# Appendix Data for Research Project:

## Exoskeleton Data Glove With Finger Tracking and Tactile Feedback for VR Environments and Telerobotics Applications

By Jose Martin Dominguez Abanto | Student ID: 220761860
Supervisor: Dr Lorenzo Jamon | QMUL 2023/24

## Appendix 1: MATLAB Code for Kinematics Analysis Workspace

```
%% 3D Visualisation workspace with limitations

syms theta1 theta2 real

% DH parameters

% Format: [a alpha d theta]

% a: link length, alpha: twist angle, d: link offset, theta: joint angle

test_dh = [65     1*pi/6   0    theta1;  % First revolute joint (perpendicular)

          40   0      0    theta2; % Second revolute joint

          0   0      0    0;]

% Convert boundary angles from degrees to radians

theta1_min = deg2rad(-30);  % 30 degrees in radians

theta1_max = deg2rad(30);  % 30 degrees in radians

theta2_min = deg2rad(15);  % 10 degrees in radians

theta2_max = deg2rad(-90);  % 80 degrees in radians

% Parameter ranges with boundary conditions

theta1_range = linspace(theta1_min, theta1_max, 100);  % Restricted range for theta1

theta2_range = linspace(theta2_min, theta2_max, 100);  % Restricted range for theta2

% Create a map for the ranges

test_map = containers.Map({'theta1', 'theta2'}, {theta1_range, theta2_range});

% Workspace plotting function (3D)

plot3dworkspace(test_dh, test_map)

function plot3dworkspace(dh_params, map)

   % Extract the parameter ranges from the map

   theta1_range = map('theta1');

   theta2_range = map('theta2');

   % Initialize arrays to hold the end-effector positions

   X = [];

   Y = [];

   Z = [];

   % Loop over the ranges of theta1 and theta2

   for theta1 = theta1_range

       for theta2 = theta2_range

           % Compute the homogeneous transformation matrix for each joint

           T1 = DH_matrix(65, pi/2, 0, theta1);
```

```
            T2 = DH_matrix(40, 0, 0, theta2);

            % Multiply the transformations to get the end-effector position

            T = T1 * T2;

            % Extract the end-effector position from the transformation matrix

            position = T(1:3, 4);  % Extract the translation part (x, y, z)

            X = [X; position(1)];

            Y = [Y; position(2)];

            Z = [Z; position(3)];

        end

    end

    % Plot the 3D workspace

    figure;

    plot3(X, Y, Z, 'b.');

    xlabel('X');

    ylabel('Y');

    zlabel('Z');

    grid on;

    axis equal;

    title('3D Workspace of 2 DOF Planar Exoskeleton Index Finger');

end

function T = DH_matrix(a, alpha, d, theta)

    % Calculate the DH transformation matrix based on the parameters

    T = [cos(theta) -sin(theta)*cos(alpha)  sin(theta)*sin(alpha) a*cos(theta);

         sin(theta)  cos(theta)*cos(alpha) -cos(theta)*sin(alpha) a*sin(theta);

         0           sin(alpha)             cos(alpha)            d;

         0           0                      0                     1];

end
```

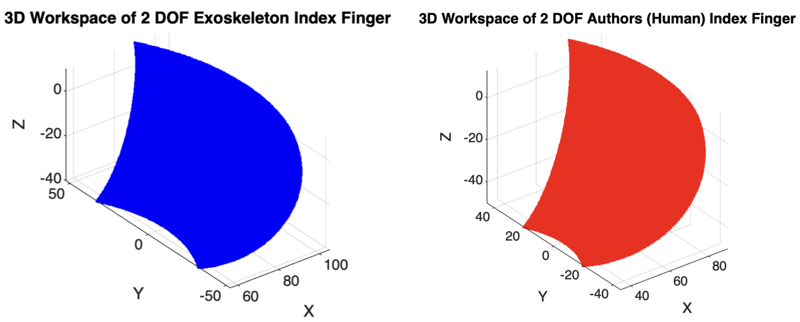


Figure 1. 3D Workspace of 2 DOF Exoskeleton Index Finger.

