ARL-TN-0699 ● SEP 2015

US Army Research Laboratory

Hand Gesture Data Collection Procedure Using a Myo Armband for Machine Learning

by Michael Lee and Nikhil Rao

ARL-TN-0699 ● SEP 2015

US Army Research Laboratory

Hand Gesture Data Collection Procedure Using a Myo Armband for Machine Learning

by Michael Lee and Nikhil Rao
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* Sep 2015	**2. REPORT TYPE** Final
3. DATES COVERED (From - To)	
4. TITLE AND SUBTITLE Hand Gesture Data Collection Procedure Using a Myo Armband for Machine Learning	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Michael Lee and Nikhil Rao	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-CII-B 2800 Powder Mill Road Adelphi, MD 20783-1138	**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TN-0699
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
In 2015, the Battlefield Information Processing Branch investigated using machine learning (ML) to identify military hand gestures. A Naïve Bayes model was trained and created as an initial test. Naturally, the capability of the model relies on the quality and quantity of the training data. Therefore, the data collection is an important and necessary step in building a ML classifier. This report describes the procedure used to collect arm gesture movement data using a Myo armband. The source code for this work is included as an Appendix.

15. SUBJECT TERMS
Myo, Machine Learning, Classifier, Data Collection

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Michael Lee
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	UU	20	**19b. TELEPHONE NUMBER (Include area code)** (301) 394-5608

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

One area of focus in the Battlefield Information Processing Branch is in human and information interaction, which addresses getting the right information to the right people at the right time to achieve the right objective. This work is divided into 2 areas: 1) information access and 2) information-based collaboration and negotiation. An example of work in the information access initiative is using machine learning (ML) technology to model human gestures to exploit human physiological information. One aspect of this study is collecting arm gesture data from volunteers wearing an arm motion-tracking sensor to train a classifier (e.g., Naïve Bayes) capable of recognizing similar arm gestures. The data collection procedure is described in the following sections.

2. Myo Armband Setup

The wearable sensor used for this data collection process is the Myo armband (Fig. 1). The Myo armband is a commercially available Bluetooth device worn on the forearm. It is capable of transmitting various positioning and pose information to a target application. Myo detects 5 poses: fingers spread, wave in, wave out, fist, and double tap. Positioning data (e.g., roll, pitch, yaw, and acceleration) are accessed using an accompanying software development kit (SDK) from the Myo manufacturer. Further details and specifications are available on the company website: https://www.myo.com/techspecs.

