SHAMrOCk: Sphero HAnd MOtion Control

Emil TYLÉN, Timothée GERMAIN & Hector ANADON LEON
Luleå tekniska universitet
Sweden

Abstract— **To get better familiarised with motion control systems, we developed such a system to control a Sphero (robotic ball) with a LeapMotion (device that tracks hands and fingers) through a computer.**

The resulting system has no real use, but is rather flexible and can easily be modified to control other things.

I. INTRODUCTION

As motion controls and gesture detection systems become more well-known and used by both regular consumers and professionals alike, the benefits of learning how to set-up, implement and utilize such a system become ever larger. New, relatively cheap hardware is emerging that makes this work even easier. One example of such hardware is the LeapMotion. The LeapMotion device is easy to integrate into a project due to a well-defined and documented API and drivers for all big platforms, which includes ARM support.

This project was started to gain an increased knowledge about these devices.

To get a relatively simple introduction to how the LeapMotion works, and how it can be used, the Sphero robotic ball was chosen as a sort of ”companion” device to the Leap. This is because both of the devices have a Python API, and because the Sphero provides a very physical, direct way to see and understand what the Leap is doing.

II. GESTURE BASED INTERACTION

A. LeapMotion

The LeapMotion is a sensor device that supports hand and finger motions as input, analogous to a mouse, but requiring no hand contact or touching.

Its firmware has a built-in gesture recognizer that can detect things like grasping, gripping, rolling a finger or pointing with a pen.

The device is really precise. It has an accuracy of millimetres while tracking hands and fingers position. [1] [2]

B. Sphero

The Sphero is a spherical robot toy designed by Orbotix. It is capable of rolling around and it is controlled through Bluetooth.

It also has some LEDs inside, so the ball can change colours while rolling.

Nowadays, there are two versions in the market. The later version (the V2.0), although heavier, is faster and has higher turning speeds.

III. IMPLEMENTATION

The implementation is written in *Python*, using the official Sphero driver[3].

The driver is multi-platform, but the Bluetooth stack implementation in Linux is not complete, which means that the program does not run correctly. Therefore, Windows is recommended.

To move the Sphero, the official API provides the following function:

```
roll(speed, heading)
```

where *speed* is a 0-255 value representing 0-max speed of the Sphero, and *heading* is a value in degrees from 0 to 359 from the base heading direction. When turning, the Sphero automatically figures out which direction is faster and uses that.

A. Communication with the Sphero

At the beginning of the project, the communication with the Sphero was designed to be handled synchronously, so that messages from the device could be handled in order. However, either the driver does not support this or the messaging system is set up wrong. Whatever the reason, this forced the message handling to be asynchronous.

Although Bluetooth in Windows is more reliable, the connection sometimes gets lost and needs to be restored manually.

Sphero V1 and Sphero V2 communicate in a different way. The V2 has to be sent one order per frame, while the V1 keeps the last order and executes it every frame.

B. First version: 8 directions control

In order to test the communication, the first control system was a simple 8 direction control for the heading of Sphero (up, down, left, right and combinations thereof) without controlling the speed (it was always the same).

It only needed two dimensions to interact (X and Z). The height of the hand is not used but has a minimum value (about 15 cm above the LeapMotion) in order to have enough space to move in side the vision cone of the device.

C. Final version: Analog control

The second and implementation uses analog control for both speed and angle. This implementation still only uses 2 dimensions.

It requires the user to hold their hand still above the LeapMotion and press the *enter* key to calibrate a zero point. After this is done, all positional changes are reported and turned from cartesian coordinates (X/Y) into polar coordinates (r/). These coordinates are then packaged as a command and sent to the Sphero.

If hand position is in the "deadzone" (a small, configurable radius around the starting point), the ball will not move. When the hand leaves it, it will start moving. This is mainly for convenience, as stopping the baal would otherwise require exact precision.

