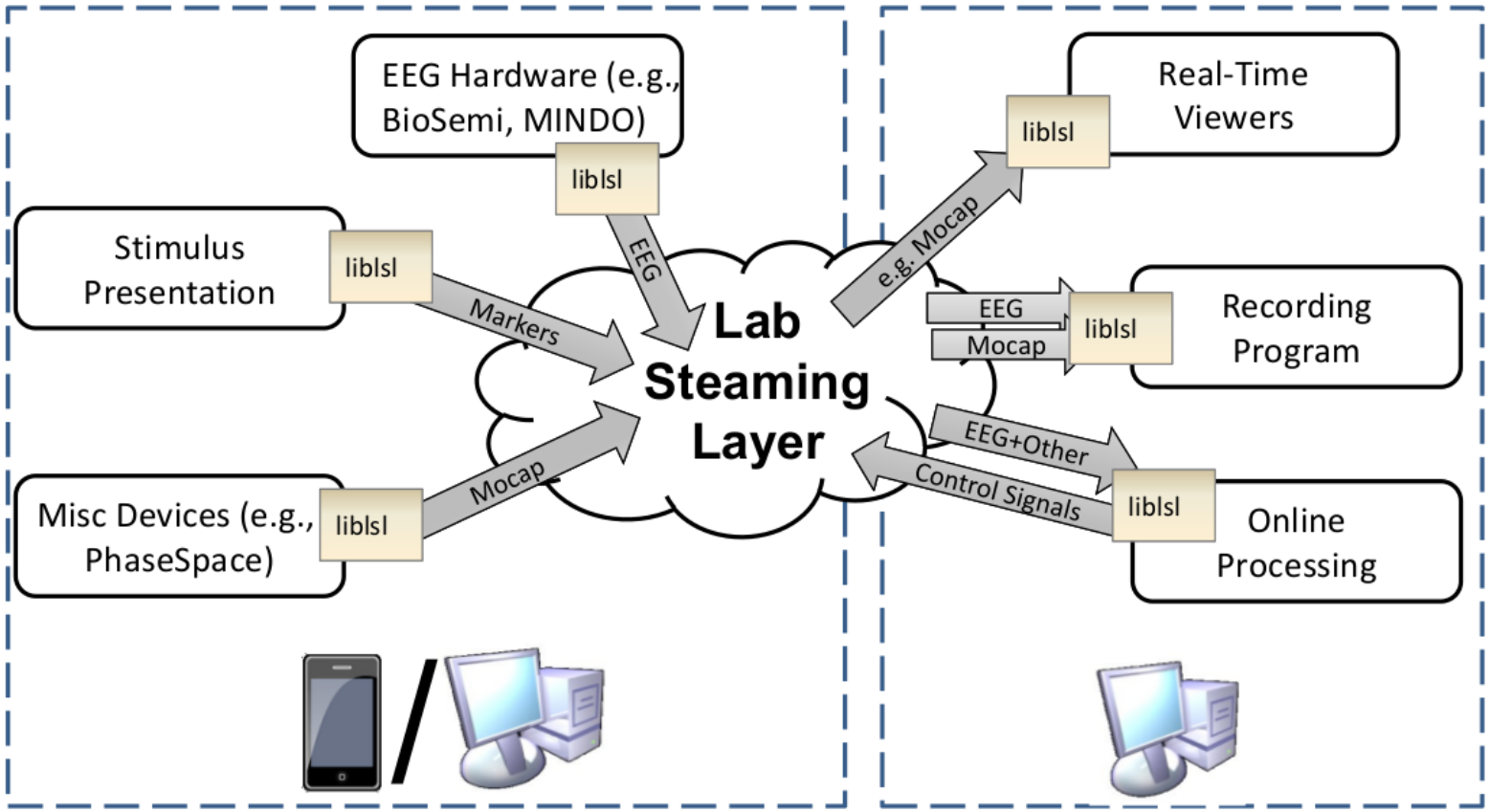
LabStreamingLayer Overivew

David Medine
dmedine@ucsd.edu
Swartz Center for Computational Neuroscience

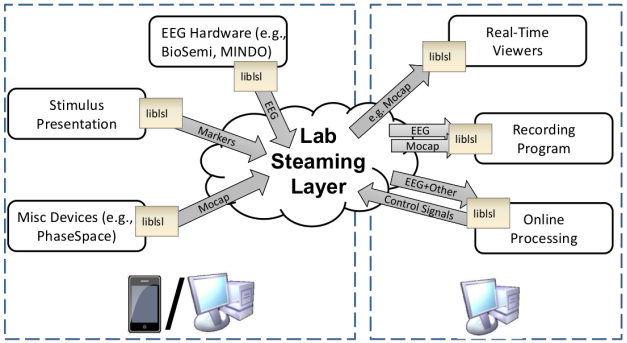
EEGLab Workshop
November 20, 2016

Outline

What is LabStreamingLayer?

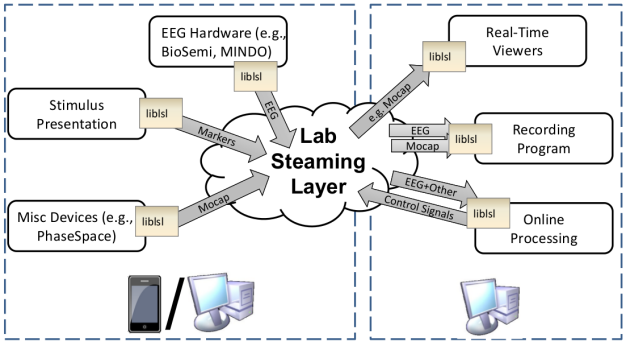


What is LabStreamingLayer?



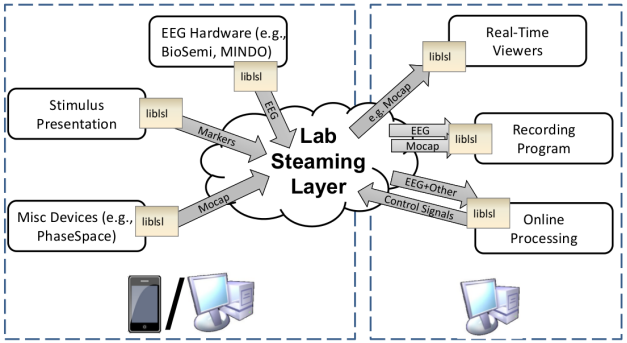
- open source software library

What is LabStreamingLayer?



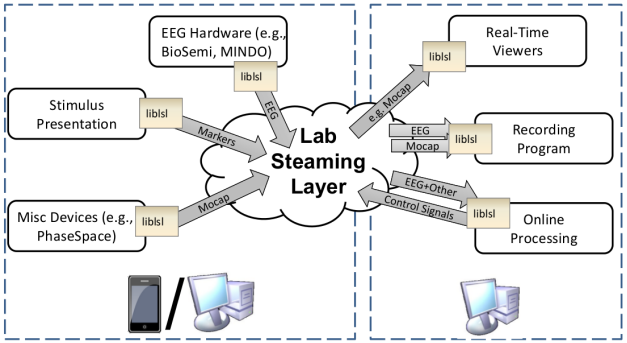
- open source software library
- highly extensible and supports wrappers for numerous languages

What is LabStreamingLayer?



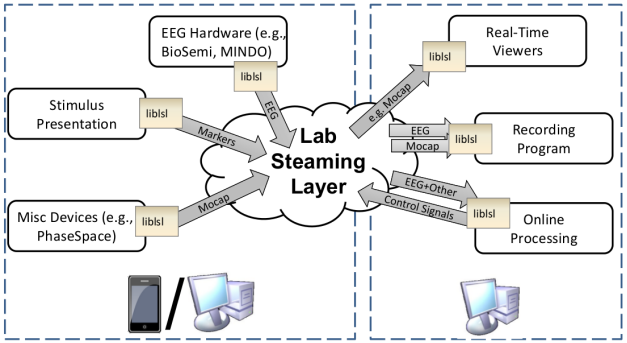
- open source software library
- highly extensible and supports wrappers for numerous languages
- overlay network

What is LabStreamingLayer?



