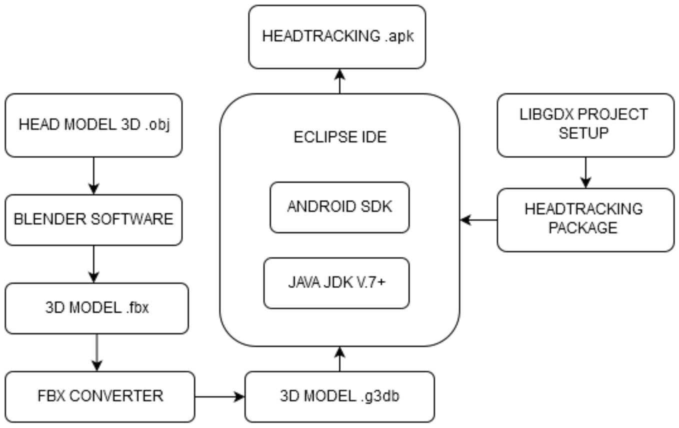


Figure 6. Block diagram of the main software components.

- 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 7.

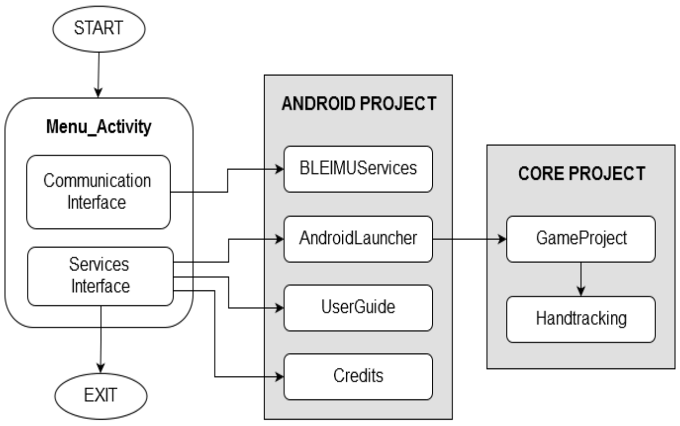


Figure 7. Application Structure.

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. A screenshot of the main menu is shown in Figure 8.

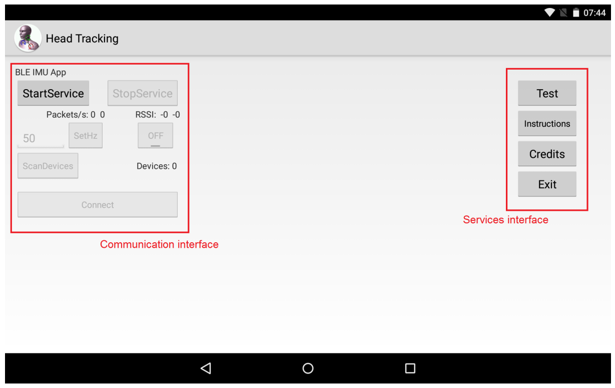


Figure 8. The main menu.

A. *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.

B. *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 the menu illustrated in Figure 9, which 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.

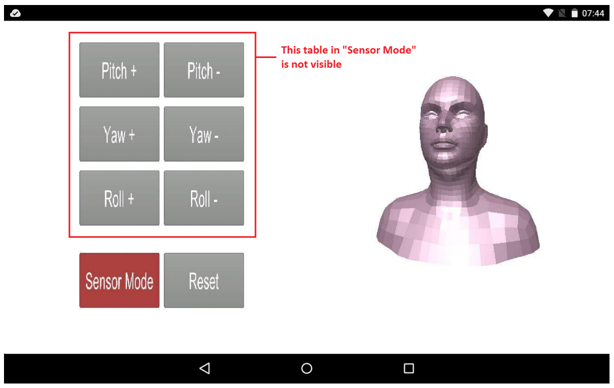


Figure 9. The Test Activity.

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

V. 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.

A. Orientation accuracy

The orientation errors of the sensor node was evaluated with respect to a reference given by a Polhemus Patriot system [1], 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.

In the performed test, both the Polhemus and the head sensor have been mounted on a rotating link, as illustrated in Figure 10, to avoid any measurement error caused by misalignments.

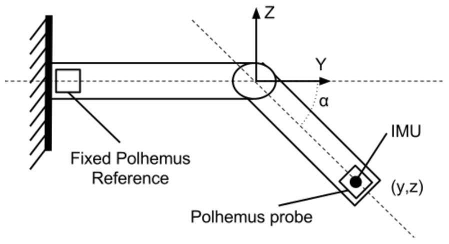


Figure 10. Experimental setup used to evaluate the accuracy.

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 11 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.

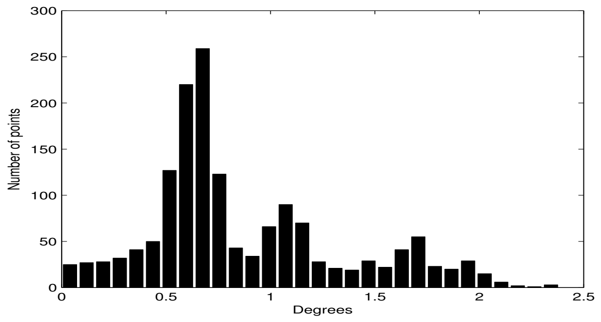


Figure 11. Distribution of the angular error.

B. Processing time and communication delay

Another specific test has been carried out to evaluate the computation time spent by the sensor node to process the data coming from the IMU and the end-to-end delay of data transmission, also taking into account the graphic representation performed by the tablet.

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. In particular, the following times have been captured on the sensor node:

- arrival times of the interrupts from the IMU device;
- duration of the packet read procedure from the I^{2}C bus;
- duration of data processing.

