Loughborough University

Final Year Project

January Deliverable

Prakash Waghela
B221256

Technical Aspects

In this section, I will cover the technical details of my project that I have researched so far.

Myo Armband

The Myo Armband (Thalmic Labs, 2017) is a wearable band to be worn around the forearm of the user.

Sensors

- 8 Electromyography sensors
    - The EMG sensors can be used to detect the electrical signals in the muscle. These readings can be manipulated to detect specific hand gestures and motions.
    - The raw EMG readings can be captured from the Myo through the SDK with a single reading being available per sensor, at a rate of 200Hz (Bernhardt, 2015).
- Three-axis gyroscope
    - The gyroscope can be used to detect rotational movement of the arm. There are three planes in which an object can be rotated (roll, pitch, yaw), all of which can be detected by the armband.
    - The data from the gyroscope can be captured through the SDK, with a single reading in degrees per second (Thalmic Labs, 2017) being available for each rotation axis at a rate of 50Hz (Bernhardt, 2015).
- Three-axis accelerometer
    - The accelerometer can be used to detect movement of the arm. In specific, the sensor can detect the change in velocity and position of the arm.
    - The data from the accelerometer can be captured from the Myo through the SDK, with a single reading in units of G (Thalmic Labs, 2017) being available per sensor at a rate of 50Hz (Bernhardt, 2015).
- Three-axis magnetometer
    - A magnetometer can be used to detect magnetic fields. In theory, this should allow the armband to act as a compass, using the Earth's magnetic field to determine the absolute orientation.
    - I have not found any function in the SDK that allows direct access this sensor, so this sensor may be used in improving the accuracy of the values reported by the other sensors.

Interface

- Connectivity - The Myo armband has a Bluetooth interface that allows the user to connect the armband to a variety of compatible devices. This means the device can be operated without the use of wires, allowing for unrestricted movement while using the device.
- Battery - The Myo armband has a built-in battery which can be recharged through a micro USB port on the device. The battery is advertised to allow for "One full day use out of a single charge" (Thalmic Labs, 2017).
- Feedback - The Myo armband can provide direct feedback through two methods.
    - Haptic - The armband utilises short, medium, and long vibrations to notify the user of different events.
    - LEDs - The armband utilises LEDs on the device to notify the user of different device statuses.

Usability

The Myo armband is a lightweight and small device which easily fits around the forearm of the user.

- Gestures - The software for the Myo armband has five pre-programmed hand gestures (Thalmic Labs, 2017) that it can detect through the EMG sensors.
- Calibration - The software runs the user through a calibration step to create a user profile for the armband. This improves the reliability of the device and is specific to each user due to the variation in EMG readings through the arm of the user.

My Experience

In testing the Myo armband, my goal was to gain an understanding of how to extract data from the armband and find a method of analysing the data in some form.

I found that the armband did not have any stand-by mode. This meant that I would need to charge the armband before every use as any leftover charge from the previous use would be wasted while the device is idle.

Connecting to the device was incredibly simple. The software package installed smoothly and connected to the Myo easily.

I found that the armband did not recognise the pre-programmed gestures very accurately. This could be frustrating at times when completing the tutorial as I would perform the required gesture but it would not be detected by the armband. Creating a user profile through the software improved the reliability of the gesture detection but the gestures required very exaggerated movements to be detected correctly. As an example, for the “fist” gesture to be detected, I would need to ball my hand into a tight fist and tense my arm for the gesture to be detected, rather than balling my hand into a relaxed fist like I would prefer.

The SDK for the device supports Windows and has a library available in C++. The SDK was bundled with sample applications that demonstrated how to programmatically connect with the device and pull raw data from the device.

I wrote a script to pull the raw EMG data from the device which tested while my arm was stationary. This gave me a large data set as I had a reading for each sensor on the armband (8) at 200Hz. I exported this data set to a spreadsheet where I averaged the readings and plotted them on a graph. From this graph, I was looking to see if there would be any trends I could use to detect the use of the muscle. However, as I experienced a large variation in results while my arm was stationary, I concluded that I would be unable to utilise the EMG data to accurately capture the use of the arm.

