**Gesture Interpreter**

Andrew Ladlow

Contents

1. [Introduction 3](#_Toc432374052)

[1.1 Motivation 3](#_Toc432374053)

[1.2 Objectives 3](#_Toc432374054)

[1.3 Report Overview 3](#_Toc432374055)

1. [Background 4](#_Toc432374056)

[2.1 Research 4](#_Toc432374057)

[2.2 Related work 4](#_Toc432374058)

[2.3 Similar Applications 4](#_Toc432374059)

1. [Design 5](#_Toc432374060)

[3.1 A 5](#_Toc432374061)

1. [Implementation 6](#_Toc432374062)

[4.1 A 6](#_Toc432374063)

1. [The System in Operation 7](#_Toc432374064)

[5.1 A 7](#_Toc432374065)

1. [Testing and Evaluation 8](#_Toc432374066)

[6.1 A 8](#_Toc432374067)

1. [Conclusions 9](#_Toc432374068)

[7.1 A 9](#_Toc432374069)

1. [References 10](#_Toc432374070)

**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 publically available devices which aid in monitoring, recording and displaying these more natural forms of communication. The gesture interpreter aims to utilise one of these devices, the Leap Motion, in order to facilitate natural gesture communication – particularly gestures found in the British Sign Language (BSL). The use of this software will allow users who are deaf or hearing impaired to communicate 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:

* Record gestures for BSL letters in order to account for each user’s own individual communication style.
* Accurately recognise the formation of these gestures and distinguish between them without false positives.
* Allow for recording and recognition of custom, user defined gestures to improve the scope of the system.

## 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**

# Background

## 2.1 Research

As a key aspect of the project, it’s crucial to investigate the various methods in which gesture recognition has been previously developed for any and all systems, not just those linked with Leap Motion.

## 2.2 Related Work

Despite being a relatively new device, Leap Motion has seen numerous projects developed for it.

One such example is ‘LeapTrainer.js’, created by O’Leary in 2013 [1]. As described in the readme file on its GitHub repository, LeapTrainer is “A gesture and pose learning and recognition framework for the Leap Motion.” Developed in JavaScript, LeapTrainer allows users to create and store gestures. The software recognises a gesture using a custom template matching algorithm. 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.

## 2.3 Similar Applications

A commercial application, UNI, is currently in development and is scheduled to be released in summer 2016 by MotionSavvy [2]. UNI utilises the Leap Motion in order to translate gestures into spoken text. As shown on the current site, there will be two models available to order – a base version including only the Leap Motion, and a ‘PRO’ version including the Leap Motion built into a custom designed tablet. The software is closed-source and based on a subscription model of $20 / month as well as an initial purchase price. The UNI appears to be similar to the idea expressed in the opening chapter – its closed nature and hence limited expandability may hinder its success, however. With no proof of concept or other form of demo available it is difficult to truly gauge how effective the final product will be.

**Chapter 3**

# Design

## 3.1 A

A

**Chapter 4**

# Implementation

## 4.1 A

A

**Chapter 5**

# The System in Operation

## 5.1 A

A

**Chapter 6**

# Testing and Evaluation

## 6.1 A

A

**Chapter 7**

# Conclusions

## 7.1 A

A

**Chapter 8**

# References

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

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