The results of this experiment are shown in Table I, 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.

TABLE I. EXECUTION TIMES (ms)

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

Unfortunately, GPIO switching cannot be used for measuring the communication delay between sensor and tablet. In fact, tablets do not allow the user to change a GPIO pin state in a simple way, as in microcontrollers, and there is no access to the internals of the BLE stack.

For these reasons, the message delay was 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 seconds.

The results of this test are shown in Figure 12, 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. Table II 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. The high variance observed in the end-to-end delay measurements is due to a number of reasons, among which the transmission variability of the BLE protocol, the lack in the Android framework of a structured support for managing time constraints at the application and communication layer, 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.

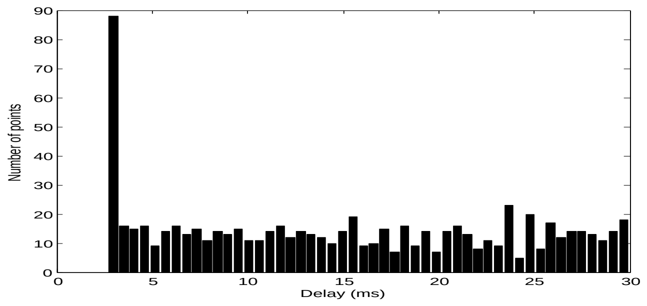


Figure 12. Distribution of communication delay, including the processing time of the Android application.

TABLE II. DELAY TIMES (ms).

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

VI. 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 a large number of applications, including rehabilitation, sport, gaming, and virtual reality. As a future work we plan to integrate the presented tracking device in a more complete monitoring system for detecting driver distractions to improve road traffic safety.

REFERENCES

[1] Polhemus. Polhemus Patriot Product Specification. [Online]. Available: http://polhemus.com/support/product-manuals/ [retrieved: March, 2016]

[2] Vicon. Mx hardware reference manual. [Online]. Available: http://www.udel.edu/PT/Research/MAL/ [retrieved: March, 2016]

[3] Microsoft Corp. Kinect. [Online]. Available: http://www.xbox.com/en-US/xbox-360/accessories/kinect [retrieved: March, 2016]

[4] Xsens Technologies. Mti technical documentation. [Online]. Available: https://www.xsens.com/wp-content/uploads/2013/12/MTi-User-Manual.pdf [retrieved: March, 2016]

[5] InterSense Inc. Inertiacube2+ manual. [Online]. Available: http://www.intersense.com/pages/33/142/ [retrieved: March, 2016]

[6] 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.

[7] 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, Sept. 20-22, 2004.

[8] M. Keir, C. Hann, G. Chase, and X. Chen, "Accelerometer-based Orientation Sensing for Head Tracking in AR & Robotics," in Proc. of the 2nd International Conference on Sensing Technology (ICST 2007), Palmerston North, New Zealand, November 26-28, 2007.

[9] H. Zhou and H. Hu, "Inertial motion tracking of human arm movements in home-based rehabilitation," in Proc. of IEEE Int. Conference on Mechatronics and Automation, Niagara Falls, Canada, July 2005.

[10] D. Roetenberg, H. Luinge, and P. Slycke, "Xsens mvn: Full 6dof human motion tracking using miniature inertial sensors," Xsens Motion Technologies BV, Tech. Rep, Tech. Rep., April 2013.

[11] 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 International Immersive Projection Technologies Workshop (9th Eurographics Workshop on Virtual Environments), 2003.

[12] J. C. Lee. Head-tracking for desktop VR displays using the Wii remote. [Online]. Available: http://www.cs.cmu.edu/j̃ohnny/projects/wii/ [retrieved: March, 2016]

[13] Google. Google cardboard. [Online]. Available: http://www.google.com/get/cardboard/ [retrieved: March, 2016]

[14] O. V. LLC. Oculus gear vr. [Online]. Available: http://www.oculus.com/en-us/gear-vr/ [retrieved: March, 2016]

[15] Z. VR. Zeiss vr one. [Online]. Available: http://zeissvrone.tumblr.com/ [retrieved: March, 2016]

[16] W. Labs. Run-n-read. [Online]. Available: http://weartrons.com/ [retrieved: March, 2016]

[17] M. Benedetto, A. Gagliardi, P. Buonocunto, and G. Buttazzo, "An android application for head tracking (poster paper)," in Proceedings of the 31st ACM Symposium on Applied Computing (SAC 2016), Pisa, Italy, April 4-8 2016.

[18] Android. Android operating system. [Online]. Available: http://www.android.com [retrieved: March, 2016]

[19] Oracle. Java se documentatio. [Online]. Available: http://www.oracle.com/technetwork/java/javase/documentation/index.html [retrieved: March, 2016]

[20] Google. Bluetooth low energy. [Online]. Available: http://developer.android.com/guide/topics/connectivity/bluetooth-le.html [retrieved: March, 2016]

[21] Blender. Fbx file format specification. [Online]. Available: http://code.blender.org/2013/08/fbx-binary-file-format-specification/ [retrieved: March, 2016]

[22] LibGDX. Libgdx. [Online]. Available: http://libgdx.badlogicgames.com/ [retrieved: March, 2016]

[23] 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.

[24] N. Semiconductor. nrf51822 product specification v1.3. [Online]. Available: http://www.nordicsemi.com/eng/Products/Bluetooth-Smart-Bluetooth-low-energy/nRF51822 [retrieved: March, 2016]

[25] Bluetooth. The low energy technology behind bluetooth smart. [Online]. Available: http://www.bluetooth.com/Pages/low-energy-tech-info.aspx [retrieved: March, 2016]