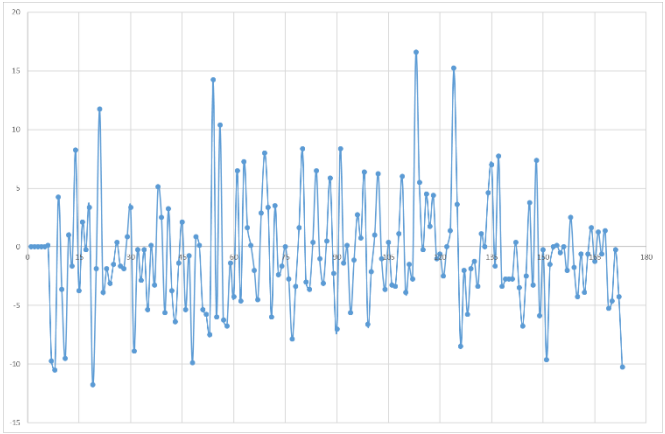


Figure 1- Graph showing raw EMG readings while stationary

Within the sample program provided in the SDK, there was a small program that utilised the gyroscope to detect the rotation of the arm around the three planes of rotation. Running this program showed that the device could detect rotation accurately, with the results detected closely matching the rotations I made with my arm.

Leap Motion

The Leap Motion is a small bar-shaped motion detector. The device can be placed on a flat service and comes packaged with software to detect the user's hands.

Sensors

- Infrared Cameras - All data provided by the Leap Motion is derived from images taken by two infrared cameras contained within the base (Colgan, 2014), taken at up to 200 frames per second (Leap Motion Inc, 2017a). The Leap software analyses the images and provides data from the motion of the tracked hands.
    - The cameras utilise wide angle lenses creating a large interaction space which is shaped like an inverted pyramid (Colgan, 2014).
    - The images are fed into the Leap Motion software which returns a 3D representation of the device's field of view. The software can detect when a user's hand is captured in an image and the software infers a large amount of data from the analysis of the image, allowing for the user's hand to be tracked in 3D space.
    - This data is accessible through the SDK and is organised into the following structure:

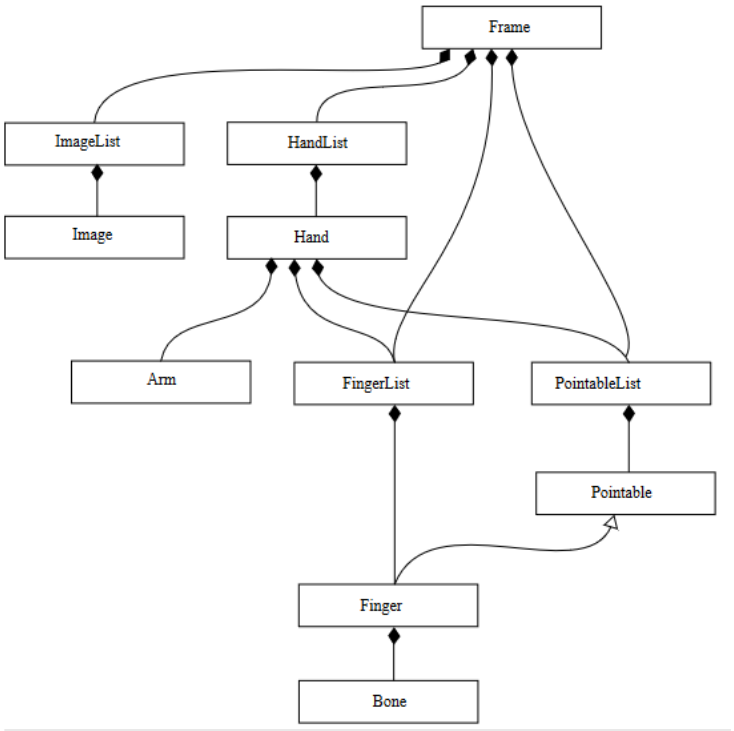


Figure 2 - Structure of data available (Colgan, 2014)

    - The bulk of data available can be accessed through the Hand object. The data allows the user to determine the following about a hand:
        - Handedness - Left or right hand
        - Position - Displacement in relation to the Leap Motion
        - Orientation - Gyroscopic information of the hand
        - Posture - Positioning of fingers to point or grab
        - Motion - The movement of the hand

Interface

- Connectivity - A single wire is required to connect the Leap Motion device to any USB port on a compatible device. The device is powered through the USB connection and does not require charging.
- Feedback - The Leap Motion has a green LED on the side of the panel to indicate that it is connected to a PC. When powered and connected, there are three red lights lit on the surface of the panel. All other feedback is given through the software.

My Experience

In testing the Leap Motion, my goal was to understand what information was captured and made available by the device.

Like the Myo, the Leap Motion was easy to set up. The software installed smoothly and connected with the Leap Motion device easily. The software provided a visualizer which would draw a wireframe hand and trace the movement of your hand as detected by the device.

