U-Art: Your Art and Ubiquitous Art

Tzu-Chieh Tsai
Chung-Yu Chen
Gon-Jong Su
Department of Computer Science
National Chengchi University
NO.64, Sec.2, ZhiNan Rd., Taipei
City 11605, Taiwan
ttsai@cs.nccu.edu.tw
102753022@nccu.edu.tw
102971022@nccu.edu.tw

Ubicomp/ISWC'15 Adjunct , September 07-11, 2015, Osaka, Japan
ACM 978-1-4503-3575-1/15/09.
http://dx.doi.org/10.1145/2800835.2807927.

Abstract

With wearable computing a new art form is evolving not only for artistic performances but now you can also interact with live digital performance content. Integrated with AR/VR (Augmented Reality/Virtual Reality) technologies, the future art performance will link with virtual characters. In this way the future artistic work may no longer be presented only on a stage, theater, or concert hall. In this paper we will present our idea on how to combine wearable sensors with art performances. Some technical problems need to be addressed. We surveyed the wearable hardware in the market and implemented the WISE (Wearable Item Service runtimE) platform. While wearing the sensors, skeleton (posture) information is delivered in real-time to WISE to render animations displayed on the screen or wearable glasses. This performance can be performed anywhere because it can be quickly deployed and supports improvisation. Even though there are hardware limitation due to non-customized hardware, we can still execute a live field performance to validate the idea and obtain feedback. We believe this will be a new art venue.

Author Keywords

Motion capture; wearable computing.

ACM Classification Keywords

C.2.1 [Network Architecture and Design]: Network communications

Introduction

There are many performers playing with virtual characters based on pre-calculated animations and physics simulations. In [1], the performer uses his sword to interact with the virtual animals on the screen. This type of user input requires a lot of practice and a novice may not play with or interact with the performance. If we can somehow capture the motion of the performers in real time then we are able to support interactions with virtual characters. If the performance can be improvised the experience will be more real. To capture motion we tried a variety of sensors on the market and improved the performance to reach the requirement.

We also present in this paper some problems that we can't overcome currently. A commercial product, Xsens MVN motion capture[2], will also be utilized for the performance to see the difference and compare the flexibility.

Development Evolution

The interactive performance may need a skeleton (motion capture) for the performers. Depending on the artistic work, it may need the position of the performers. To this end, we surveyed various sensors for studying Skeleton rotation and User positioning.

Skeleton rotation

First we used a Raspberry Pi model B to obtain RSSI from a TI-CC2540 BLE interface for motion capture (see figure 1). We found that the frame rate is limited to between 5 ~ 20 fps. It is also not stable and nor precise.

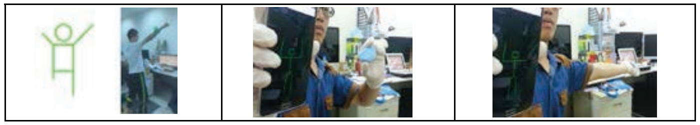


Figure 1: Use RSSI to track human body.

Next we used the Pololu MinIMU-9[3] with gyro, accelerometer, and compass to capture human body. This is not a cheap method and the available package is not enough. Then we tried the MPU6050[4] and found some useful tools. With the library from Jeff Rowberg [5], we can obtain the pitch, roll, yaw of the sensors with gyro and accelerometer and reduce the noise. In addition we used Arduino Uno as our bridge to control it, and tried to make our sensors wireless.

Finally we surveyed the following BLE boards. We used Redbearlab BLE Shield[6] and BLE Mini[7] but the 'AT' commands were too limited. Then we surveyed other TI-CC2540 sensors (see figure 2).

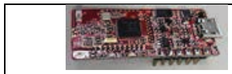
		
Redbearlab BLE Mini	HM-10	MBTV4

Figure 2: Three BLEs we used.

In order to use RSSI for human tracking, we put BLE devices in different locations to map the RSSI and used three BLE devices to receive peripherals. The mapping is shown in figure 3.