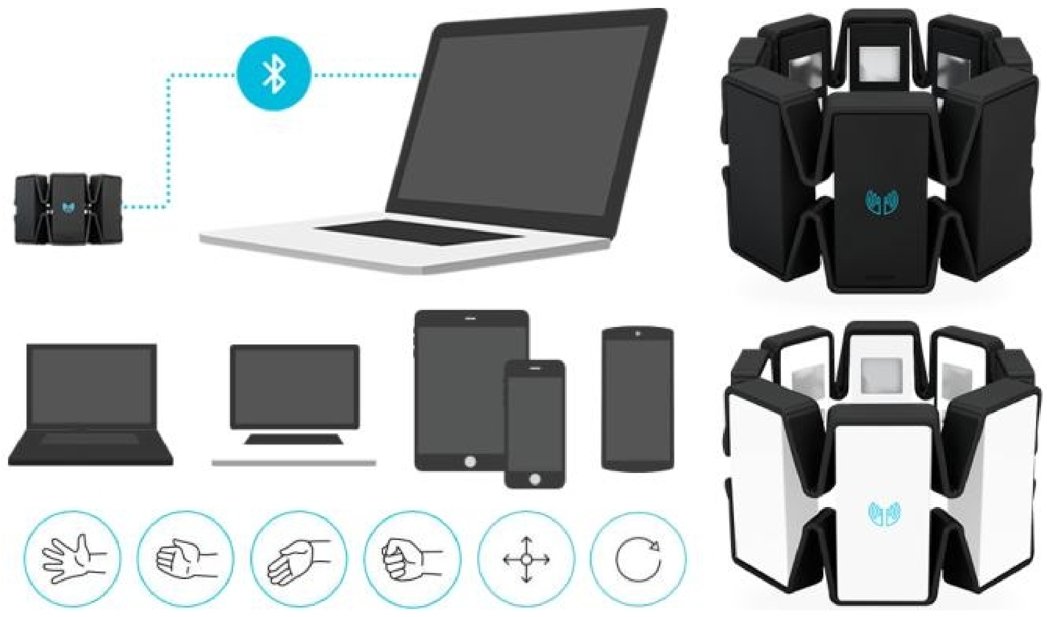


Fig. 1 Myo armband

The initial step in setup for the Myo armband data collection is to install the Myo software on the computer that will be used for the data collection. The software installation process is quick and straightforward. This software allows the user to configure the Myo armband, add custom scripts, add applications, create new profiles, etc. In order to synchronize the Myo, follow the onscreen instructions to verify the Myo is connected to the computer. If the software is installed correctly, a new Myo icon is added in the notification area (Fig. 2).

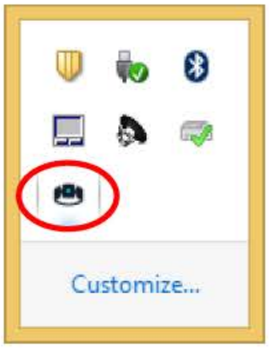
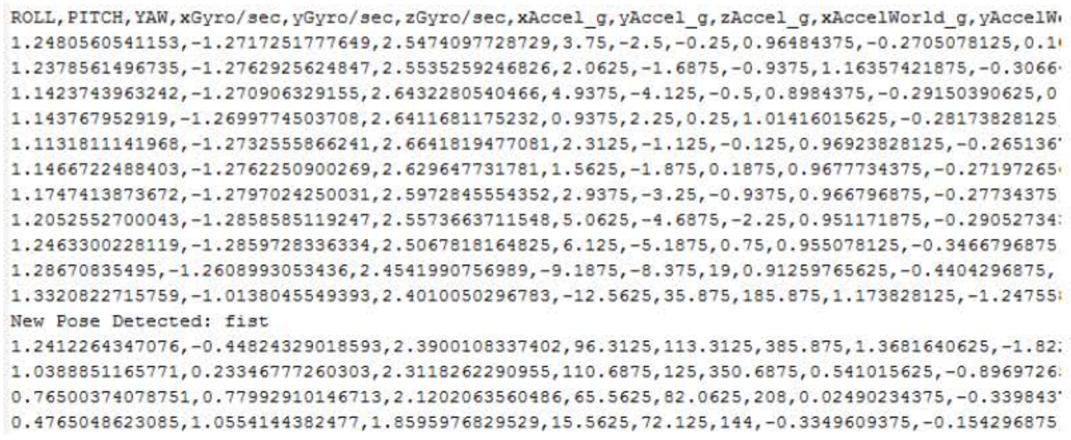