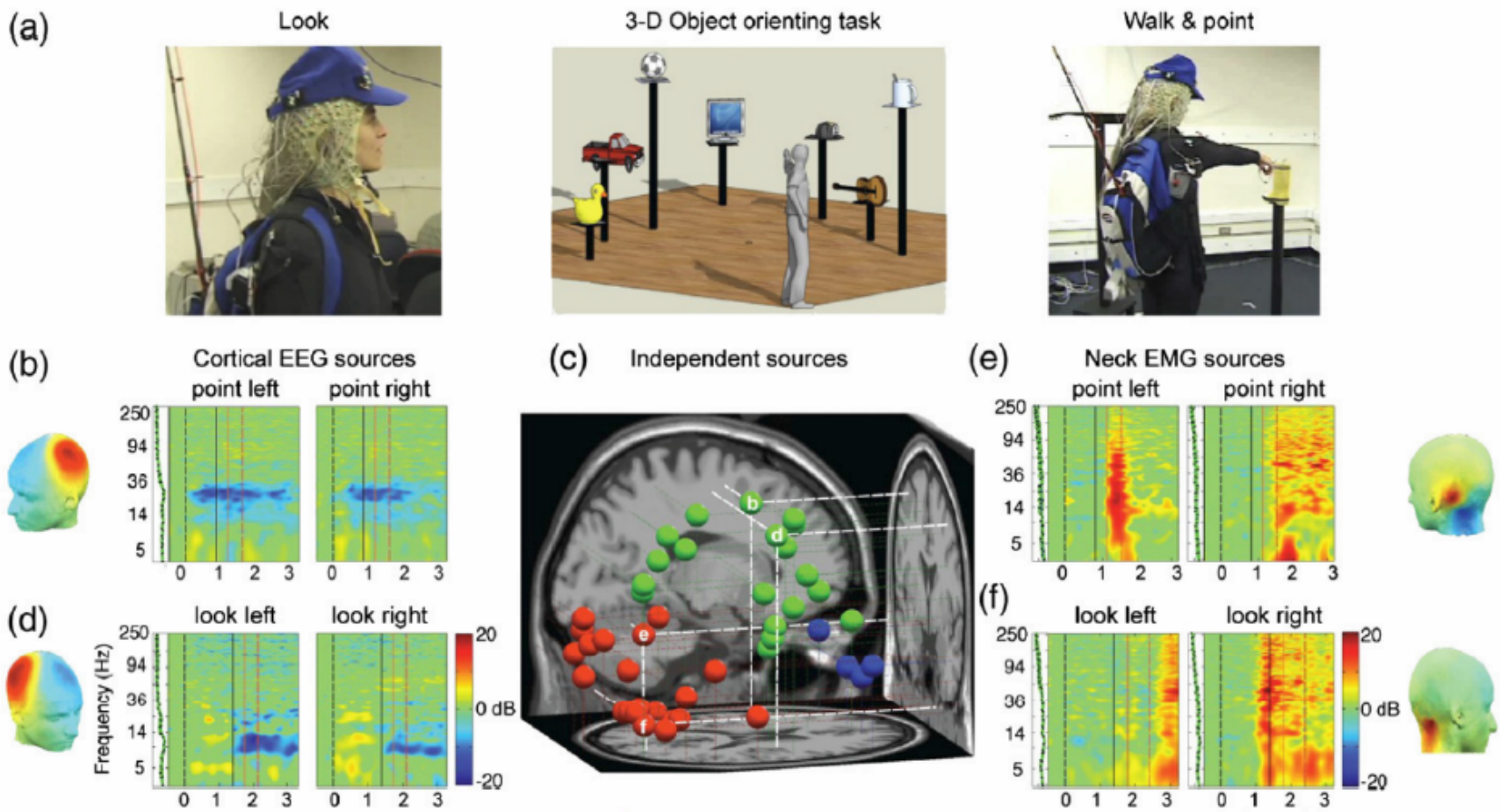
- open source software library
- highly extensible and supports wrappers for numerous languages
- overlay network
- realtime data streaming

What is LabStreamingLayer?

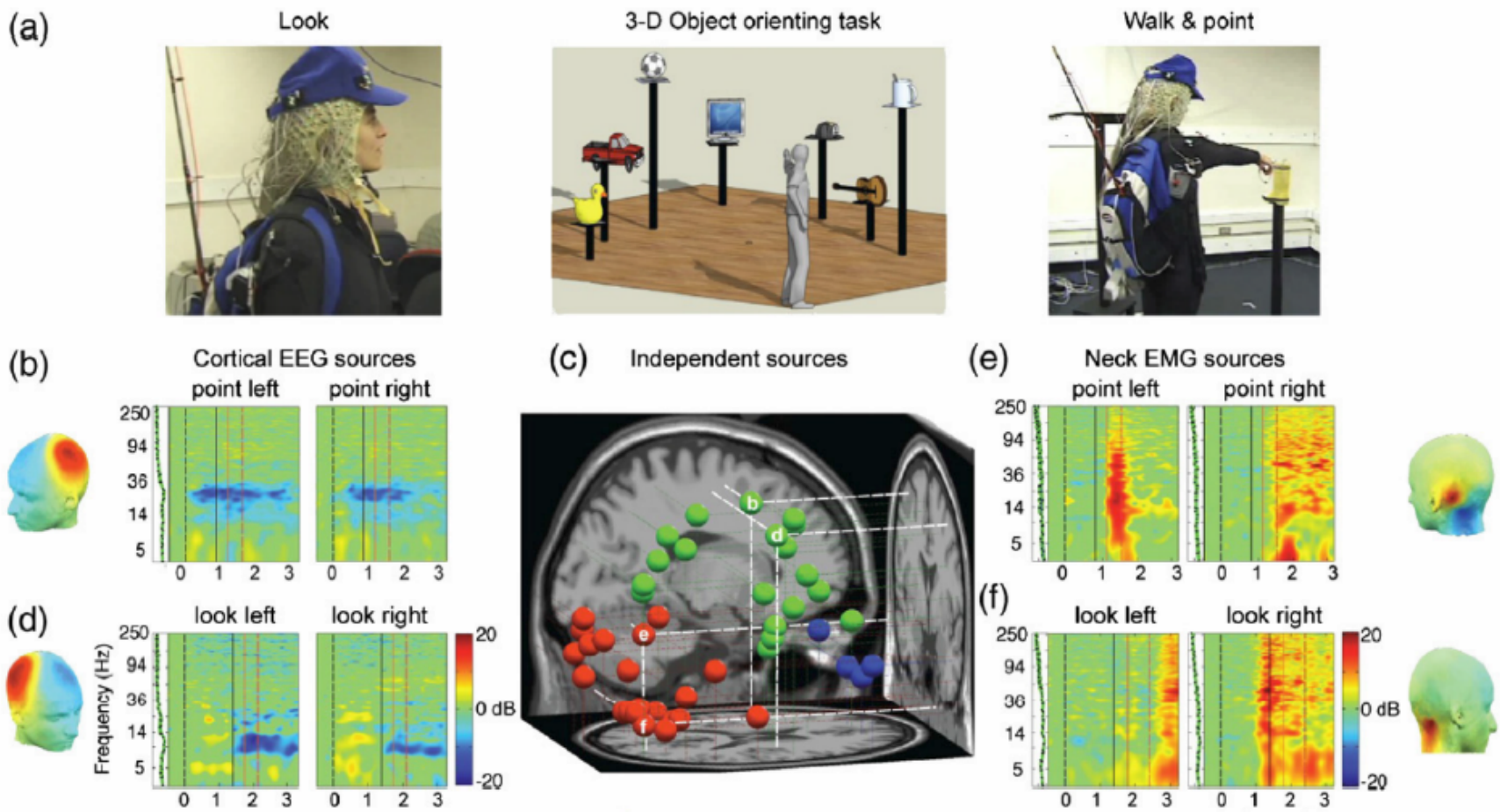


- open source software library
- highly extensible and supports wrappers for numerous languages
- overlay network
- realtime data streaming
- facilitates multimodal data stream synchronization without hardware triggers

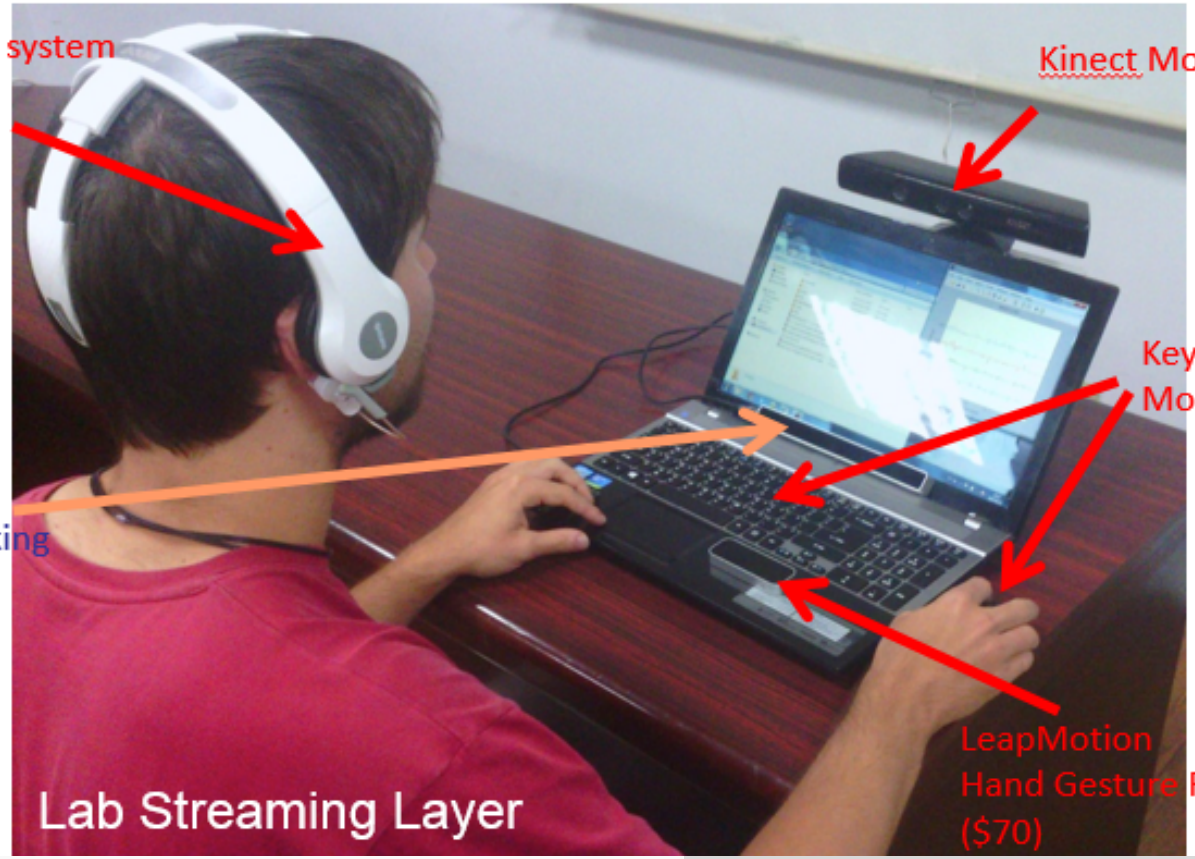
Mobile Brain/Body Imaging (MoBI)



