Module Interface Specification for Mechatronics

Team #20, OpenASL
Robert Zhu zhul49
Zifan Meng mengz17
Jiahui Chen chenj194
Kelvin Huynh huynhk12
Runze Zhu zhur25
Mirza Nafi Hasan hasanm21

January 18, 2023

1 Revision History

Date	Version	Notes
January 18, 2023	1.0	Everyone - Initial MIS Draft

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [give url —SS]

[Also add any additional symbols, abbreviations or acronyms —SS]

Contents

3 Introduction

The following document details the Module Interface Specifications for OpenASL, a device developed with the aim of translating sign language into text-to-speech, with the purpose of helping members of the deaf and mute community communicate with those who do not know sign language.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/kelhuynh/OpenASL/.

4 Notation

[You should describe your notation. You can use what is below as a starting point. —SS]

The structure of the MIS for modules comes from **?**, with the addition that template modules have been adapted from **?**. The mathematical notation comes from Chapter 3 of **?**. For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Mechatronics.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

The specification of Mechatronics uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Mechatronics uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	Video Capture Module
Behaviour-Hiding Module	Text-to-Speech Module Key Point Classification Module - Communicates with ML module with data from coordinate normalization module Training Module - Communicates with ML module to update dataset Coordinate Export Module - Read data from video capture and stores into file Motion Tracking Module - Controller (ties everything together)
Software Decision Module	Video Analysis Module - requires data to be used Machine Learning Module Coordinate Normalization Module

Table 1: Module Hierarchy

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table **??**.

7 MIS of Motion Tracking Module (M1)

7.1 Module

motionTrack

7.2 Uses

Video Capture, Coordinate Normalization, Coordinate Export, Video Analysis, Keypoint Classification, TTS

7.3 Syntax

7.3.1 Exported Constants

Name	In	Out	Exceptions
results	image	Object	-
hand_landmarks	-	Tuple of tuples	-
handedness	-	R	-

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Environment Variables

f - file variable for coordinate export purposes

7.4.3 Assumptions

None

7.4.4 Access Routine Semantics

motionTrack():

- output: video consisting of overlay for hand gesture classification into ASL

- exception: exc := cv2.error

8 MIS of Coordinate Normalization Module (M2)

8.1 Module

Coordinate Normalization

8.2 Uses

Video Capture

8.3 Syntax

8.3.1 Exported Constants

Name	In	Out	Exceptions
pre_processed_landmark_list	landmark_list	Tuple of tuples	-

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

None

8.4.3 Assumptions

None

8.4.4 Access Routine Semantics

pre_process_landmark(landmark_list):

- output: tuple of 20 tuples consisting of x and y coordinates for each hand joint
- exception: exc := ListIndexOutofBounds

8.4.5 Local Functions

- __calc_landmark_list

9 MIS of Coordinate Export Module (M3)

9.1 Module

Coordinate Export

9.2 Uses

Coordinate Normalization

9.3 Syntax

9.3.1 Exported Constants

Name	In	Out	Exceptions
keypoint.csv	-	File containing normalized coordinates	-

9.4 Semantics

9.4.1 State Variables

None

9.4.2 Environment Variables

None

9.4.3 Assumptions

None

9.4.4 Local Functions

- __make_csv

10 MIS of Video Capture Module (M4)

10.1 Module

Video Capture

10.2 Uses

None

10.3 Syntax

10.3.1 Exported Constants

Name	In	Out	Exceptions
success	-	R	-
image	-	Object	-

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Environment Variables

success - indicates that the camera input is initialized for use

10.4.3 Assumptions

There is an available camera connected to the system

11 MIS of Video Analysis Module (M5)

11.1 Module

Video Analysis

11.2 Uses

Coordinate Normalization, Video Capture

11.3 Syntax

11.3.1 Exported Access Programs

Name	In	Out	Exceptions
draw_bounding_rect	-	image	-
draw_landmarks	-	image	-
draw_info_text	-	image	-

11.4 Semantics

11.4.1 State Variables

None

11.4.2 Environment Variables

None

11.4.3 Assumptions

None

11.4.4 Access Routine Semantics

draw_bounding_rect(self, use_brect, image, brect):

- output: image with overlaid bounding rectangle around hand

- exception: exc := None

draw_landmarks(self, image, landmark_point):

- output: image with overlaid hand joints and connections

- exception: exc := None

draw_info_text(self, image, brect, handedness, hand_sign_text):

- output: image with overlaid classifier label

- exception: exc := None

11.4.5 Local Functions

- __calc_bounding_rect

12 Appendix

[Extra information if required —SS]