MyoMex Quickstart

Table of Contents

Before you begin, please read through READ_ME.txt and follow all steps for setting up the Myo Connect application, Myo SDK, and building the MEX function myo_mex.

Before Using MyoMex

If you decided not to read through READ_ME.txt, let's at least show you the quickest possible way to get started.

```
install_myo_mex; % adds directories to MATLAB search path
% install_myo_mex save % additionally saves the path

sdk_path = 'C:\myo-sdk-win-0.9.0'; % root path to Myo SDK
build_myo_mex(sdk_path); % builds myo_mex
```

```
Evaluating mex command:
 'mex -I"C:\myo-sdk-win-0.9.0\include" -L"C:\myo-sdk-win-0.9.0\lib" -lmyo6

MEX-file 'myo_mex' built successfully!
```

Simplest Usage

This m-code class MyoMex is the intended interface for the myo_mex file that was just built. The simplest lifecycle for a MyoMex object is,

```
m = MyoMex() % instantiate MyoMex (calls into myo_mex init)
m.delete()   % destroy MyoMex (calls into myo_mex delete)
clear m
```

```
m =
  MyoMex with properties:

    streaming_frame_time: 0.0400
     streaming_data_time: 0.0400
            is_streaming: 0
                    time: []
                    quat: []
                    gyro: []
                   accel: []
```

```
                    emg: []
                   pose: []
              pose_rest: []
              pose_fist: []
           pose_wave_in: []
          pose_wave_out: []
    pose_fingers_spread: []
        pose_double_tap: []
           pose_unknown: []
                on_arm: []
            is_unlocked: []
              which_arm: []
              curr_time: 0.0110
                    rot: []
```

Polling Data

The next thing we can do is poll for a single sampling of Myo data. First, instantiate `MyoMex` again.

```
m = MyoMex();
```

Call the method `getData()` get current data from Myo and populate `MyoMex` data properties.

```
m.getData();
```

The most recent data from Myo will be stored in the relevant properties of the `MyoMex` object (i.e. `quat`, `gyro`, `accel`, `emg`, `pose`, etc.).

```
m.time
m.quat
m.gyro
m.accel
m.emg
m.pose
m.pose_rest
m.pose_fist
m.pose_wave_in
m.pose_wave_out
m.pose_fingers_spread
m.pose_double_tap
```

```
        ans =
            0.0390
        ans =
            0.7381   -0.1190   -0.1228    0.6527
        ans =
          340.8750   -1.5000    7.4375
        ans =
            0.0156   -0.5361    0.9722
        ans =
          Columns 1 through 7
                 0   -0.0313    0.0469   -0.5234    0.0156    0.0547    0.0234
          Column 8
```

```
    0.0078
ans =
     6
ans =
     0
ans =
     0
ans =
     0
ans =
     0
ans =
     0
ans =
     0
```

After many subsequent calls to `getData()`, you'll notice that the `data_log` properties of are keeping track of the data received from Myo.

```
for ii = 1:10, m.getData(); end % poll for data ten times

m.time_log
m.accel_log
```

```
ans =
    0.0390
    0.0740
    0.0950
    0.1150
    0.1350
    0.1550
    0.1750
    0.1950
    0.2150
    0.2350
    0.2550
ans =
    0.0156   -0.5361    0.9722
    0.0791    0.0435    0.8174
    0.0605    0.0020    1.1494
    0.0605    0.0020    1.1494
   -0.0225    0.2852    0.9360
    0.0684    0.1279    1.0737
    0.0684    0.1279    1.0737
    0.0107   -0.1274    1.1855
    0.0449    0.0752    0.9429
    0.0449    0.0752    0.9429
    0.0166    0.3926    0.8032
```

As the logs become large in size you may want to clear them using the `clearLogs()` method.

```
m.clearLogs(); % sets *_log properties back to empty

m.time_log
m.accel_log
```

```
ans =
     []
ans =
     []
```

