**Gesture Interpreter**

Andrew Ladlow

Contents

1. [Introduction 3](#_Toc433059387)

[1.1 Motivation 3](#_Toc433059388)

[1.2 Objectives 3](#_Toc433059389)

[1.3 Report Overview 3](#_Toc433059390)

1. [Background 5](#_Toc433059391)

[2.1 Research 5](#_Toc433059392)

[2.2 Similar Applications 5](#_Toc433059393)

1. [Design 7](#_Toc433059394)

[3.1 A 7](#_Toc433059395)

1. [Implementation 8](#_Toc433059396)

[4.1 A 8](#_Toc433059397)

1. [The System in Operation 9](#_Toc433059398)

[5.1 A 9](#_Toc433059399)

1. [Testing and Evaluation 10](#_Toc433059400)

[6.1 A 10](#_Toc433059401)

1. [Conclusions 11](#_Toc433059402)

[7.1 Future Work 11](#_Toc433059403)

[References 12](#_Toc433059404)

**Chapter 1**

# Introduction

## 1.1 Motivation

Human-computer interaction has, for many years, been limited to mouse pointing and typing based devices. With the more recent development of smartphones we have seen numerous breakthroughs with touch based devices. These methods are all inherently unnatural when compared with real world interaction, however. In all of these cases users are forced to interact through some kind of interface as opposed to naturally communicating via hand signals or body language.

Currently, there are several released devices which aid in monitoring, recording and displaying these more natural forms of communication. Devices such as the Microsoft Kinect, ASUS Xtion Pro and Apple Primesense Carmine all feature some degree of motion capture – the issue lies in the accuracy of movement, especially with more precise or intricate gestures. The gesture interpreter aims to utilise one of these devices, the Leap Motion, in order to facilitate this natural gesture communication – particularly gestures found in the British Sign Language (BSL).

According to Rudak, et al (2013) [1] the Leap Motion is capable of recognising movement with an overall average accuracy of 0.7mm, a degree that devices such as the Kinect were unable to reach in comparison. In addition, a recent V2 firmware update released for the Leap Motion improved tracking performance and enhanced its API features.

The use of this software will aid users who are deaf or hearing impaired in communicating as they would in real world scenarios via the BSL.

## 1.2 Objectives

The application should fulfil the following main objectives in order to achieve the goal described above:

* Allow for recording of custom, user defined gestures.
* Accurately recognise the formation of gestures and distinguish between them without generating false positives. (Degree of accuracy?)
* Transform the recognised gesture into speech output.

## 1.3 Report Overview

The structure of the report is as follows:

Chapter 1 discusses the problem and the proposed solution, including its required properties. Chapter 2 analyses background research and related work as well as any closely related applications that currently exist or are in development. Chapter 3 … Chapter 4… Chapter 5… Chapter 6… Finally, chapter 7…

**Chapter 2**

# Background

## 2.1 Related Work

The use of the Leap Motion controller in order to detect sign language gestures was previously investigated by Araullo, et al (2012) [2], though in this case the Australian Sign Language (Auslan) was used rather than the BSL. The authors noted that although the controller showed potential, further development of the API was required – this was mainly due to inaccurate hand detection in certain scenarios e.g. pinching fingers together, interlocking hands or holding one hand above the other. It’s important to clarify that the paper was published in 2012, prior to the release of the V2 firmware update for the Leap Motion in 2014 which boasted improved tracking performance. As such it’s likely that the current accuracy of the device is no longer the same as described.

The use of dynamic time warping (DTW) to recognize leap motion gestures in 2D space was explored by Russell, et al (2013) [3]. The authors demonstrate how a database of test data can be searched through using a similarity algorithm. The paper suggests that future work with the algorithm used for recognition could lead to support for gesture recognition in 3D space for sign language interpretation, though this is reliant on recent optimizations to DTW to ensure the recognition process is feasible in real-time.

Hidden Markov Models / Markov Chains?

Neural networks / Support Vector Machines?

The $P recognizer, as discussed by Wobbrock, et al (2012) [4] is another example of a gesture recognition algorithm. As described in the research paper, the $P is “a 2-D gesture recognizer designed for rapid prototyping of gesture-based user interfaces”. The $P aims to overcome the complex task of matching user gestures by instead treating them as groups or “clouds” of points and evaluating each one in turn. Even the simplest of gestures could be created in many different ways depending on the properties of its strokes e.g. start and end points, order or time, and direction. The use of a point cloud helps remove any ambiguity from the gesture which simplifies comparison and recognition. According to the paper, the algorithm requires only 70 lines of code to function and delivered over 99% accuracy in user-dependant testing.

## 2.2 Similar Applications

‘LeapTrainer.js’, created by O’Leary in 2013 [5], is a browser based gesture and pose learning and recognition framework for the Leap Motion. Developed in JavaScript, LeapTrainer allows users to create and store gestures then replay them at will. The software recognises a gesture using a custom template matching algorithm, based off the $P recognizer discussed above. The software supports both gestures and ‘poses’ – the difference being that a gesture is the movement of one or both hands at or above a specific velocity, whereas a pose is the stationary position of one or both hands over a period of time.

A commercial application, UNI, is currently in development and is scheduled to be released in summer 2016 by MotionSavvy [6]. UNI bares similarities to this project in that it utilises the Leap Motion in order to translate gestures into spoken text. However, the software is closed-source and based on a subscription model which limits further adaptation or extension. It is currently unknown which operating systems UNI will support and how well the software will achieve the goals described on the website.

**Chapter 3**

# Design

## 3.1 A

Comparison of explicit and implicit gesture recognition / segmentation e.g. pressing keyboard to start stop vs. detection based on hand speed or position etc.

**Chapter 4**

# Implementation

## 4.1 A

Mention of frame drop / slowdown when comparing every bone direction between current frame and deserialized frame from file, similarly when just comparing palm positions and finger tip positions but to a lesser degree - not viable for real time

**Chapter 5**

# The System in Operation

## 5.1 A

A

**Chapter 6**

# Testing and Evaluation

## 6.1 A

A

**Chapter 7**

# Conclusions

## 7.1 Future Work

Accuracy concerns

Limited to gestures not requiring body or face

Must be stationary to use - possible combination with oculus VR or other head mounted display

### **References**

[1] Rudak B., Bachmann D., Fisseler D., & Weichert F. (2013) Analysis of the Accuracy and Robustness of the Leap Motion Controller. On line publication, PubMed Central, <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3690061/pdf/sensors-13-06380.pdf>. Last accessed 16 October 2015.

|  |
| --- |
|  |
| [2] Potter, L. E., Araullo, J., & Carter, L. (2013) The leap motion controller: a view on sign language. In Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration (pp. 175-178). ACM, 2013.  [3] Vikram, S., Li, L., & Russell, S. (2013) Writing and sketching in the air, recognizing and controlling on the fly. In CHI'13 Extended Abstracts on Human Factors in Computing Systems (pp. 1179-1184). ACM, 2013. |

[4] Wobbrock J. O., Anthony L., & Vatavu R.D. (2012) Gestures as Point Clouds: A $P Recognizer for User Interface Prototypes. On line publication, University of Washington, <http://faculty.washington.edu/wobbrock/pubs/icmi-12.pdf>. Last accessed 13 October 2015.

[5] O’Leary R. (2013) LeapTrainer.js. GitHub repository, <https://github.com/roboleary/LeapTrainer.js>. Last accessed 9 October 2015.

[6] MotionSavvy. (2015) UNI. Website, <http://www.motionsavvy.com/>. Last accessed 9 October 2015.