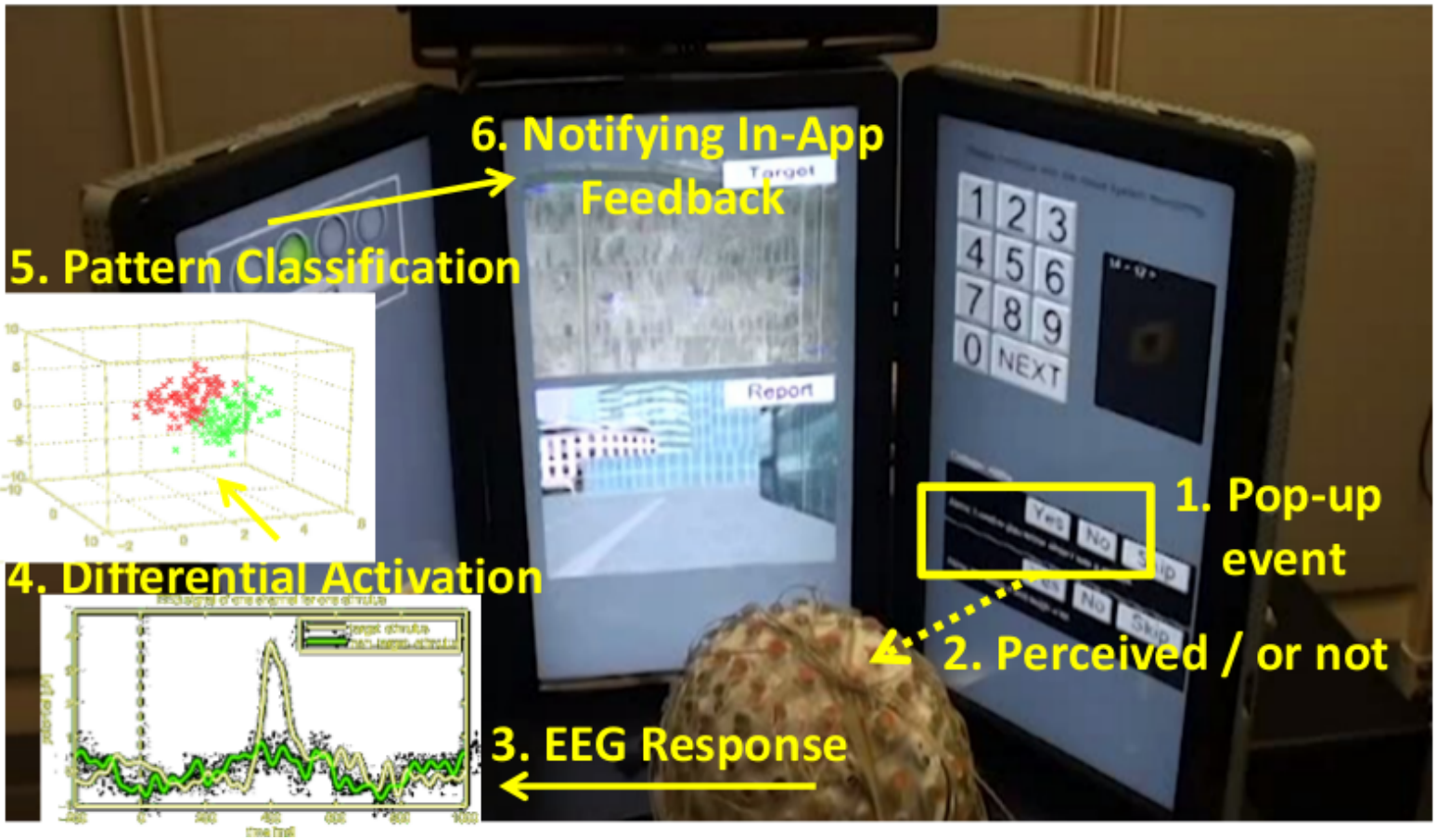
Mobile Brain/Body Imaging (MoBI)



Low-Cost Mobile Brain/Body Imaging (MoBI) Platform



Large Scale Experiments



Synchronizing

EEG and ExG	Full-Body Motion Capture	Eye-Tracking	Human Interface Devices, System State, Etc.

The Challenge: To Synchronize Multiple (Asynchronous) Streams

The Challenge: To Synchronize Multiple (Asynchronous) Streams

- EEG/EMG/EOG – i.e. with BioSemi – 512 Hz

The Challenge: To Synchronize Multiple (Asynchronous) Streams

- EEG/EMG/EOG – i.e. with BioSemi – 512 Hz
- Stimulus Presentation Markers – i.e. with Presentation – random times

The Challenge: To Synchronize Multiple (Asynchronous) Streams

- EEG/EMG/EOG – i.e. with BioSemi – 512 Hz
- Stimulus Presentation Markers – i.e. with Presentation – random times
- Motion Capture – i.e. with PhaseSpace – 400 Hz

The Challenge: To Synchronize Multiple (Asynchronous) Streams

- EEG/EMG/EOG – i.e. with BioSemi – 512 Hz
- Stimulus Presentation Markers – i.e. with Presentation – random times
- Motion Capture – i.e. with PhaseSpace – 400 Hz
- Eyetracking – i.e. with EyeLink – 60 Hz

The Challenge: To Synchronize Multiple (Asynchronous) Streams

- EEG/EMG/EOG – i.e. with BioSemi – 512 Hz
- Stimulus Presentation Markers – i.e. with Presentation – random times
- Motion Capture – i.e. with PhaseSpace – 400 Hz
- Eyetracking – i.e. with EyeLink – 60 Hz
- HCI, GSR, Heartrate, Forceplate, etc. ?? Hz