I used the visualizer to get a feel for how the device performed with different hand motions. The device requires a clear, unobstructed view of your hand for the tracking to be effective. There were cases where some fingers were incorrectly displayed; this normally occurred when some fingers were obscured from view by other fingers. In the example below, only my middle finger was perpendicular to my palm but the visualizer incorrectly detected two fingers being perpendicular.

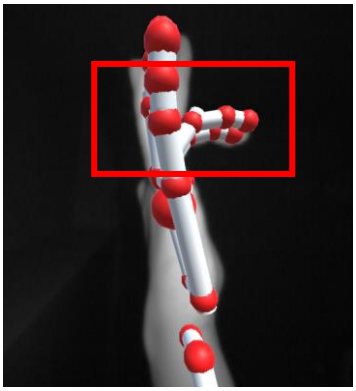
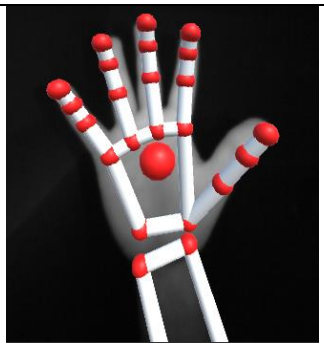
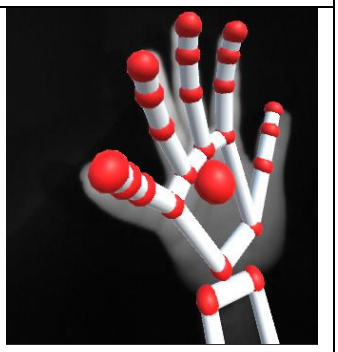
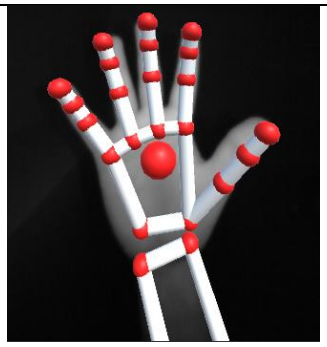
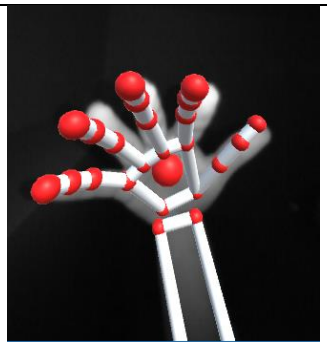
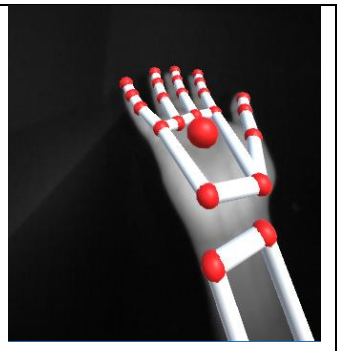
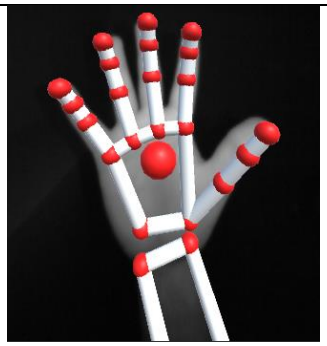
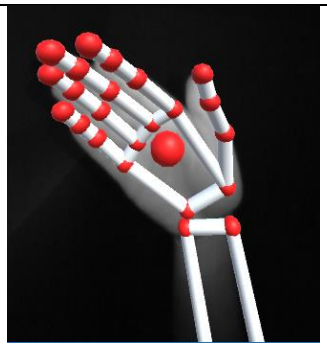
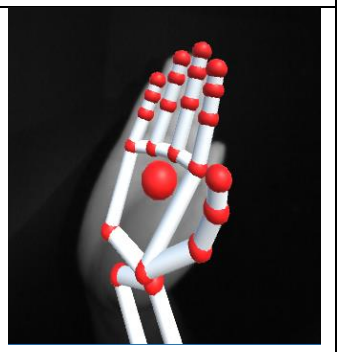


Figure 3 - Visualizer showing an incorrect representation

I found that the device could track the position and rotations of my palm well. To test this I started with my palm facing down, rotated my hand about each axis, and watched the visualisation to see how well the software could track my hand.