And the most recent data received is always maintained in the `data` properties (it's not cleared with the logs).

```
m.time
m.accel
```

```
ans =
    0.2550
ans =
    0.0166    0.3926    0.8032
```

Streaming Data

In addition to manually polling data, the `myo_mex` interface also supports a state in which it continuously polls Myo for data at an (assumed) constant rate in its own thread. This is referred to as the streaming mode. You set and get the effective sampling rate of this feature by accessing the `streaming_data_time` property. This is the number of seconds between data samples.

```
sample_rate = 30; % desired data sampling rate in Hz
m.streaming_data_time = 1/sample_rate;
m.streaming_data_time % print the value
```

```
Warning: Desired value,  0.0333, rounded to millisecond precision:  0.0330
ans =
    0.0330
```

Notice that the time is restricted to millisecond precision as indicated by the warning message.

As the MEX file polls Myo for data every `streaming_data_time` seconds, the `MyoMex` object will set up a timer that calls into `myo_mex` to fetch the data every `streaming_frame_time` seconds,

```
m.streaming_frame_time = 1/10; % fetch data frames at 10 Hz
```

Begin and end a streaming data session by calling the methods `startStreaming()` and `stopStreaming()`. Since the `MyoMex` object takes care of fetching the data using a `timer` object, you're free to access the command line for other program behavior.

```
m.startStreaming();
% other program behavior
tic;
while toc<5
  fprintf('Number of samples: %5d\n',length(m.time_log));
  pause(1);
end
m.stopStreaming();
```

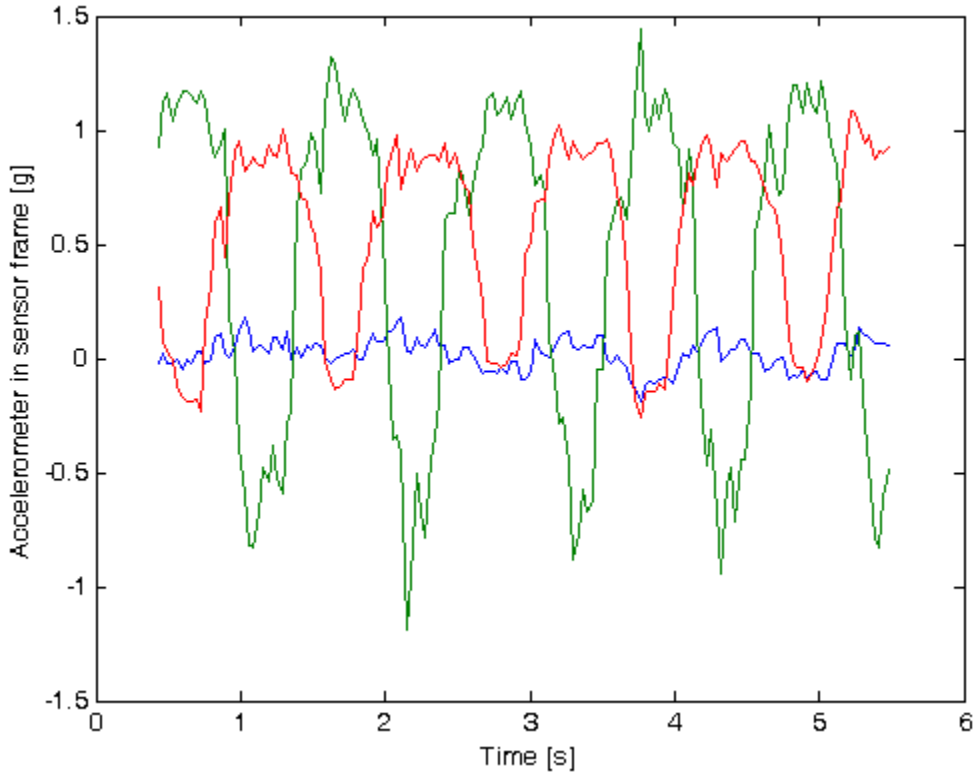
```
Number of samples:     0
Number of samples:    30
```

```
Number of samples:    60
Number of samples:    90
Number of samples:   121
```

You'll notice that as time goes on, the size of the `data_log` properties grows as data is fetched by `MyoMex`. You can read data from these properties while streaming, but most other methods are not functional in this state.