The Challenge: To Synchronize Multiple (Asynchronous) Streams

The Challenge: To Synchronize Multiple (Asynchronous) Streams

- Attach timestamps to data

The Challenge: To Synchronize Multiple (Asynchronous) Streams

- Attach timestamps to data
- Attach timestamps to record times

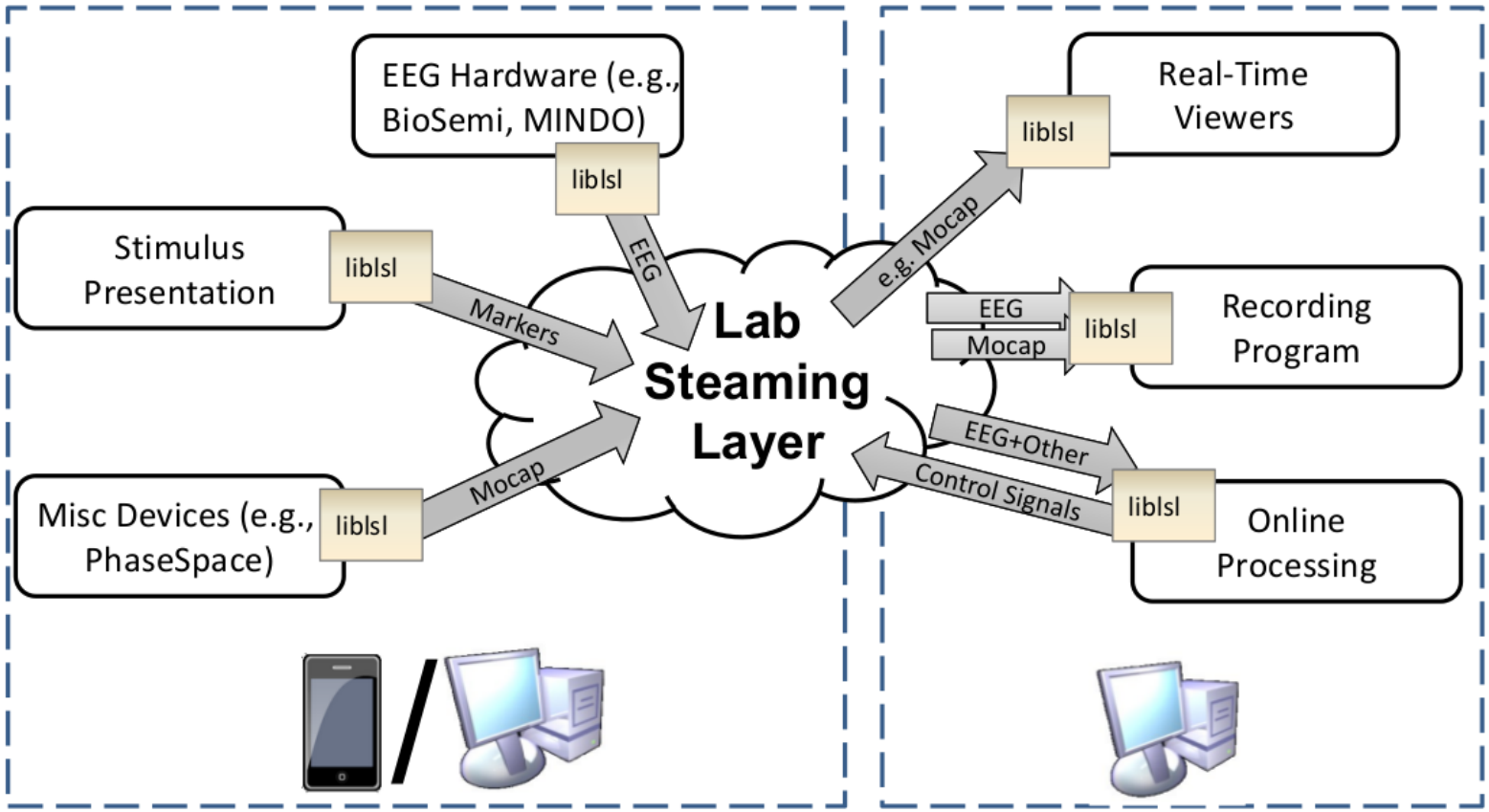
The Challenge: To Synchronize Multiple (Asynchronous) Streams

- Attach timestamps to data
- Attach timestamps to record times
- Measue and test EVERYTHING

The Challenge: To Synchronize Multiple (Asynchronous) Streams

- Attach timestamps to data
- Attach timestamps to record times
- Measue and test EVERYTHING OVER AND OVER

LSL Network View



liblsl

liblsl is an opensource C/C++ library and API for timestamping and streaming multimodal data accross a network.

- Basically provides methods for creating 3 objects:

liblsl

liblsl is an opensource C/C++ library and API for timestamping and streaming multimodal data accross a network.

- Basically provides methods for creating 3 objects:
  - `stream_info` (metadata)

liblsl

liblsl is an opensource C/C++ library and API for timestamping and streaming multimodal data accross a network.

- Basically provides methods for creating 3 objects:
  - `stream_info` (metadata)
  - `stream_outlet` (send data)

liblsl

liblsl is an opensource C/C++ library and API for timestamping and streaming multimodal data accross a network.

- Basically provides methods for creating 3 objects:
  - `stream_info` (metadata)
  - `stream_outlet` (send data)
  - `stream_inlet` (receive data)

stream_info constructor params

Specifies the nature of an lsl stream (essentials)

stream_info constructor params

Specifies the nature of an lsl stream (essentials)

- name ('ActiChamp')
- type ('EEG')
- channel count ('64')
- sample rate('512')
- format ('float32')
- source Id ('8JIAes263D' some such serial number)

stream_info constructor params

Specifies the nature of an lsl stream (essentials)

- name ('ActiChamp')
- type ('EEG')
- channel count ('64')
- sample rate('512')
- format ('float32')
- source Id ('8JIAes263D' some such serial number)

append meta-data

XML based information that adheres to a basic template, but can be extended to contain anything, anyhow

Details – stream_outlet

stream_outlet constructor params

Details – stream_outlet

stream_outlet constructor params

- stream_info object

Details – stream_outlet

stream_outlet constructor params

- stream_info object
- (optionally) chunk size

Details – `stream_outlet`

`stream_outlet` constructor params

- `stream_info` object
- (optionally) chunk size
- (optionally) data buffer size

Details – stream_outlet

stream_outlet constructor params

- stream_info object
- (optionally) chunk size
- (optionally) data buffer size

stream_outlet life-cycle

Details – stream_outlet

stream_outlet constructor params

- stream_info object
- (optionally) chunk size
- (optionally) data buffer size

stream_outlet life-cycle

- once created, a stream_outlet object sits on an open socket (a data stream publisher) and can be pinged by clients to initiate connection (subscription)

Details – stream_outlet

stream_outlet constructor params

- stream_info object
- (optionally) chunk size
- (optionally) data buffer size

stream_outlet life-cycle

- once created, a stream_outlet object sits on an open socket (a data stream publisher) and can be pinged by clients to initiate connection (subscription)
- a loop on a separate thread can stream data or poll for data (push_sample) to output sporadically

Details – stream_outlet

stream_outlet constructor params

- stream_info object
- (optionally) chunk size
- (optionally) data buffer size

stream_outlet life-cycle

- once created, a stream_outlet object sits on an open socket (a data stream publisher) and can be pinged by clients to initiate connection (subscription)
- a loop on a separate thread can stream data or poll for data (push_sample) to output sporadically
- destroy after using...(show some code)

Details – stream_inlet

resolve_stream

Details – `stream_inlet`

`resolve_stream`

- @param prop (‘name’, ‘type’, etc.)
- @param value (‘ActiChamp’, ‘EEG’, etc.)
- @return `std::vector<stream_info>`

Details – `stream_inlet`

`resolve_stream`

- @param prop (‘name’, ‘type’, etc.)
- @param value (‘ActiChamp’, ‘EEG’, etc.)
- @return `std::vector<stream_info>`

`stream_inlet` constructor

Details – `stream_inlet`

`resolve_stream`

- @param prop ('name', 'type', etc.)
- @param value ('ActiChamp', 'EEG', etc.)
- @return `std::vector<stream_info>`

`stream_inlet` constructor

- @param info (one of the `stream_info` objects returned by `resolve_stream`)
- (optional) @params buffer length, chunk length, recover (bool)

stream_inlet life-cycle

stream_inlet life-cycle

- automatically connect to corresponding outlet

Details – `stream_inlet`

`stream_inlet` life-cycle

- automatically connect to corresponding outlet
- launch a 'listen' thread to repeatedly call `pull_sample`

Details – `stream_inlet`

`stream_inlet` life-cycle

- automatically connect to corresponding outlet
- launch a 'listen' thread to repeatedly call `pull_sample`
- if recover=true, automatically respawn if lost

Details – `stream_inlet`

`stream_inlet` life-cycle

- automatically connect to corresponding outlet
- launch a 'listen' thread to repeatedly call `pull_sample`
- if recover=true, automatically respawn if lost
- destroy when all done (show some code)

31 Apps to interface LSL with various devices

31 Apps to interface LSL with various devices

- Link to LSL and vendor libraries (if any)

31 Apps to interface LSL with various devices

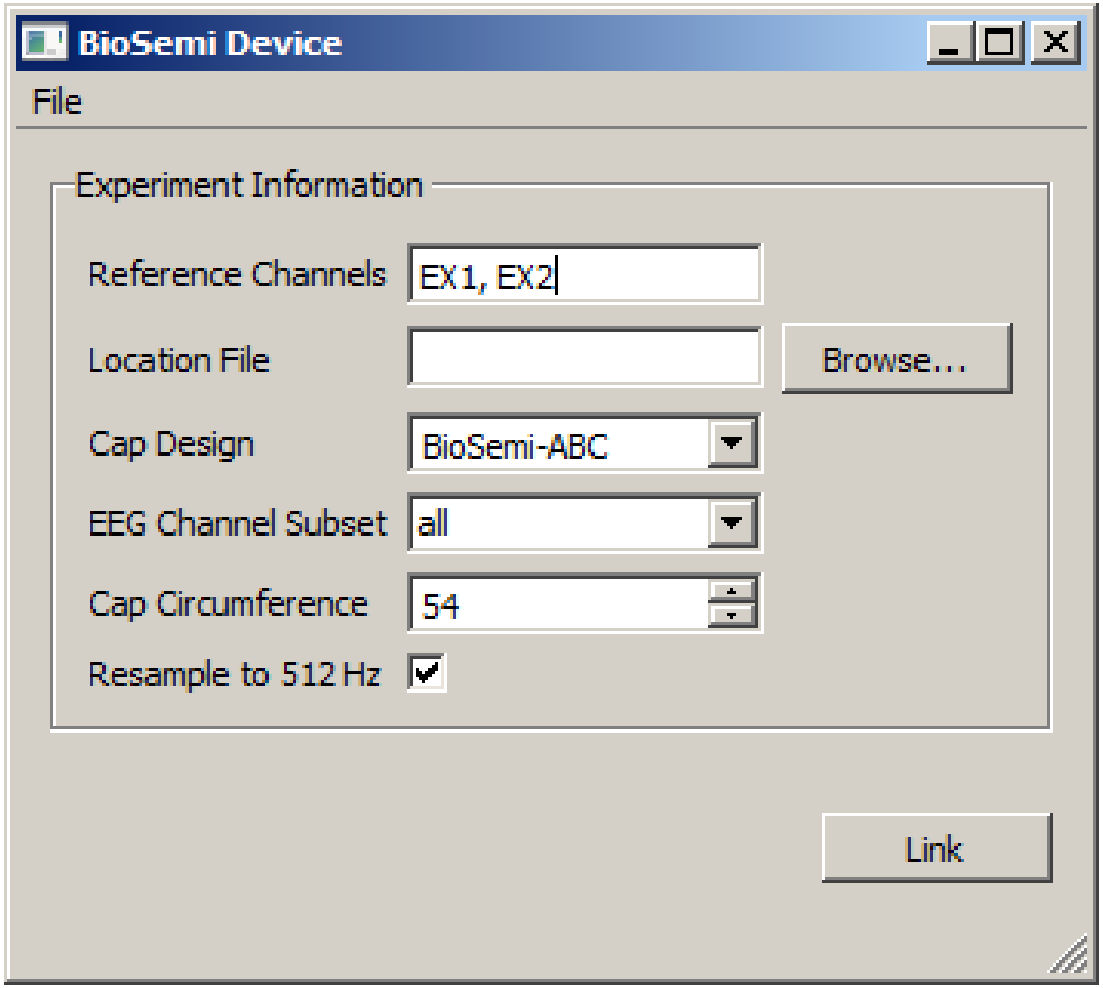
- Link to LSL and vendor libraries (if any)
- Setup ‘streaminfo’ and metadata (xml)