## Appendix 2: For Finger Tracking and Tactile Feedback

### For Arduino - Hall-Effect Sensor + Coin Vibration Motor

```
void setup() {
 Serial.begin(19200);
 pinMode(A0, INPUT); // For Hall-Effect Sensor
 pinMode(6, OUTPUT); // For Coin Vibration Motor
}

void loop() {
 /// Use it to Calibrate the sensor first: Read (raw data vs Angle) to find linear
regression ///
 // int raw = analogRead(A0);
 // Serial.println(raw);
 // delay(100); // use it to read raw data

 /// With calibration ///
 //int Index_Finger = calibration(analogRead(A0));
 float Index_Finger = calibration(analogRead(A0)); // Probably with Float???
 Serial.println(Index_Finger); // displaying angle position

 /// Vibrate if it reaches -80 degrees ///
 if(Index_Finger <= -80){
   vibrate();
 }
}

/// Function for Calibration with Linear Regression ///
int calibration(int x){
 //x = -1.67 * x + 833; // 0 to 90 degrees v1 - not too bad
 //x = -2.5 * x + -1273; // 15 to -90 degrees ??? v2 - not working
 //x = -2.41 * x + 1262; // -15 to 90 degrees - experiment 4: works good but need to invert
it to negative
 //x = 2.41 * x + -1262; // 15 to -90 degrees - experiment 4: works great :)
 x = 1.25 * x + -704; // 15 to -90 degrees - experiment 5: works great :)
 return x;
}

/// To make motor vibrate ///
void vibrate(){
 digitalWrite(6, HIGH); // ON
 delay(500);
 digitalWrite(6, LOW); // OFF
 delay(500);
}
```

Figure 2.1. Arduino Code for Finger Tracking and Tactile Feedback.

### For Unity - Hand Rotation Controller

```
using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;
using System.IO.Ports; // make sure to select ".Net Framework" in the
"ApiCompatibilityLevel" (Edit>ProjectSettings>Player>)

public class Rotation_4 : MonoBehaviour
{
   public Transform ThumbFinger, Thumb2, IndexFinger, Index2, Index3, MiddleFinger,
Middle2, Middle3, RingFinger, Ring2, Ring3, LittleFinger, Little2, Little3; // Objects For
Rotation
   SerialPort data_stream = new SerialPort("/dev/cu.usbmodem2101", 19200); // Arduino is
connected to /dev/cu.usbmodem2101, with 19200 Baud Rate
   public string receivedstring;

   // Start is called before the first frame update
   void Start()
   {
       data_stream.Open(); // Inititate the Serial Stream
       InvokeRepeating("Serial_Data_Reading", 0f, 0.0002f); // Gives time to read the data
every 0.0001 seconds
   }

   // Update is called once per frame
   void Update()
   {
       int recv_angl = (-1) * Serial_Data_Reading(); // Getting Data from Serial port

       // This is how does eulerAngle work (only for y rotation)
       Debug.Log(recv_angl); // Display values
       // Debug.Log(recv_angl*(-1)); // Display values and inverse it in negative for
accuracy

       // Thumb Finger
       ThumbFinger.transform.eulerAngles = new Vector3(0, recv_angl, 0);
       Thumb2.transform.eulerAngles = new Vector3(0, ThumbFinger.transform.eulerAngles.y +
recv_angl, 0);

       // Index Finger
       IndexFinger.transform.eulerAngles = new Vector3(0, recv_angl, 0);
       Index2.transform.eulerAngles = new Vector3(0, IndexFinger.transform.eulerAngles.y +
recv_angl, 0);
       Index3.transform.eulerAngles = new Vector3(0, Index2.transform.eulerAngles.y +
recv_angl, 0);

       // Middle Finger
       MiddleFinger.transform.eulerAngles = new Vector3(0, recv_angl, 0);
       Middle2.transform.eulerAngles = new Vector3(0, MiddleFinger.transform.eulerAngles.y
+ recv_angl, 0);
       Middle3.transform.eulerAngles = new Vector3(0, Middle2.transform.eulerAngles.y +
recv_angl, 0);
```

```
        // Ring Finger
        RingFinger.transform.eulerAngles = new Vector3(0, recv_angl, 0);
        Ring2.transform.eulerAngles = new Vector3(0, RingFinger.transform.eulerAngles.y +
recv_angl, 0);
        Ring3.transform.eulerAngles = new Vector3(0, Ring2.transform.eulerAngles.y +
recv_angl, 0);

        // Little Finger
        LittleFinger.transform.eulerAngles = new Vector3(0, recv_angl, 0);
        Little2.transform.eulerAngles = new Vector3(0, LittleFinger.transform.eulerAngles.y
+ recv_angl, 0);
        Little3.transform.eulerAngles = new Vector3(0, Little2.transform.eulerAngles.y +
recv_angl, 0);
    }

    // Sync the unity and serial port frequency
    int Serial_Data_Reading()
    {
        // Reading the Serial Data Below------------------
        receivedstring = data_stream.ReadLine();
        //int recv_angl_data = receivedstring;
        int recv_angl_data = Mathf.RoundToInt(float.Parse(receivedstring)); // Round up the
data obtained

        return recv_angl_data;
    }
}
```