Fig. 2 Myo icon in the Windows notification area

3. Myo Script Setup

There are several ways to develop an application that communicates with the Myo armband. Myo supports development in multiple platforms (e.g., Windows, iOS, Android, etc.) and many languages (e.g., Java, C++, C#, Lua, etc.). For the data collection process, a custom Lua script was written to print the position information to the screen. This script is the critical piece to the data collection process because it is the component that displays the volunteer's movement data. Once the script executes, it waits for the volunteer to perform a double-tap pose. At that point, the script prints 10 sets of position information per second until another double-tap pose is detected to stop the script. Each set contains 16 comma-separated attributes: ROLL, PITCH, YAW, xGyro/sec, yGyro/sec, zGyro/sec, xAccel_g, yAccel_g, zAccel_g, xAccelWorld_g, yAccelWorld_g, zAccelWorld_g, X-Direction, xOrientationWorld, yOrientationWorld, and zOrientationWorld. An example of a partial script printout is displayed in Fig. 3. For convenience, the Lua script (PrintArmMovements.lua) is appended at the end of this report in the Appendix.



```
ROLL,PITCH,YAW,xGyro/sec,yGyro/sec,zGyro/sec,xAccel_g,yAccel_g,zAccel_g,xAccelWorld_g,yAccelW
1.2480560541153,-1.2717251777649,2.5474097728729,3.75,-2.5,-0.25,0.96484375,-0.2705078125,0.1
1.2378561496735,-1.2762925624847,2.5535259246826,2.0625,-1.6875,-0.9375,1.16357421875,-0.3066
1.1423743963242,-1.270906329155,2.6432280540466,4.9375,-4.125,-0.5,0.8984375,-0.29150390625,0
1.143767952919,-1.2699774503708,2.6411681175232,0.9375,2.25,0.25,1.01416015625,-0.28173828125
1.1131811141968,-1.2732555866241,2.6641819477081,2.3125,-1.125,-0.125,0.96923828125,-0.265136
1.1466722488403,-1.2762250900269,2.629647731781,1.5625,-1.875,0.1875,0.9677734375,-0.27197265
1.1747413873672,-1.2797024250031,2.5972845554352,2.9375,-3.25,-0.9375,0.966796875,-0.27734375
1.2052552700043,-1.2858585119247,2.5573663711548,5.0625,-4.6875,-2.25,0.951171875,-0.29052734
1.2463300228119,-1.2859728336334,2.5067818164825,6.125,-5.1875,0.75,0.955078125,-0.3466796875
1.28670835495,-1.2608993053436,2.4541990756989,-9.1875,-8.375,19,0.91259765625,-0.4404296875,
1.3320822715759,-1.0138045549393,2.4010050296783,-12.5625,35.875,185.875,1.173828125,-1.24755
New Pose Detected: fist
1.2412264347076,-0.44824329018593,2.3900108337402,96.3125,113.3125,385.875,1.3681640625,-1.82
1.0388851165771,0.23346777260303,2.3118262290955,110.6875,125,350.6875,0.541015625,-0.8969726
0.76500374078751,0.77992910146713,2.1202063560486,65.5625,82.0625,208,0.0249023437 5,-0.339843
0.4765048623085,1.0554144382477,1.8595976829529,15.5625,72.125,144,-0.3349609375,-0.154296875
```

Fig. 3 Position information printed from PrintArmMovements lua script

The Lua script must be registered in the Myo Application Manager before it can be launched. The following instructions describe this process:

1) Locate the Myo script on the computer or copy and paste the Myo script from this document to a text file named "PrintArmMovements.lua".

2) Launch the Application Manager by right-clicking on the Myo icon and selecting Application Manager. The Application Manager displays a list of other Lua scripts currently installed on the computer. Press the "+ Add" button in the Application Manager to launch the Add Connector window (Fig. 4).

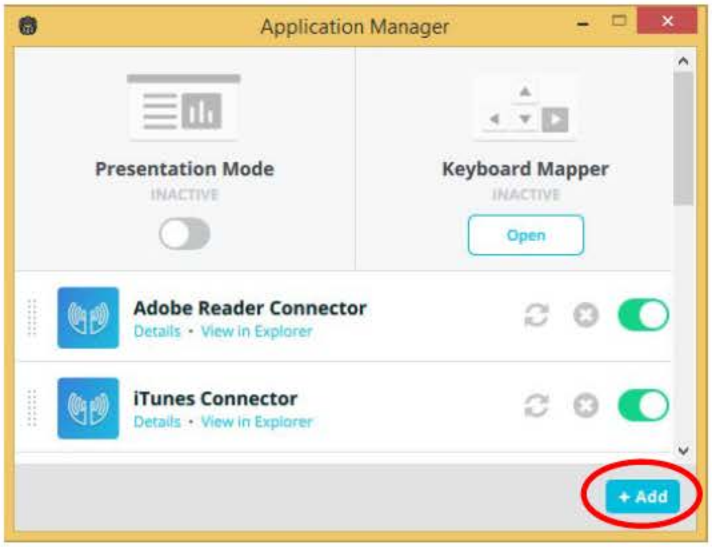
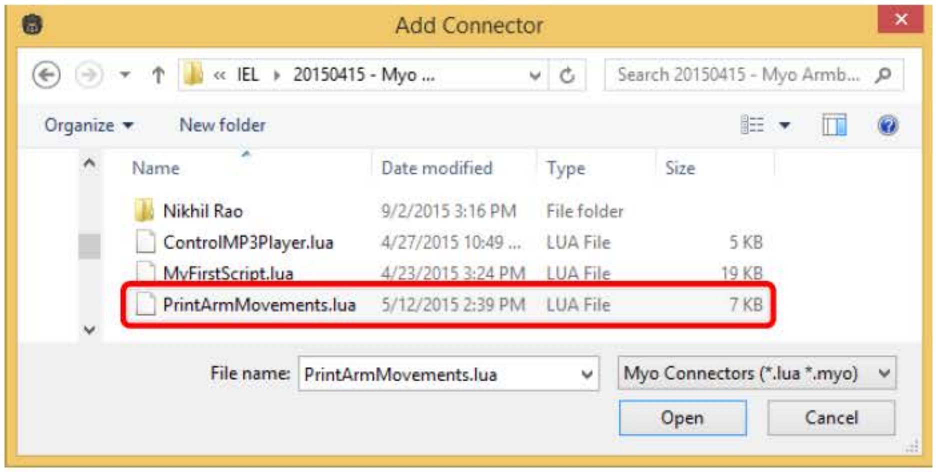