31 Apps to interface LSL with various devices

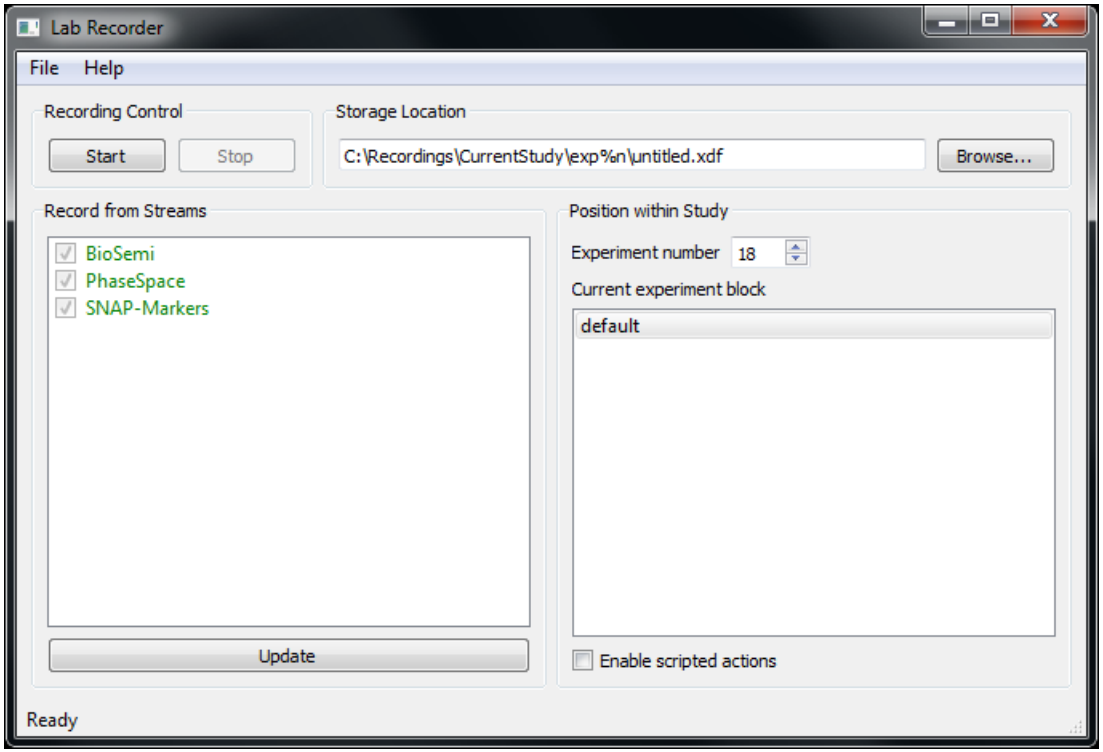
- Link to LSL and vendor libraries (if any)
- Setup ‘streaminfo’ and metadata (xml)
- Open LSL ‘outlet’

31 Apps to interface LSL with various devices

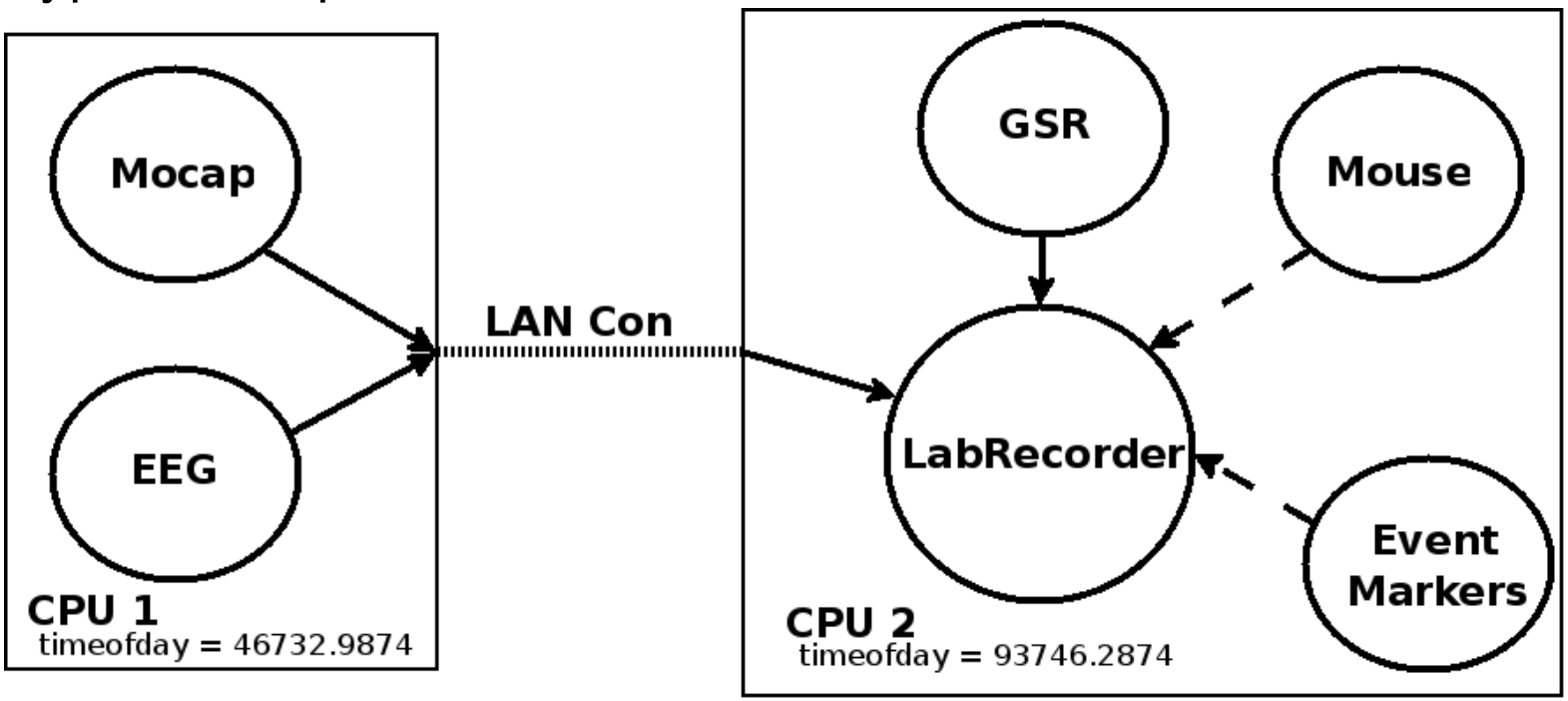
- Link to LSL and vendor libraries (if any)
- Setup ‘streaminfo’ and metadata (xml)
- Open LSL ‘outlet’
- Launch listener thread to ‘push’ LSL data as it arrives


BioSemi Device
File
Experiment Information
Reference Channels
EX1, EX2
Location File
Browse...
Cap Design
BioSemi-ABC
EEG Channel Subset
all
Cap Circumference
54
Resample to 512 Hz
Link

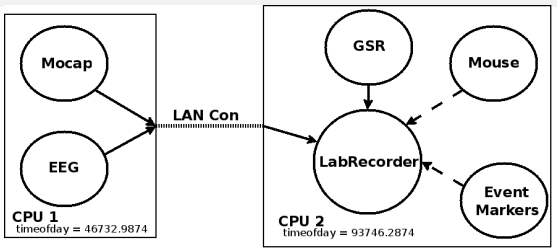
Overview – Apps



Typical Setup:



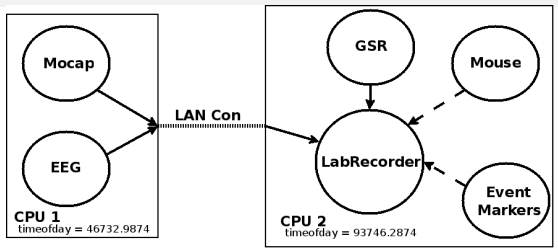
Overview – LabRecorder and load_xdf.m



LabRecorder

load_xdf.m

Overview – LabRecorder and load_xdf.m

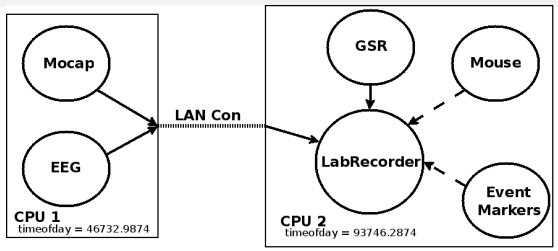


LabRecorder

- Locate streams, write metadata to ouput file header

load_xdf.m

Overview – LabRecorder and load_xdf.m

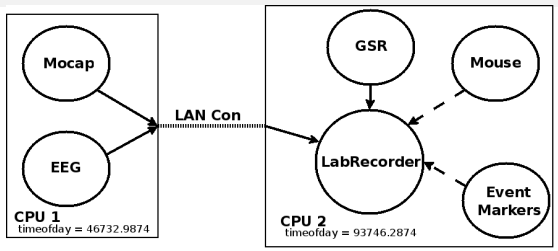


LabRecorder

- Locate streams, write metadata to ouput file header
- Read streams,write data to ouput file

load_xdf.m

Overview – LabRecorder and load_xdf.m

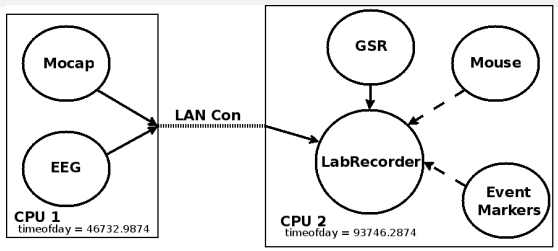


LabRecorder

- Locate streams, write metadata to ouput file header
- Read streams,write data to ouput file
- Periodically check clock offsets (NTP), write offsets to output file footer

load_xdf.m

Overview – LabRecorder and load_xdf.m

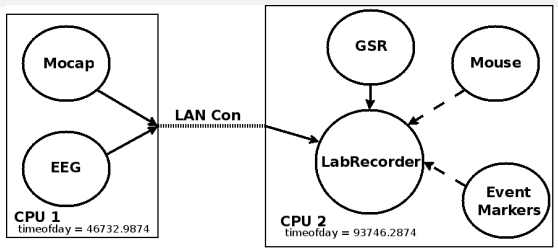


LabRecorder

- Locate streams, write metadata to ouput file header
- Read streams,write data to ouput file
- Periodically check clock offsets (NTP), write offsets to output file footer

load_xdf.m

Overview – LabRecorder and load_xdf.m



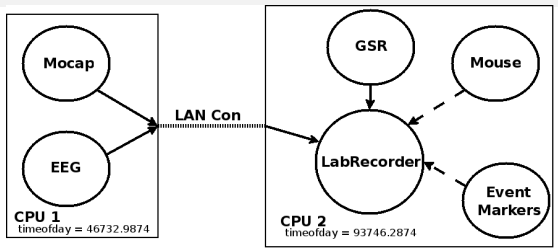
LabRecorder

- Locate streams, write metadata to ouput file header
- Read streams,write data to ouput file
- Periodically check clock offsets (NTP), write offsets to output file footer

load_xdf.m

- Organize meta-data into a datastructure

Overview – LabRecorder and load_xdf.m



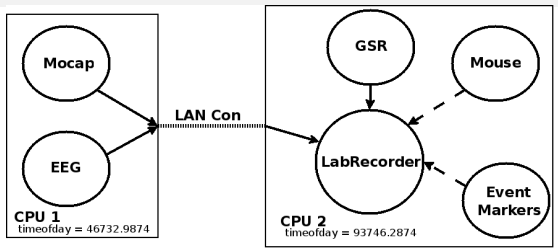
LabRecorder

- Locate streams, write metadata to ouput file header
- Read streams,write data to ouput file
- Periodically check clock offsets (NTP), write offsets to output file footer

load_xdf.m

- Organize meta-data into a datastructure
- Linearize non-sporadic timestamps

Overview – LabRecorder and load_xdf.m



LabRecorder

- Locate streams, write metadata to ouput file header
- Read streams,write data to ouput file
- Periodically check clock offsets (NTP), write offsets to output file footer

load_xdf.m

- Organize meta-data into a datastructure
- Linearize non-sporadic timestamps
- Use clock offsets and known latencies to synchronize data streams

1: Calculate Clock Offsets

1: Calculate Clock Offsets

- Record clock offsets peridiocally during data acquisition using clock filter algorithm (NTP) using `get_time_of_day` to record sets of 4 timestamps in rapid succession:

1: Calculate Clock Offsets

- Record clock offsets peridiocally during data acquisition using clock filter algorithm (NTP) using `get_time_of_day` to record sets of 4 timestamps in rapid succession:
  - send from inlet to outlet (t0)

1: Calculate Clock Offsets

- Record clock offsets peridiocally during data acquisition using clock filter algorithm (NTP) using `get_time_of_day` to record sets of 4 timestamps in rapid succession:
  - send from inlet to outlet (t0)
  - receive from inlet at outlet (t1)

1: Calculate Clock Offsets

- Record clock offsets peridiocally during data acquisition using clock filter algorithm (NTP) using `get_time_of_day` to record sets of 4 timestamps in rapid succession:
  - send from inlet to outlet (t0)
  - receive from inlet at outlet (t1)
  - immediately send from outlet to inlet (t3)

1: Calculate Clock Offsets

- Record clock offsets peridiocally during data acquisition using clock filter algorithm (NTP) using `get_time_of_day` to record sets of 4 timestamps in rapid succession:
  - send from inlet to outlet (t0)
  - receive from inlet at outlet (t1)
  - immediately send from outlet to inlet (t3)
  - receove from outlet at inlet (t4)

1: Calculate Clock Offsets

- Record clock offsets peridiocally during data acquisition using clock filter algorithm (NTP) using `get_time_of_day` to record sets of 4 timestamps in rapid succession:
  - send from inlet to outlet (t0)
  - receive from inlet at outlet (t1)
  - immediately send from outlet to inlet (t3)
  - receove from outlet at inlet (t4)
- round trip time (RTT) = (t3-t0) - (t2-t1)

1: Calculate Clock Offsets

- Record clock offsets peridiocally during data acquisition using clock filter algorithm (NTP) using `get_time_of_day` to record sets of 4 timestamps in rapid succession:
  - send from inlet to outlet (t0)
  - receive from inlet at outlet (t1)
  - immediately send from outlet to inlet (t3)
  - receove from outlet at inlet (t4)
- round trip time (RTT) = (t3-t0) - (t2-t1)
- clock offset (OFS) = ((t1-t0) + (t2-t3))/2 (for lowest RTT)

2: Map drifting clock values and fit

2: Map drifting clock values and fit

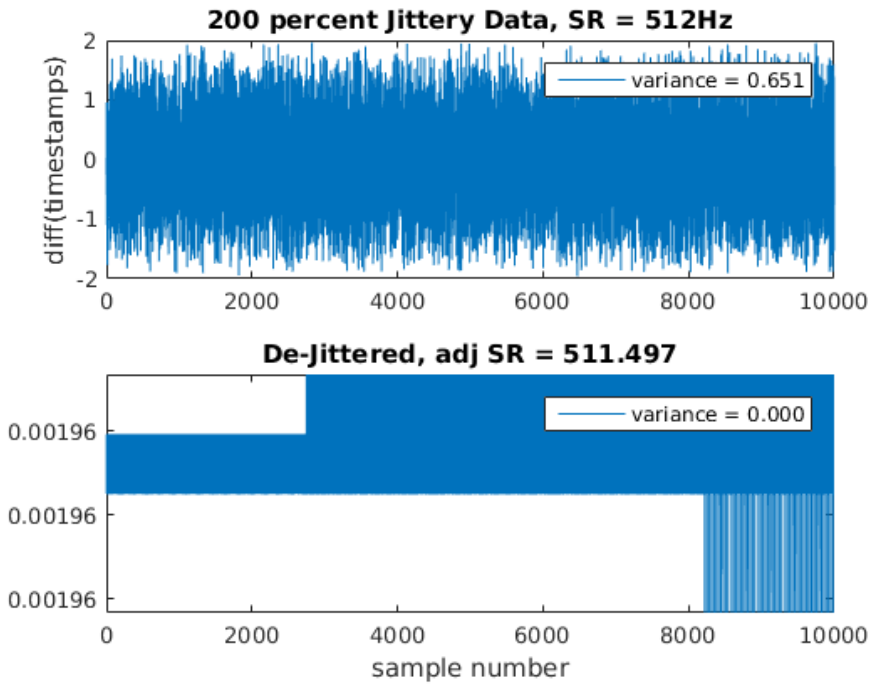
- This is normally done post-hoc in `load_xdf.m` using a fitting procedure. Each map is calculated using an ADMM method incorporating the Huber loss function (`http://www.stanford.edu/~boyd/papers/distr_opt_stat_learning_admm.html`)
- Each map is a DC offset and a slope adjustment ($y_n = ax_n + b$) for each intermittent OFS record point (default is 5s between queries).