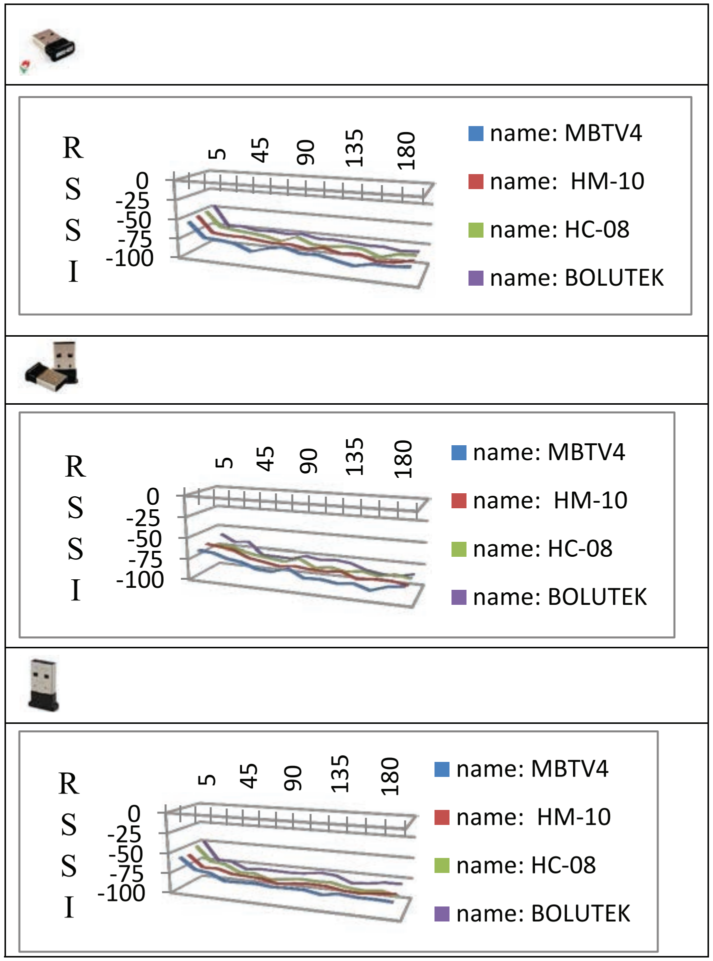


Figure 3: RSSI maps to distance from different sensors.

This experiment results showed that the MBTV4 is stable on RSSI.

distance(cm)

Next we used the Raspberry Pi as a signal collector and put an Arduino Uno MPU6050 and TI-CC2540 BLE on the body to get skeleton rotation (see figure 4).

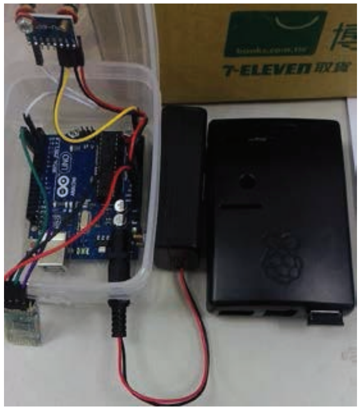


Figure 4: Architecture of all sensors

Finally we made our sensor as small as possible. We chose Arduino Nano instead of the Uno and re-designed the circuit diagram (figure 5) to combine the Arduino Nano and BLE together. We also surveyed batteries to make wearable motion capture devices more light and comfortable. The lithium battery we used can last for 10 hours. Also we designed and printed its 3D shells (figure 6).

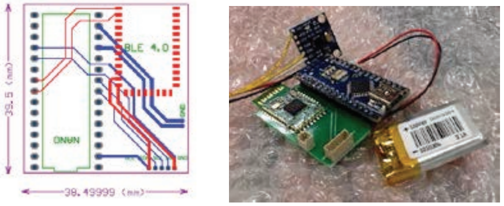


Figure 5: Circuit diagram of BLE and Arduino Nano.

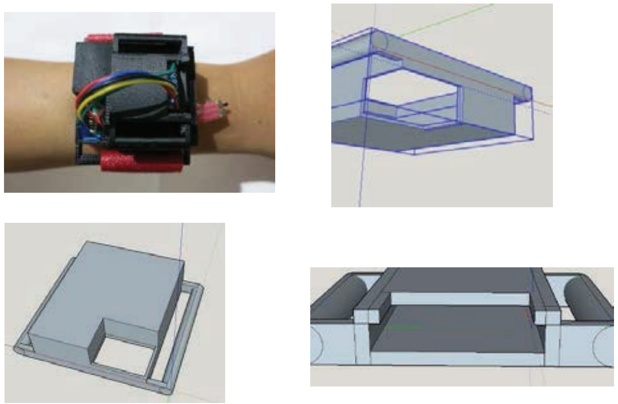


Figure 6: 3D shell.

However when we tested the MPU6050 we discovered some bias and noise. We believe that this is because human muscle is not strictly horizontal or vertical surfaced.

The gyroscope signal can be modeled as:

$$G_i = GV_i + GB_i + GN_i \qquad (1)$$

Where GV_i,GB_i and GN_i are the angular velocity vector, the related angular velocity vector bias, and a white noise.