Fig. 4 Myo Application Manager

3) In the Add Connector window, navigate to the location of the PrintArmMovements.lua script. Select the PrintArmMovements.lua script (Fig. 5) and click the “Open” button.



Name	Date modified	Type	Size
Nikhil Rao	9/2/2015 3:16 PM	File folder	
ControlMP3Player.lua	4/27/2015 10:49 ...	LUA File	5 KB
MyFirstScript.lua	4/23/2015 3:24 PM	LUA File	19 KB
PrintArmMovements.lua	5/12/2015 2:39 PM	LUA File	7 KB

File name: PrintArmMovements.lua

Myo Connectors (*.lua *.myo)

Fig. 5 Add Connector window to select the PrintArmMovements.lua script

4) The PrintArmMovements.lua script is now registered with the Myo Application Manager and ready for use. The new script titled “Print Arm Movements” is now included in the list (Fig. 6). The toggle switch on the right indicates whether the script is currently running (indicated by the color green) or not running (indicated by the color gray). Switch the toggle to disable the script until it is needed.

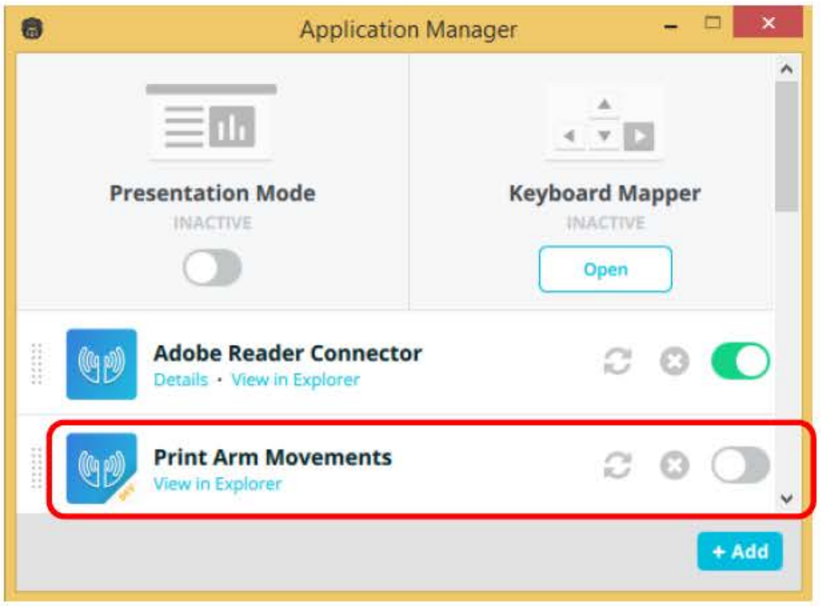


Fig. 6 Print Arm Movement script added to the Myo Application Manager

4. Data Collection Procedure

At this point, the testing environment is ready and the data collection administrator can begin instructing the volunteer subject. The volunteer will perform the steps in this section of the report 10 times for each of the 8 hand gestures (Fig. 7): freeze, rally point, hurry up, down, come, stop, line abreast formation, and vehicle. At the conclusion of the data collection session, there should be 80 data samples collected from each subject. The remainder of this section assumes that the subject is tasked to perform a rally point gesture, but the instructions are applicable for any of the other 7 gestures.