Axis	Normal	Anticlockwise	Clockwise
Roll			

Pitch			
Yaw			

I moved on to see how I would be able to access the rotational tracking data. There is support for multiple programming languages across multiple platforms; I chose to work in C++ on the Windows operating system. Like the Myo, the SDK for the Leap Motion was bundled with sample applications.

Most of these applications had very extensive and detailed outputs which were unnecessary for the testing I wanted to conduct. I wrote a simple, console based application that allowed me to output the following basic readings:

- Palm rotation
  - Roll
  - Pitch
  - Yaw
- Palm height above Leap Motion
- Hand grab strength

This allowed me to get a feel for using the SDK and programmatically interfacing with the Leap Motion. During my work on the application, I found there were two methods of obtaining information; listening or polling.

```
Height: 61.5555
Height: 61.2792
Height: 60.5547
Height: 60.0942


Duplicate Frames Processed: 17461080
New Frames Processed: 381
Time Elapsed (ms): 3357
Frame Rate (FPS): 127
```

Figure 4 - Example output

Using a listener allowed for a very efficient processing as frames were processed as they were available. However, this introduced the concept of threading into my application. This has potential to cause problems in the future if the output medium was not able to cope with the frame rate

achieved by the Leap Motion or if the output medium cannot handle multiple input threads. Using polling allows me to fine tune how the frames would be processed but requires more validation to ensure that I am processing the correct frame.

Oculus Rift Development Kit 2

The Oculus Rift DK2 is a head mounted virtual reality headset.

Sensors

- External camera combined with LEDs on the HMD to track position.
- Leap Motion can be attached to the front of the HMD.

Interface

- Output - Virtual reality displays inside the headset.
- Connectivity - Utilises multiple cables and requires a high-performance PC to function.
    - Could possibly utilise a lower-performance PC if only the positional tracking was utilised as the graphics rendering is the cause of the high-performance PC requirement.

Usability

The Oculus Rift DK2 is a heavy device which requires you to be tethered to the PC.

- The headset requires a tight fit to create an immersive experience. This can be uncomfortable for some users.
- While wearing the headset, you will be aware of the wires that tether you to the spot.

My Experience

In testing the Oculus Rift, my goal was to get a basic understanding of what the device was capable of and how it could be utilised in my project.

I installed the required software to run the Oculus Rift easily. Connecting the device to my PC was magnitudes more difficult than it was with the Leap Motion or Myo as there were many different wires that had to be connected for the device to function correctly.

As part of the SDK, there were some sample applications that I used to see what the device was capable of. One such application displayed an ocean-side house scene on the display and allowed the user to move their head around to explore the space.

I found that the device was exceptional at tracking movement and numerical values that were shown on screen showed that this data was available through the SDK.

I also wanted to test how well the Oculus Rift coped at streaming video. By utilising video streaming apps through the Oculus Store, I could experience video playback through the Oculus Rift. The videos I viewed were created for an immersive experience (360° videos) and allowed the user to turn their head to rotate the view of the video. In the past, I have used other VR devices to watch normal, non-360° YouTube videos, with the videos being projected onto a virtual screen.

I found that the video playback could be disorientating unless the quality and playback rate of the video was exceptional. In the case of the virtual projections, I found that the experience was of a lower quality than of that of watching the video feed on a normal screen. I found that the need to wear a head mounted display was a hindrance as the display would need to be always perfectly lined up for the video to be displayed correctly and at times this could be uncomfortable.

Nao Robot

The Nao robot is a humanoid robot that can be programmed to carry out a wide variety of tasks.

My Experience

So far, my experience with the Nao robot has been limited in comparison to the other devices. This is mainly due time constraints at the end of the first semester and being unable to explore the technology fully. However, I have been able to get a high-level introduction to the interface the robot utilises and believe that I have a rough understanding of what it is capable of.

I have found that a Mac or Linux operating system is required to compile applications for the Nao (Aldebaran Robotics, 2015b). This has affected my previous plans as I am unfamiliar with these operating systems so developing on them will be slower. Work has begun to set up a development environment in Linux; I have successfully set up a virtual box running Linux, installed the Leap Motion software, and have successfully compiled and run a sample Leap Motion application.

Progress Made

I believe progress at the start of this project is best defined as the project taking shape and me gaining an understanding of what to aim for. A project to create a telepresence system is very vague and I have spent a large amount of time testing the different devices available to me to deduce how the end application will function. Thus far I have come to the following conclusions:

Oculus Rift

The Oculus Rift is a device created primarily to create very immersive and rich experiences by rendering high definitions scenes to the lens's inside the device. While this requires a large amount of resources and hardware, if done correctly the outcome can be fantastic. There are two ways in which the Oculus could be used in the scope of this project:

- Utilise the motion tracking to allow the user to move their head to control some part of the robot (most likely neck rotation)
- Utilise the internal displays to give the user a live video stream, utilising cameras in the robot.