The acceleration signal can be modeled as:

$$A_i = AV_i + AB_i + AN_i \qquad (2)$$

Where AV_i, AB_i and AN_i are the angular velocity vector, the related angular velocity vector bias, and a white noise.

To overcome it, we use calibration to make it precise, as following.

$$Y = A \times X + B$$

$$\begin{bmatrix} Y_w \\ Y_p \\ Y_r \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \times \begin{bmatrix} X_w \\ X_p \\ X_r \end{bmatrix} + \begin{bmatrix} B_w \\ B_p \\ B_r \end{bmatrix} \qquad (3)$$

Y is the corrected value, and X is what we detect from sensors with some noise and bias. Then each Y and X with yaw(w)、pitch(p)、roll(r), used A and B to map X to Y.

User position

Over the years many moviemakers use high quality cameras to distinguish body postures. Although cameras can record with high precision, the light and other factors may affect precision. To this end we tried a number of methods to locate positions.

First we integrated the acceleration from MPU6050 to measure distance. We discovered that due to the gravity of Earth, the acceleration from MPU6050 includes noise and gravity. We tried to filter it out and measure the distance, but it still suffers from accumulation of error over time (see figure 7). Therefore, it was too difficult to use the acceleration to obtain a real distance for use in such a performance scenario.

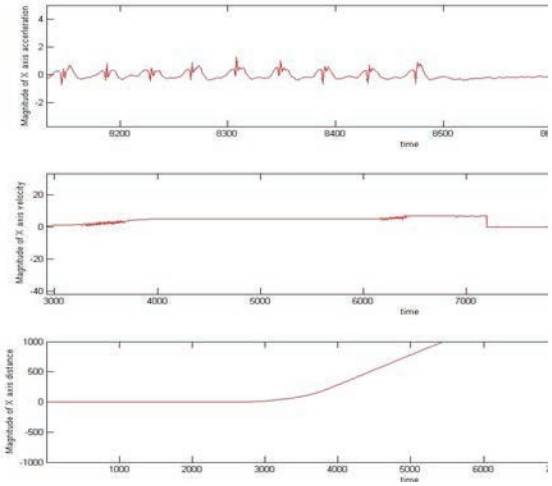


Figure 7: Accelerate, velocity, and distance from MPU6050.

Secondly we tried to use the rotation of legs to measure the distance as described in Bogdan Muset [8]. We analyzed the degree interval of legs (figure 8)[9]. However, there were too many patterns for human moving. It was not possible to handle all the cases.

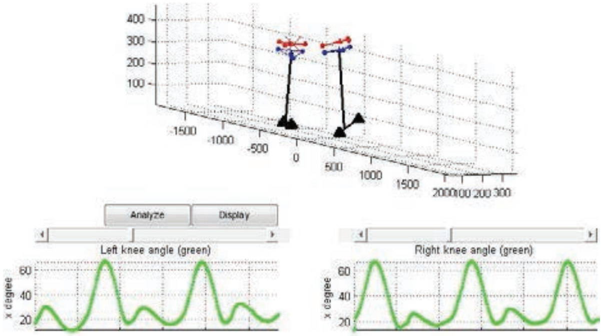


Figure 8: The degree between legs.

Finally the wireless location by BLE was used to find the position. There are three steps for us to get position as precise as possible.

Step 1:
Use Raspberry Pi Model 2B to scan RSSI from BLE and turn RSSI to determine distance by the following model:

$$\text{Distance} = \log_{12} 1.5 * (\text{rssi} / \text{measuredPower}) - 1 \quad (4)$$

where measuredPower is the value we obtain from the beacon at 1m.

Then we use a moving average filter to reduce noise and improve identification. However the RSSI is susceptible to drift and not stable so we tried the following algorithm. At the beginning we use triangle localization. Because BLE distance can be detected precisely at near distance within 30 cm, we give different weights for three near BLEs to improve the precision (figure 9).

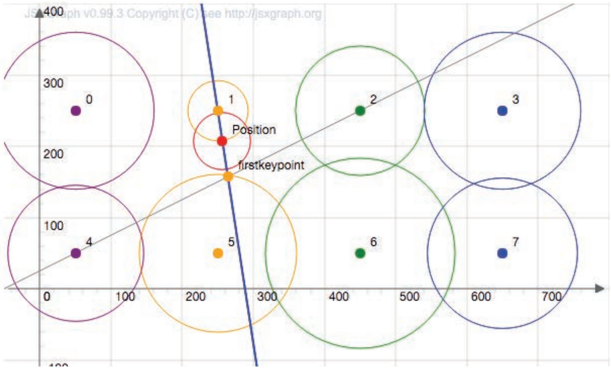


Figure 9: The position we calculated by 8 BLEs.