This is the way the position of the hand (xpos, zpos) related to the base position (0,0) and the speed (length of the vector) and angle (of the vector) are calculated.

```
xpos = hand.palm_position[0]-self.x
zpos = -(hand.palm_position[2]-self.z)
speed = math.sqrt(xpos**2 + zpos**2)
angle = (math.degrees
        (math.atan(xpos / zpos))) % 360
```

The final implementation also utilizes the LEDs to change the colour of the Sphero while on the move. The baal starts out blue, but as it speeds up, it turns red.

	8 directions	*analog*
speed	all or nothing	analog (0-255)
heading	8 fixed values	analog (0-360)

TABLE I
COMPARISON OF THE TWO IMPLEMENTATIONS

D. Setting the heading

As the Sphero is a ball, the way the little robot inside is pointing isn't possible to see. The heading is therefore indicated by a blue LED on its back. This LED is used as a refcerence point and should be calibrated to point at the user before driving to avoid confusion.
However, when the Sphero hits something, the robot inside jumps a bit and the heading moves. This means that after every collision the user has to set it again.

In order to set the heading, the gesture recognition of the LeapMotion is used. The user rolls their finger clockwise or anticlockwise to slowly rotate the Sphero on the spot until the rear LED is pointing at them.

In the first version, no filtering was done. This meant that the rotaion gesture could be active at the same time as the movement gesture, which led to some weird issues as the ball could change heading while moving, without the user noticing. It was solved by making it compulsory for the Sphero to be at a complete stop when setting the heading.

IV. Results

This video is a demonstration of the project showing the functionalities of Sphero controlled through LeapMotion
`https://www.youtube.com/watch?v=PFO7df--THo`
Here is a link to the code of the project in Python and information how to run it.
`https://github.com/shamRockOrg/SHAMrOCk`

V. CONCLUSIONS

Since the aim of the project has mainly been to gain familiarity with the technologies chosen, the conclusions are basic.

A. The LeapMotion

The Leap has all the marks of a great product. It's small, easy to use and has a very simple, modular API that can be tailored to include the functionality desired depending of the scope of a project. Considering its relatively low price-point and despite this very good accuracy, the Leap is a viable replacement to traditional control methods in places where these would be otherwise undesirable.

Examples include:

- Places where hygiene is important, such as hospitals or food processing plants,
- Places where wear and tear is a big factor, like mines and factories,
- People with damaged extremities or nerve damage hindering fine motor control.

All in all, a very versatile system that is accurate enough for daily use.

B. The Sphero Robotic Ball

The Sphero is a toy. Nothing more, nothing less. It has no practical use other than to amuse. To this end, its API is rather broken. This is especially true between versions, as the message structure differs slightly between the old and the new ball. Nowhere is this documented.

Developing for the Sphero quickly leads to frustration, especially considering that there is no actual reason for using it. In the defence of Orbotix, they do not advertise the Sphero (or, for that matter, its successor, the Ollie) as anything other than a toy.

The Sphero has fit the needs of this project, acting only as an "avatar" of sorts to illustrate the actions of the LeapMotion. However, it has few other uses.

C. Future work

The LeapMotion has lots of fascinating uses, as documented above. Connecting it up with some other, more practical systems would a priority for any potential future work. To do this, a standardized control scheme should be developed. The Leap already has a databank of gestures that can be used to achieve this goal. After some sort of general method of control has been established, the Leap can be plugged into a multitude of systems and be ready to go as a replacement for, say, a keyboard or touch screen.

ACKNOWLEDGMENTS

The authors would like to thank Kåre Synnes for his support.

References

[1] Leap Motion — Mac & PC Motion Controller for Games, Design, & More. 2014. Available at: http://www.leapmotion.com.

[2] Weichert, F., Bachmann, D., Rudak, B., Fisseler, D., Analysis of the accuracy and robustness of the leap motion controller, Sensors, 13(5) (2013) 63806393.

[3] Python driver for the sphero from the Robot Operating System (ROS) project Available at: https://github.com/mmwise/sphero_ros/blob/groovy-devel/sphero_driver/src/sphero_driver/sphero_driver.py