Using the motion tracking of the Oculus Rift would mainly take the rotation in the yaw axis and translate that into a similar rotation on the robot. However, there are other devices that allow for a similar and potentially better experience, for example using the Leap Motion and using my hand.

Providing a live video stream is a great concept but I don't believe the Nao robot's camera is powerful enough to provide a smooth experience. The camera can provide a 1280x960 resolution at 30 frames per second. A source from Oculus have said that "The best, most immersive and comfortable experience for users runs consistently at 90fps and developers should target this frame rate aggressively.". Therefore, streaming a video feed at 30 frames per second with the low resolution that is available would not result in a smooth experience and would likely cause motion sickness and headaches. As an alternative, it should be possible to provide a live video feed shown on screen through the application.

Myo Armband

The Myo Armband is a great device that is think is best used to track motion. The combination of the accelerometer and gyroscope work well together and the fact that it is wireless is a bonus. My experience with the gesture recognition wasn't good; I believe this was caused by the large number of variables in collecting the EMG data used to recognise gestures.

The Myo armband could be used as an input device. I could translate the motion, both linear and rotational, made by the wearer into motion that the Nao robot would perform.

However, although I think the Myo is functionally better at tracking position than the Leap Motion, I believe the Leap Motion is the better tool to use in my case. This is due to the Leap Motion providing a wider array of parameters I can utilise; on top of the motion data like the data provided by the Myo, I have access to data about fingers being used to point and multiple hands being detected. I believe that having a larger store of data available that I can use will improve the utility of the application as I have more data to work with.

Leap Motion

The Leap Motion is a powerful device that provides a wide range of data to developers. It can track the user's hands well and although it isn't wireless, you don't feel tethered to the desk like you do when using the Oculus.

The Leap Motion will be used as an input device. Initially I would like to create an interface where I can control different parts of the Nao using the rotation of my palm. Building on this, I would like to implement more complicated control schemes that can be utilised to perform more complex actions with the robot. For example, I could utilise different finger combinations to perform different actions.

Depending on how much time I have available, I would like to look at utilising a Raspberry Pi to improve the portability of the application. Preliminary research indicates that I would be able to create a data stream that utilises a wi-fi network to send the input data to a computer for processing, which would then be transferred to the Nao for action. However, more research is required to assess the feasibility of this.

References

Aldebaran Robotics. (2015, 10 06). *Aldebaran - Nao Developer Docs (a)*. Retrieved January 2017, from Aldebaran: http://doc.aldebaran.com/2-1/news/index.html

Aldebaran Robotics. (2015). *Aldebaran - Nao Developer Docs (b)*. Retrieved January 2017, from Aldebaran: http://doc.aldebaran.com/2-1/dev/cpp/index.html

Bernhardt, P. (2015, March 14). *Myo - Blogs*. Retrieved January 2017, from Myo: http://developerblog.myo.com/myocraft-logging-imu-and-raw-emg-data/

Colgan, A. (2014, August 9). *Leap Motion - Blogs*. Retrieved January 2017, from Leap Motion: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/

Dean Beeler, A. G. (2016, March 25). *Oculus - Developer Blogs*. Retrieved January 2017, from Oculus: https://developer3.oculus.com/blog/asynchronous-timewarp-on-oculus-rift/

Leap Motion Inc. (2017, January). *Leap Motion - Developer Docs (a)*. Retrieved January 2017, from Leap Motion: https://developer.leapmotion.com/documentation/javascript/api/Leap.Controller.html

Leap Motion Inc. (2017, January). *Leap Motion - Developer Docs (b)*. Retrieved January 2017, from Leap Motion: https://developer.leapmotion.com/documentation/cpp/devguide/Leap_Hand.html

Leap Motion Inc. (2017, January). *Leap Motion - Developer Docs (c)*. Retrieved January 2017, from Leap Motion: https://developer.leapmotion.com/documentation/cpp/devguide/Leap_Tracking.html

Thalmic Labs. (2017). *Myo - Device Listeners*. Retrieved January 2017, from Myo: https://developer.thalmic.com/docs/api_reference/platform/classmyo_1_1_device_listener.html

Thalmic Labs. (2017). *Myo - Technical Specs*. Retrieved January 2017, from Myo: https://www.myo.com/techspecs