Figure 2.2. Unit Code to control VR Hand for Finger Tracking and Tactile Feedback.

## Appendix 3: To Obtain Graph Results Using Arduino and Python

### <u>Arduino Code:</u>

```
unsigned long previousMillis = 0; // To store the last time the data was sent
const long interval = 1000; // Interval in milliseconds (e.g., 1000 ms = 1 second)

void setup() {
 Serial.begin(19200);
 pinMode(A0, INPUT); // For Hall-Effect Sensor
 pinMode(6, OUTPUT); // For Coin Vibration Motor
}

void loop() {
 unsigned long currentMillis = millis(); // Get the current time in milliseconds

  if (currentMillis - previousMillis >= interval) {
   previousMillis = currentMillis; // Update the time

   // Read and calibrate the sensor value
   float Index_Finger = calibration(analogRead(A0)); // Probably with Float???
   Serial.print(currentMillis / 1000.0); // Print time in seconds
```

```
   Serial.print(","); // Comma separator for CSV format
   Serial.println(Index_Finger); // Display angle position


  // Vibrate once it reaches -80 degrees
  if (Index_Finger <= -80) {
    vibrate();
  }
  }
}

/// Function for Calibration with Linear Regression ///
int calibration(int x){
 x = 1.25 * x + -704; // 15 to -90 degrees - experiment 5: works great :)
 return x;
}

/// To make motor vibrate ///
void vibrate(){
 digitalWrite(6, HIGH); // ON
 delay(500);
 digitalWrite(6, LOW); // OFF
 delay(500);
}
```

Figure 3.1. Arduino Code to obtain data for Finger Tracking and Tactile Feedback.

## <u>Python to Obtain Graph Results:</u>

```python
import serial
import time
import pandas as pd
import csv

# Replace 'COM3' with the appropriate port for your system
ser = serial.Serial('/dev/cu.usbmodem2101', 19200)

time.sleep(2)  # Wait for the serial connection to initialize

data = []
start_time = time.time()

while True:
   try:
       line = ser.readline().decode('utf-8').strip()
       print(line)  # Print the data for debugging

       # Split the line into time and Index_Finger values
       parts = line.split(',')
       if len(parts) == 2:
           elapsed_time = float(parts[0])
           index_finger_value = float(parts[1])
```

```
            data.append([elapsed_time, index_finger_value])

        # Stop after 60 seconds (you can adjust this)
        if time.time() - start_time > 60:
            break
    except KeyboardInterrupt:
        break  # Allow manual interruption with Ctrl+C

ser.close()

# Save the data to a CSV file
df = pd.DataFrame(data, columns=["Time (s)", "Index_Finger"])
df.to_csv("Index_Finger_Data.csv", index=False)

print("Data saved to Index_Finger_Data.csv")
```

Figure 3.2. Python Code to obtain data to Plot Graph Results.