2: Map drifting clock values and fit

- This is normally done post-hoc in `load_xdf.m` using a fitting procedure. Each map is calculated using an ADMM method incorporating the Huber loss function (`http://www.stanford.edu/~boyd/papers/distr_opt_stat_learning_admm.html`)
- Each map is a DC offset and a slope adjustment ($y_n = ax_n + b$) for each intermittent OFS record point (default is 5s between queries).
- The latest version of LSL has methods for doing this online, but it is not yet validated

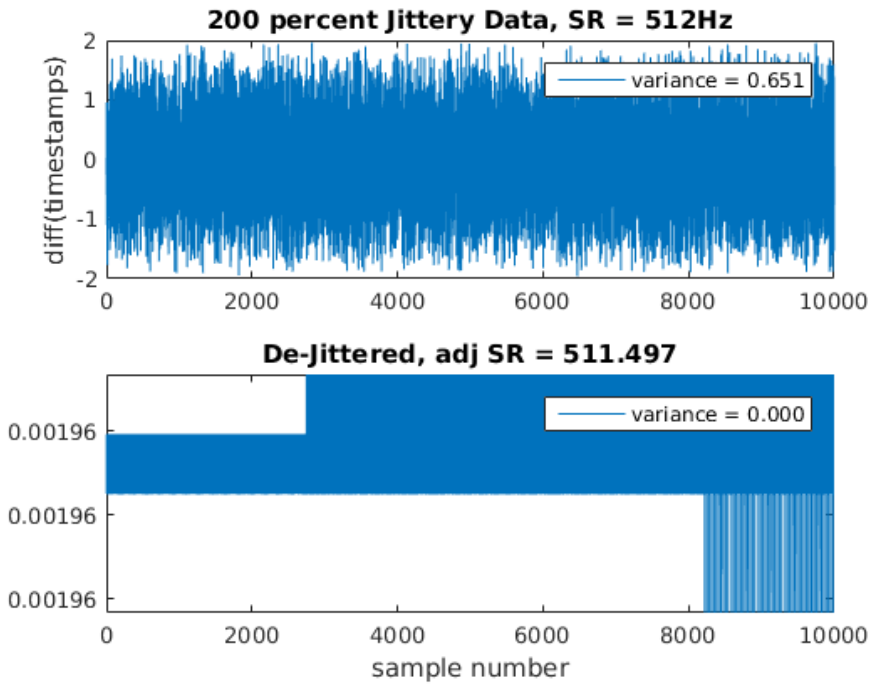
3: Linearize/De-Jitter the timestamps (if appropriate)

- Simple linear regression is very robust:



3: Linearize/De-Jitter the timestamps (if appropriate)

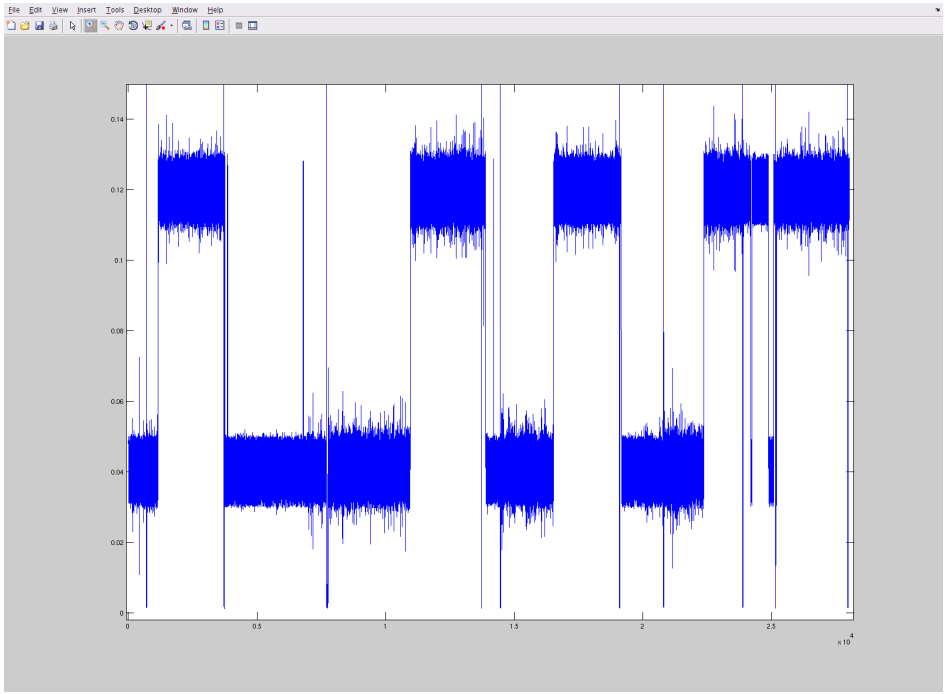
- Simple linear regression is very robust:



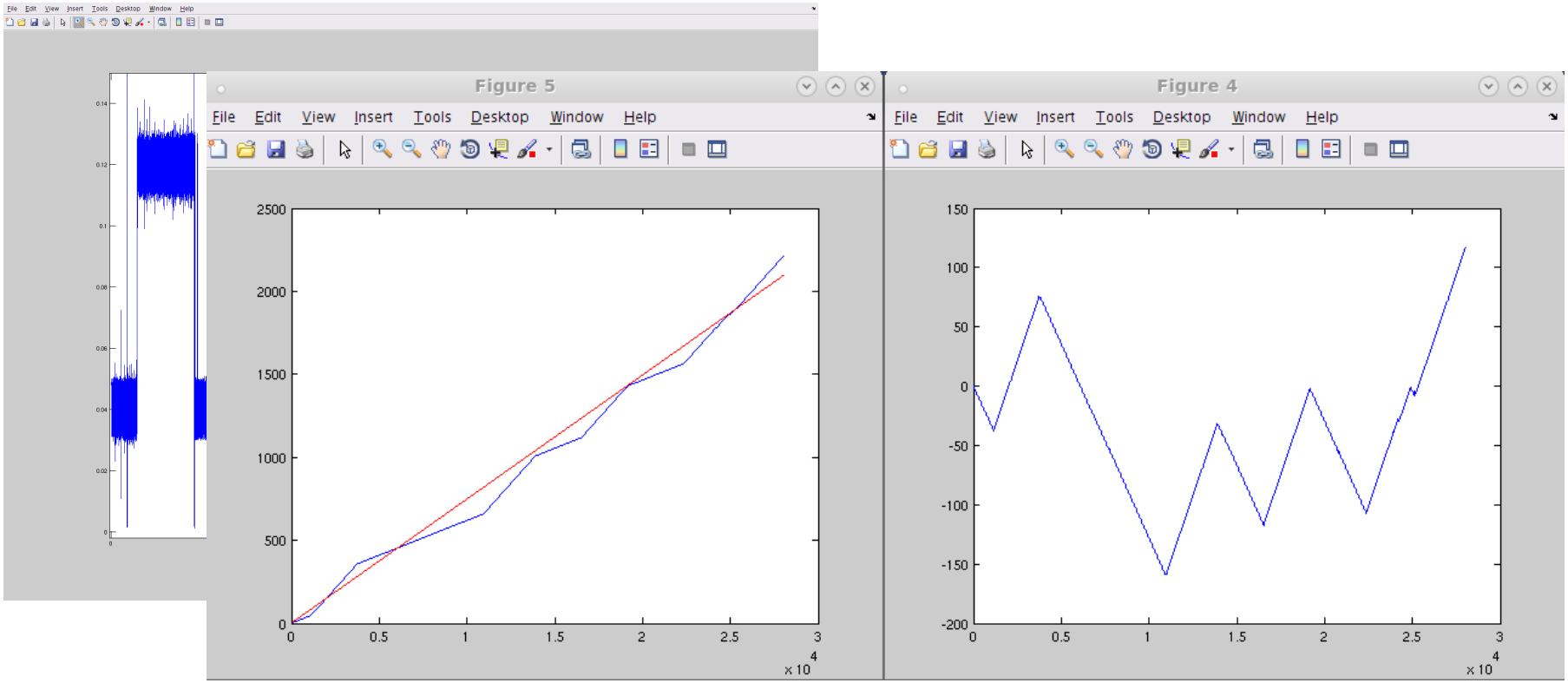
This will fail if the sampling rate changes!