Since we've accumulated some logged data, let's look at a plot of the accelerometer data.

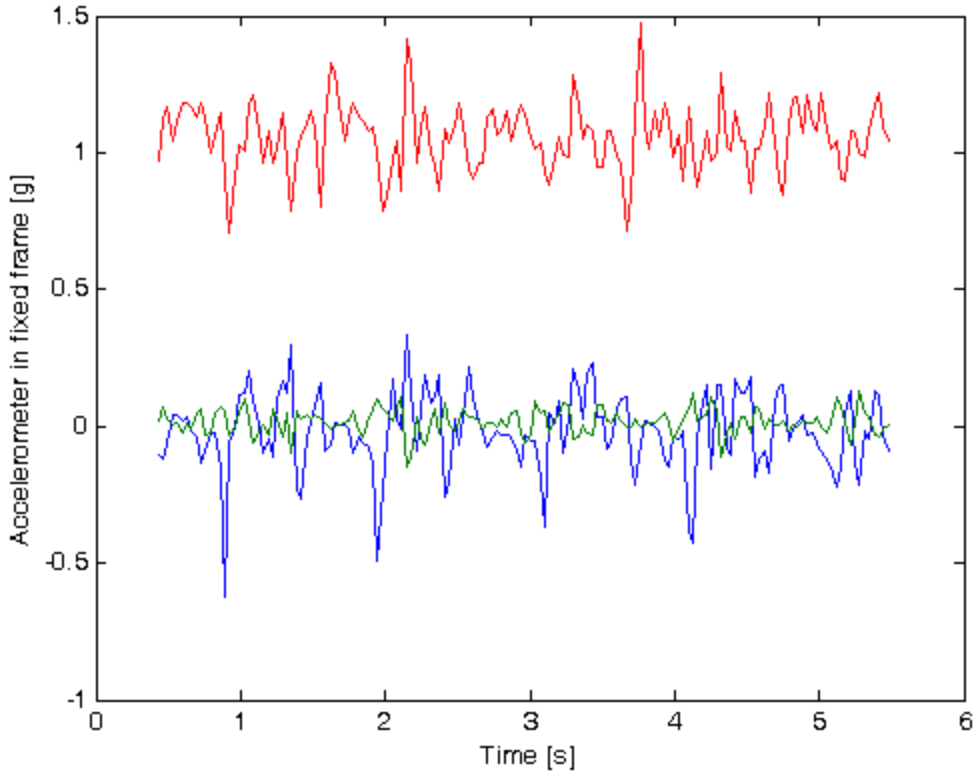
```
figure;
plot(m.time_log,m.accel_log);
ylabel('Accelerometer in sensor frame [g]'); xlabel('Time [s]');
```



We can also transform the `gyro` and `accel` data from sensor frame to fixed frame by getting the dependent property `rot_log`. This is a 3D array in which each 3x3 2D slice is the rotation matrix corresponding to the rows of `quat_log`.

```
R = m.rot_log;
accel_fixed = zeros(size(m.accel_log));
for kk = 1:size(R,3)
  accel_fixed(kk,:) = (R(:,:,kk)*m.accel_log(kk,:)')';
end

figure;
plot(m.time_log,accel_fixed);
ylabel('Accelerometer in fixed frame [g]'); xlabel('Time [s]');
```



Finally, when you're done with `MyoMex`, don't forget to clean up!

```
m.delete;
clear m
```

Finally, use the help documentation and other command line tools for for additional information about `MyoMex`!

```
% properties MyoMex
% methods MyoMex
% help MyoMex
% help MyoMex.time
% help MyoMex.quat
% help MyoMex.getData
% help MyoMex.startStreaming
```

Published with MATLAB® R2013a