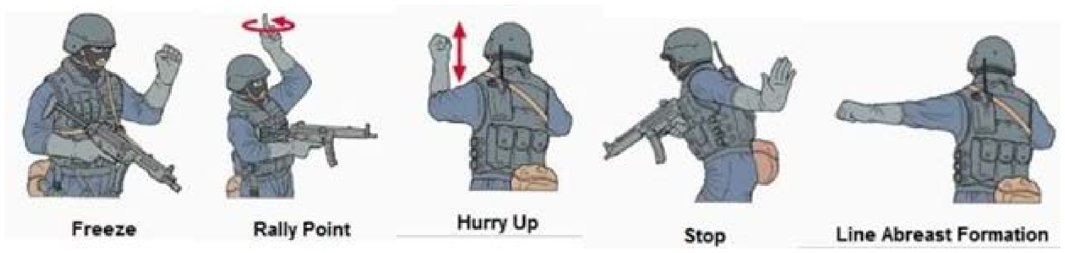


Fig. 7 Examples of military hand gestures used for the data collection

The data collection administrator will execute the following procedure:

1) Review each of the 8 hand gestures with the volunteer. For this data collection, the volunteer will be asked to stand while performing the gestures and exclusively use the right arm. Each gesture will start with the right palm resting on the right side of the body. Verify that the volunteer performs each hand gesture correctly.

2) Provide the Myo armband to the volunteer. The armband should be worn on the right arm with the universal serial bus (USB) outlet pointing toward the wrist. Position the Myo so that the blue glowing logo faces upward along the forearm. Ask the volunteer to perform the synchronize motion and wait until the Myo is warmed up. Verify that the Myo Armband Manager indicates the Myo is currently connected (Fig. 8) before continuing to the next step.

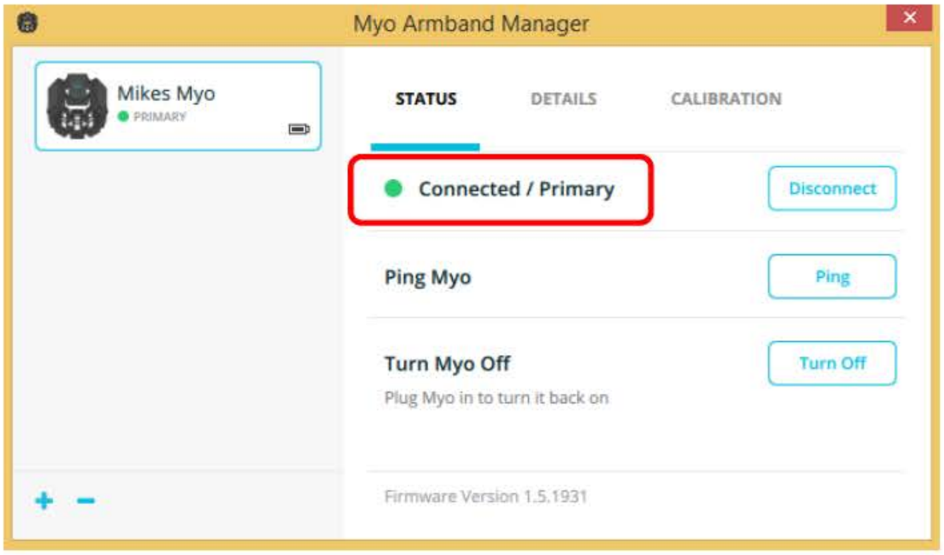
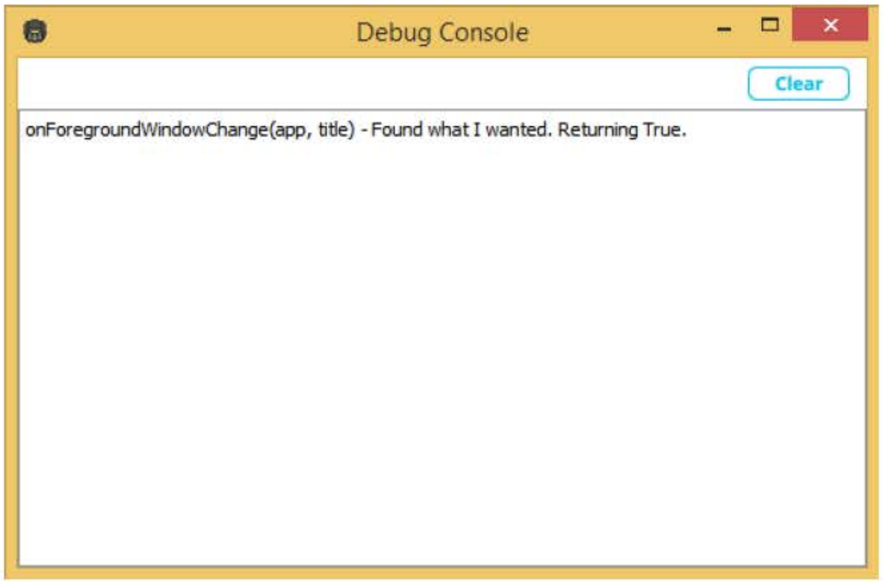
3) 