Step 2:

After we sort all BLE and determine three near BLE namely BLE_1, BLE_2, and BLE_3, then we then weight BLE_2 and BLE_3 to obtain the First_keypoint:

$$\text{First\_keypoint} = \text{BLE}_2 + \frac{(\text{BLE}_2)}{(\text{BLE}_2+\text{BLE}_3)} * \text{Vector}_{2,3} ,$$

where $\text{Vector}_{2,3}$ is from BLE_2 to BLE_3 (5)

Finally we weight the First_keypoint and BLE_1 to obtain the real position.

$$\text{Real position} = \text{BLE}_1 + \frac{(\text{BLE}_1)}{(\text{BLE}_1+\text{First\_keypoint})} * \text{Vector}_{1,\text{First\_keypoint}} \qquad (6)$$

Due to the fluctuating nature of the RSSI signal, we also utilized the user leg movement information measured from gyro sensors. That is, we combine the rotation of legs from MPU6050 as the credibility to reduce the errors of RSSI. The calculation is as follows.

$$\text{Position}_{\text{final}} = \text{A}*\text{Position}_{\text{RSSI}} + (1\text{-A})*\text{Position}_{\text{final}-1} ,$$

where A is the change of legs from MPU6050 , range from 0.5 m to 1m (7)

Step 3:

We still face the question of how many BLE sensors should we deploy in the 2m * 6m capture area? A test was performed with 8, 11, and 21 BLE sensors (table 1). Based on these results, we deployed 21 BLE sensors on the stage for determining position (figure 10).

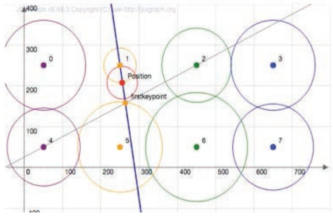
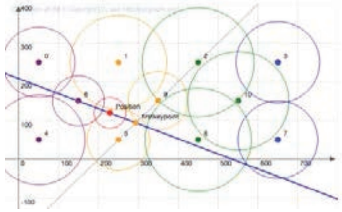
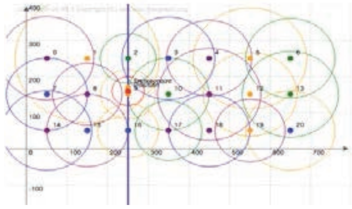
Number of BLE devices	8			11			21		
Distance from BLE	100cm	30cm	10cm	100cm	30cm	10cm	100cm	30cm	10cm
Real position	(250,150)	(250,220)	(250,240)	(250,150)	(250,220)	(250,240)	(250,150)	(250,220)	(250,240)
Calculate value	(390,218)	(242,189)	(263,210)	(299,163)	(245,171)	(257,187)	(298,181)	(238,199)	(256,204)
Error	155 cm	32 cm	32 cm	50.6 cm	49 cm	53.4 cm	57 cm	15 cm	36.4 cm
Graph									

Table 1: Positions for different BLE devices.

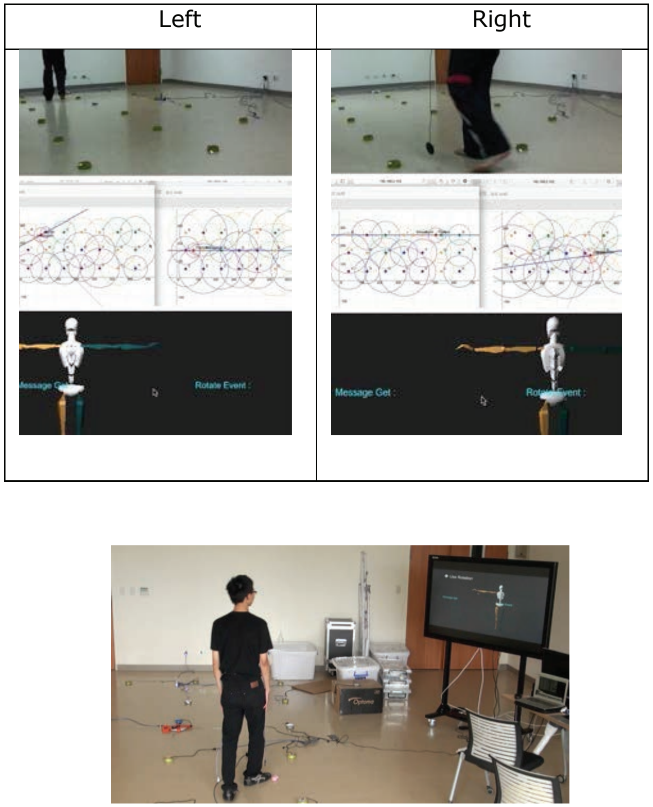


Figure 10: Human indoor motion capture and results.

