1 Evaluation of QuantumLeap: Tasks

1.1 Configure QuantumLeap

1.1.1 Initial State

The QuantumLeap framework is running in the background and its UI is opened on the "Overview" page.

1.1.2 Steps

1. Go to the "Sensor(s)" page and select the "Leap Motion Controller" module. Rename it as "lmc" and save the modifications.
2. Go to the "Static dataset(s)" page and select the "Leap Motion Dataset Loader" module. Select the "Simple dataset (static)" dataset and give it the "lmc" identifier. Select 100 templates per class. Save the modifications.
3. Go to the "Static recognizer" page and select the "GPSDa" module. Select all the points of the right hand. Save the modifications.
4. Go to the "Segmenter" page and select the "Sliding window" module. Configure it with one (1) window with a length of 20 frames. Select the right hand palm. Save the modifications.
5. Go to the "Dynamic dataset(s)" page and select the "Leap Motion Dataset Loader" module. Select the "Simple dataset (dynamic)" dataset and give it the "lmc" identifier. Select 8 templates per class. Save the modifications.
6. Go to the "Dynamic recognizer" page and select the "Jackknife" module. Configure it with the right hand palm. Save the modifications.
7. Restart QuantumLeap by toggling the play/pause button in the "Overview" page.

1.2 Add Gesture Recognition to a Small Application

1.2.1 Initial State

Visual Studio Code is opened on the App.js file. Only this file will be modified. The application is running in development mode in a Google Chrome tab. The application refreshes automatically when the modifications to App.js are saved. The QuantumLeap framework is running in the background.

1.2.2 Steps

1. To be able use the QuantumLeap API, first import GestureHandler from the quantumleapjs module.
2. Instantiate a new GestureHandler object without any option in the constructor of the App class. Assign it to a new instance variable of App (*e.g.*, this.gestureHandler). In the next steps, you will use the methods of GestureHandler to implement gesture recognition.
3. In componentDidMount, add a call to the connect method of GestureHandler in order to connect to the QuantumLeap framework after the page has loaded.
4. Make sure to disconnect from the QuantumLeap framework when the page is closed by adding a call to the disconnect method of GestureHandler in componentWillUnmount.
5. Let's add the first gesture. The application will display the "swipe left" image each time a "swipe left" gesture is recognized. In componentDidMount, add an event listener for the "gesture" event that calls this.onLeftSwipe each time "swipe left" is recognized. You should use the addEventListener method of GestureHandler.
6. You may notice that nothing happens when performing a "swipe left" gesture. To fix this, register the gesture to QuantumLeap by adding a call to the registerGestures method of GestureHandler with the type ("dynamic") and name ("swipe left") of the gesture. This will notify QuantumLeap that you want it to recognize the "swipe left" gesture.

7. Now, modify the listener that you added at step 5 to display the "swipe right" image (by calling this.onRightSwipe) each time the corresponding gesture is performed. Add the "swipe right" gesture to the call to registerGestures.

8. Modify the listener again to display the "point index" image (by calling this.onPoint) every time the user points his index.

9. The "point index" gesture is static. You should thus add a new call to the registerGestures method of GestureHandler with the type ("static") and name ("point index") of the gesture.

10. Now, modify the listener to display the "thumb" image (by calling this.onThumb) each time the corresponding gesture is performed. Add the "thumb" gesture to the call to registerGestures with the "point index" gesture.

11. Modify the listener one last time to display the name and type of each gesture when they are recognized. You can do it by adding a call to this.onGesture (with the type and name of gesture as arguments) in the listener.

12. Let's now add a text at the bottom of the screen to show if the application is connected to QuantumLeap. In componentDidMount, use the addEventListener method of GestureHandler to call this.setConnected with "true" every time a "connect" event is emitted.

13. To reset the text when the application is disconnected from QuantumLeap, use the addEventListener method of GestureHandler to call this.setConnected with "false" every time a "disconnect" event is emitted.