Fig. 8 Myo Armband Manager indicating the Myo is connected

4) Launch the Myo Application Manager and start the "Print Arm Movement" script by toggling the switch from gray to green. As soon as the script is activated and running, a new window titled "Debug Console" is launched automatically (Fig. 9). The PrintArmMovements.lua script uses the Debug Console window as a conduit to print the arm movement data. Click on the Debug Console window to give focus. It is critical that the Debug Console window maintain focus during the data collection, otherwise the PrintArmMovements.lua script will stop printing the movement data.

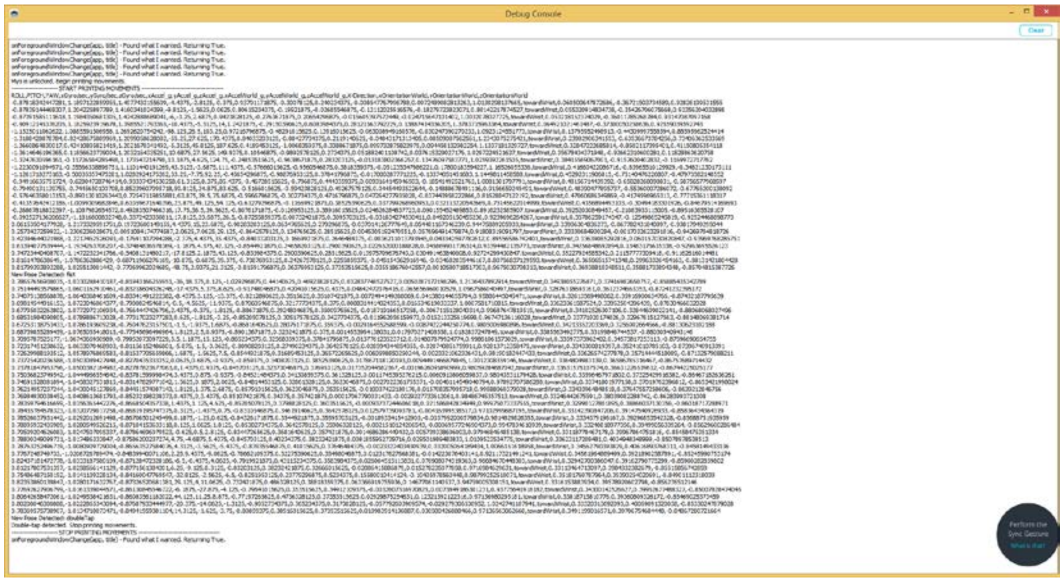


```
onForegroundWindowChange(app, title) - Found what I wanted. Returning True.
```

Fig. 9 Debug Console window launched after the Print Arm Movements script starts

5) Maximize the Debug Console window by clicking on the Maximize button near the top right corner of the window. This step is important, because the Debug Console does not provide a way to buffer text scrolling in its window. In other words, if any text scrolls pass the top of the window, it cannot be retrieved. For this reason, the largest possible window real estate is needed to capture the arm movement data (printing at a rate of 10 lines per second).