This is bad:

This is bad:


File Edit View Insert Tools Desktop Window Help
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
0.5
1
1.5
2
2.5
x 10^4

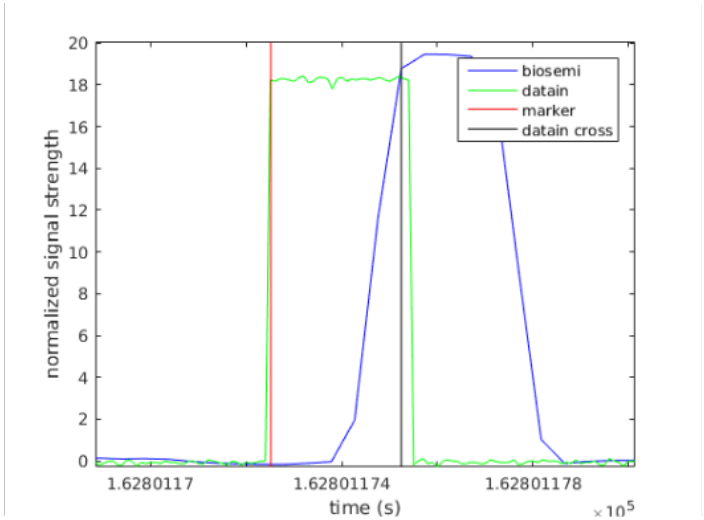
This is bad:

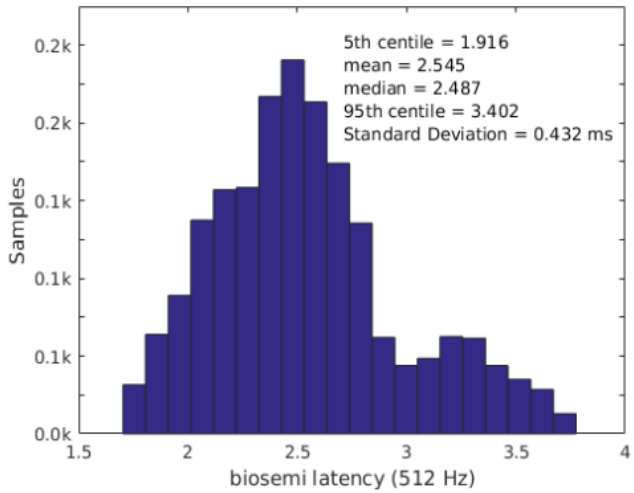


Red line is linearized timestamps, blue is raw. On the right is the difference: between +/-150s !!! .

Determine device Lag:

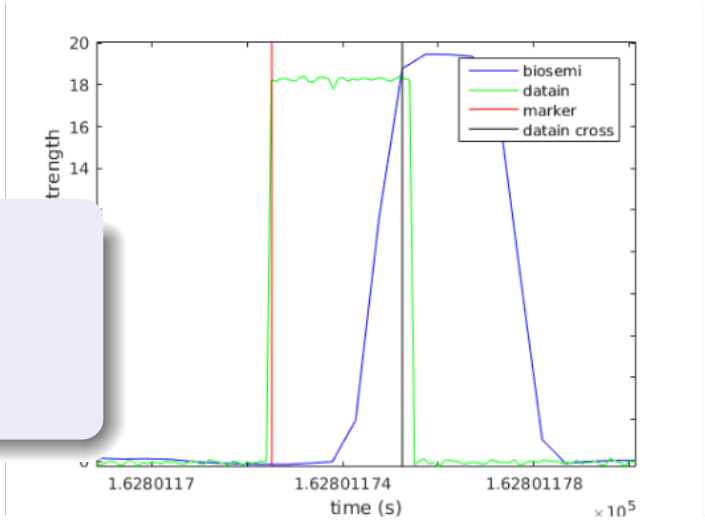
Determine device Lag:

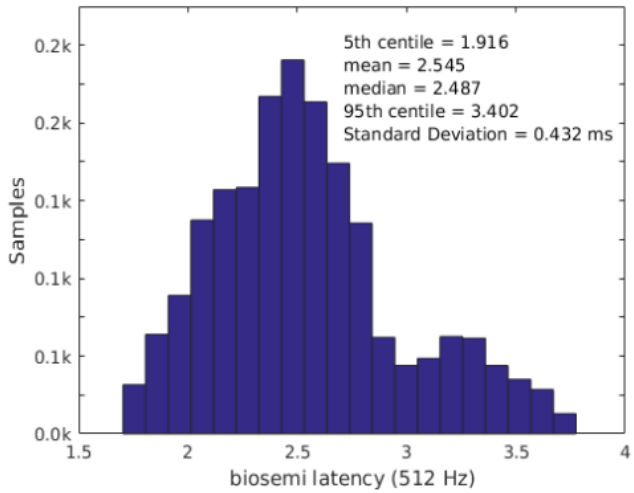


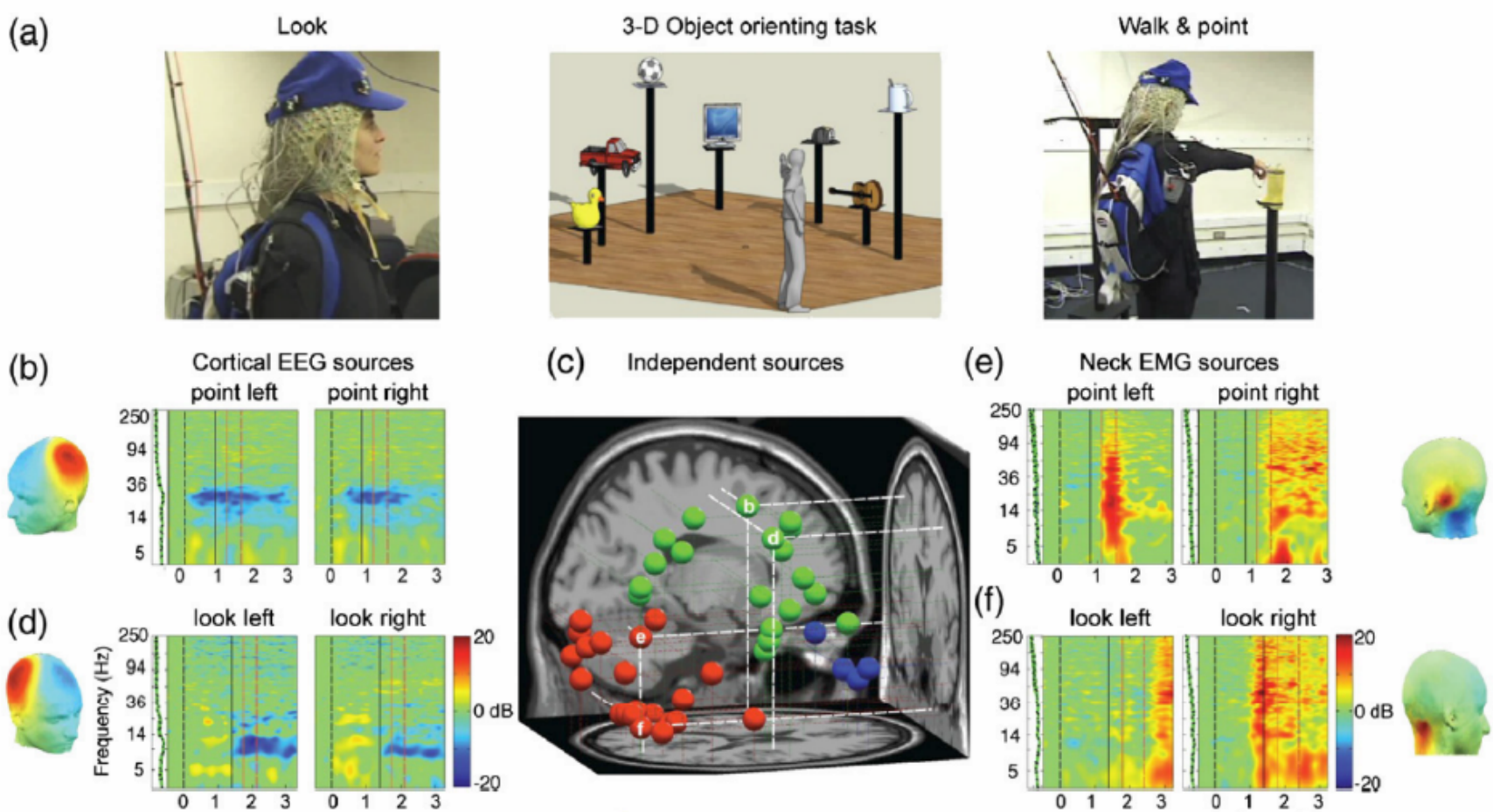


Determine device Lag:

To complete the synchronization, simply subtract the mean.






(a)
Look
3-D Object orienting task
Walk & point
(b)
Cortical EEG sources
point left
point right
(c)
Independent sources
(e)
Neck EMG sources
point left
point right
(d)
look left
look right
Frequency (Hz)
250
94
36
14
5
0 1 2 3
20
0 dB
-20
(f)
look left
look right

Thank You!