Performance Execution

We used the Wearable Item Service runtimE (WISE) server as the platform. As shown in figure 11(a), the WISE items are responsible for gathering motion data of the performers and transmitting to a WISE Coordinator via BLE (Bluetooth Low Energy) protocol in real time. Our team used Unity to create virtual characters and let performers wear our sensors to control performance content.

As shown in Figure 11(b), we performed in our campus on May 19, 2015. One wears Xsens and the other wears our sensors. After the show the audience can try to wear our sensors to experience (see figure 12) as if he/she were the performer. In addition to the stage show, our team also used AR to create virtual character interacting with real people with wearable sensors (May 20, 2015) (see Figure 13).

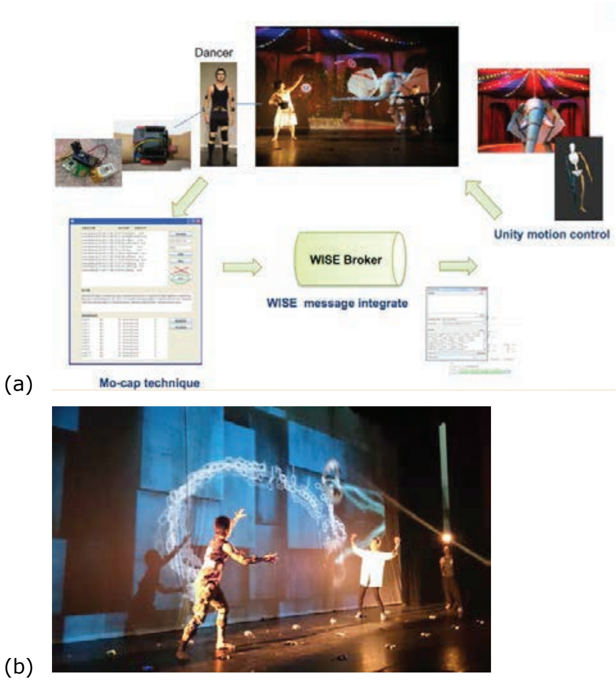


Figure 11: (a) System architecture and show on the performance (b) wearable sensors works with Xsens.

Figure 12: Experiencing our sensors

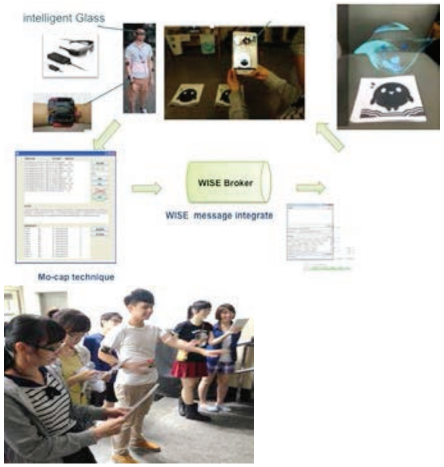


Figure 13: AR coordination with WISE and using our sensors to interact.

Conclusions

We have surveyed a number of sensors to validate usability and combine them with BLE to create a new form of artistic performance. There are still some problems that need to be overcome such as fastening wearable sensors on the body and updating the rate of human position. In the future we will further re-design different sensors for the performer comfort and wearability to control more aspects of the virtual characters.

Acknowledgements

This work is granted by MOST project 103-2218-E-004-001.

Many thanks to our team members, especially to Chun-Feng Liao, and Hsin-Chien Huang.

References

1. https://www.youtube.com/watch?v=nNushriHQ4Q &feature=youtu.be
2. Daniel Roetenberg, Henk Luinge, and Per Slycke. version Apr 3, 2013.Xsens MVN: Full 6DOF Human Motion Tracking Using Miniature Inertial Sensors. XSENS Technologies.
3. https://www.pololu.com/product/1264
4. Jia Ming Deng, Jun Shuai Qiu, Zhi Yuan Zhong, and Zhi Ping Wan. 2015.Three-dimensional Trajectory Tracking System Based on Compass and Gyroscope. International Conference on Education, Management, Commerce and Society.
5. https://github.com/jrowberg
6. http://redbearlab.com/bleshield/
7. http://redbearlab.com/blemini/
8. Bogdan Muset, Simina Emerich.2012.Distance Measuring using Accelerometer and Gyroscope Sensors. Journal of Electronic and Computer Engineering.
9. http://www.ym.edu.tw/~cflu/CFLu_course_matlab gui.html