6) Instruct the volunteer to form to the starting position (i.e., standing with right palm resting on right thigh), and perform a double-tap pose by double-tapping the right middle finger and thumb together when ready. The PrintArmMovements.lua script will detect a double-tap pose was performed and immediately begin to print the movement data to the Debug Console window (Fig. 10).



```
onForegroundWindowChange(app, title) - Found what I wanted. Returning True.
onForegroundWindowChange(app, title) - Found what I wanted. Returning True.
onForegroundWindowChange(app, title) - Found what I wanted. Returning True.
onForegroundWindowChange(app, title) - Found what I wanted. Returning True.
onForegroundWindowChange(app, title) - Found what I wanted. Returning True.
Myo is unlocked. begin printing movements
---------------- START PRINTING MOVEMENTS ----------------------
ROLL,PITCH,YAW,xGyro/sec,yGyro/sec,zGyro/sec,xAccel_g,yAccel_g,zAccel_g,xAccelWorld_g,yAccelWorld_g,zAccelWorld_g,xDirection,xOrientationWorld,[illegible]
[illegible]
New Pose Detected: fist
[illegible]
New Pose Detected: doubleTap
Double-tap detected. Stop printing movements.
---------------- STOP PRINTING MOVEMENTS ----------------------
onForegroundWindowChange(app, title) - Found what I wanted. Returning True.
```

Fig. 10 The PrintArmMovements lua script printing to the Debug Console window

7) Instruct the volunteer to perform a rally point gesture. Once the gesture is complete, keep and hold the hand in its final position. Even though the gesture is complete and the arm is not moving, the script is still running and will continue to print the data to the Debug Console window.

8) When the scrolling arm movement data approaches the bottom of the Debug Console window, instruct the volunteer to perform another double-tap pose by double-taping the right middle finger and thumb together. The PrintArmMovements.lua script will detect the double-tap pose and immediately exit.

9) Copy the rally point movement data from the Debug Console window to a plain text file assigned to this volunteer. Be sure to document the information about this data instance (e.g., hand gesture performed, date, time, etc.) in the file. This text file is merely a temporary placeholder for the data and it will be formatted during the data processing stage. Continue to append all arm gesture data from this volunteer into this file. The first instance of the rally point gesture data capture is now complete.

10) Instruct the volunteer to perform the second instance of the rally point arm gesture and repeat steps 5 through 8.

5. Conclusion

This report described the steps the Battlefield Information Processing Branch used to collect arm gesture data using the Myo armband. The data collection is the first step in training and building a ML classifier. Subsequent steps involve formatting the data and importing it into a ML library (e.g., Weka). The data processing and model creation procedure is out of the scope of this report and is not described here. Once a ML classifier is built, it will be able to evaluate the arm gesture data and identify the performed gesture. The accuracy of the gesture identification will vary depending on an array of factors. There are many types of classifiers and several ways to generate the classifier. For example, depending on the purpose, a Naïve Bayes classifier can be implemented using data from a single person to achieve high accuracy, or it can be implemented using data from many people to build a model that applies to a wider domain. As an experiment, the Battlefield Information Processing Branch created a simple Naïve Bayes classifier using arm movement data from 1 person, and it demonstrated 70% accuracy from that person. Future plans for this work are to analyze multiple permutations of training the model and study the performance and accuracy of the models.

Appendix. PrintArmMovements.lua

```
scriptId = 'mil.army.arl.20150427'
scriptTitle = "Print Arm Movements"
scriptDetailsUrl = ""

programInFocusNow Name  = ""
programInFocusNow Title = ""

isPrintMovements = false
timeLastPrinted = 0

-- When user unlocks the Myo via double-tap, keep it unlocked and
start printing the movement info.
-- Stop printing the movements and re-engage the lock when the
user double-taps again.
-- Update: 05/12/2015 - Print the pose done by the user (via
onPoseEdge call-back method) while the onPeriodic() function is
printing data.
--                This was added to further help identify the
hand gesture, because the position data alone was not enough.

function onForegroundWindowChange(app, title)
      programInFocusNow_Name  = app
      programInFocusNow Title = title

      if (platform == "Windows" and app == "Myo Connect.exe")
then
            myo.debug("onForegroundWindowChange(app, title) -
Found. Returning True.")
            return true
      else
            return false
      end
end

function onUnlock()
      if (isPrintMovements == false) then
      end

      myo.debug("Myo is unlocked. Begin printing movements.")
      myo.debug("-------------------- START PRINTING MOVEMENTS -
------------------------------------")
      myo.debug("ROLL"        ..    ","   ..    "PITCH"
      ..    ","   ..    "YAW"             ..    ","   ..
      "xGyro/sec,yGyro/sec,zGyro/sec"     ..    ","   ..
      "xAccel g,yAccel g,zAccel g"  ..    ","   ..
      "xAccelWorld g,yAccelWorld g,zAccelWorld g"     ..    ","
      ..    "X-Direction"           ..    ","   ..
      "xOrientationWorld,yOrientationWorld,zOrientationWorld")
      isPrintMovements = true
      myo.unlock("hold")
end

function onPoseEdge(pose, edge)
      pose = conditionallySwapWave(pose)
      if (edge == "on" and pose ~= "rest" and pose ~= "unknown")
then
            myo.debug("New Pose Detected: " .. pose)
      end

      if (pose == "doubleTap" and edge == "on") then
            -- Double-Tap stops printing the Myo movements.
            if (isPrintMovements == true) then  -- Myo was
unlocked and was printing movements. Return locking
            myo.unlock("timed")                 -- Lock Myo in ~1
second
                  myo.debug("Double-tap detected. Stop printing
movements.")
                  myo.debug("-------------------- STOP PRINTING
MOVEMENTS ------------------------------------")
                  isPrintMovements = false
            end
      end
end

function onPeriodic()
      local isDesiredPeriodPassed_Print = false
      local msBetweenPrints = 100               -- 100ms = 10
prints per second.
      local now = myo.getTimeMilliseconds()
      if (now - timeLastPrinted) > msBetweenPrints then
            isDesiredPeriodPassed Print = true        -- It has
been longer than the period we set. So it's okay to print.
      else
            isDesiredPeriodPassed Print = false       -- Enough
time hasn't passed. Wait longer before we print.
      end

      xGyro,yGyro,zGyro = myo.getGyro()
      local printGyro = xGyro .. "," .. yGyro .. "," .. zGyro
      xAccel,yAccel,zAccel = myo.getAccel()
      local printAccel = xAccel .. "," .. yAccel .. "," .. zAccel
      xAccelWorld,yAccelWorld,zAccelWorld = myo.getAccelWorld()
      local printAccelWorld = xAccelWorld .. "," .. yAccelWorld
.. "," .. zAccelWorld
      xOrientationWorld,yOrientationWorld,zOrientationWorld =
myo.getOrientationWorld()
      local printOrientationWorld = xOrientationWorld .. "," ..
yOrientationWorld .. "," .. zOrientationWorld

      if isPrintMovements == true and isDesiredPeriodPassed_Print
== true then
            myo.debug(myo.getRoll() ..    ","   ..
      myo.getPitch()    ..    ","   ..    myo.getYaw()      ..
      ","   ..    printGyro                               ..
      ","   ..    printAccel                              ..
      ","   ..    printAccelWorld
            ..    ","   ..    myo.getXDirection()     ..    ","
      ..    printOrientationWorld)

            timeLastPrinted = myo.getTimeMilliseconds()
      end
```

```
end

function activeAppName()
    return programInFocusNow Title
end

function conditionallySwapWave(pose)
    if myo.getArm() == "left" then
        if pose == "waveIn" then
            pose = "waveOut"
        elseif pose == "waveOut" then
            pose = "waveIn"
        end
    end
    return pose
end
```

1 (PDF)	DEFENSE TECH INFO CTR ATTN DTIC OCA
2 (PDF)	US ARMY RSRCH LAB ATTN IMAL HRA MAIL & RECORDS MGMT ATTN RDRL CIO LL TECHL LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
4 (PDF)	US ARMY RSRCH LAB RDRL CII B M LEE ROBERT P WINKLER LAUREL C SADLER RDRL-CII DEBORAH A WELSH

INTENTIONALLY LEFT BLANK.