MatRiver software for real-time multimodal analysis and visualization

Nima Bigdely Shamlo, nima@sccn.ucsd.edu
James Li
Swartz Center for Computational Neuroscience – Institute for Neural Computation, UCSD
July 10, 2009

Recent performance gains in computer hardware and the introduction of multi-core CPUs have made it possible to process data in real-time using the Matlab scripting language and processing environment. The MatRiver software package is a set of Matlab functions that communicate with the DataRiver system (A. Vankov) by reading and writing DataRiver data streams to perform data analysis and visualization and/or produce or adjust a subject stimulus stream.

Ease of use. With MatRiver, users can leverage their existing knowledge of Matlab and its extensive mathematical and visualization capabilities. Existing Matlab scripts for EEG analysis and classification can be easily used in conjunction with MatRiver functions with minimal modifications. This greatly reduces the need for re-implementing the same code in different languages (for example, replicating operations originally written as Matlab scripts for EEG classification as a C program) for real-time implementation. This ease of use may allow easier exploratory development of more complex and powerful real-time analysis methods.

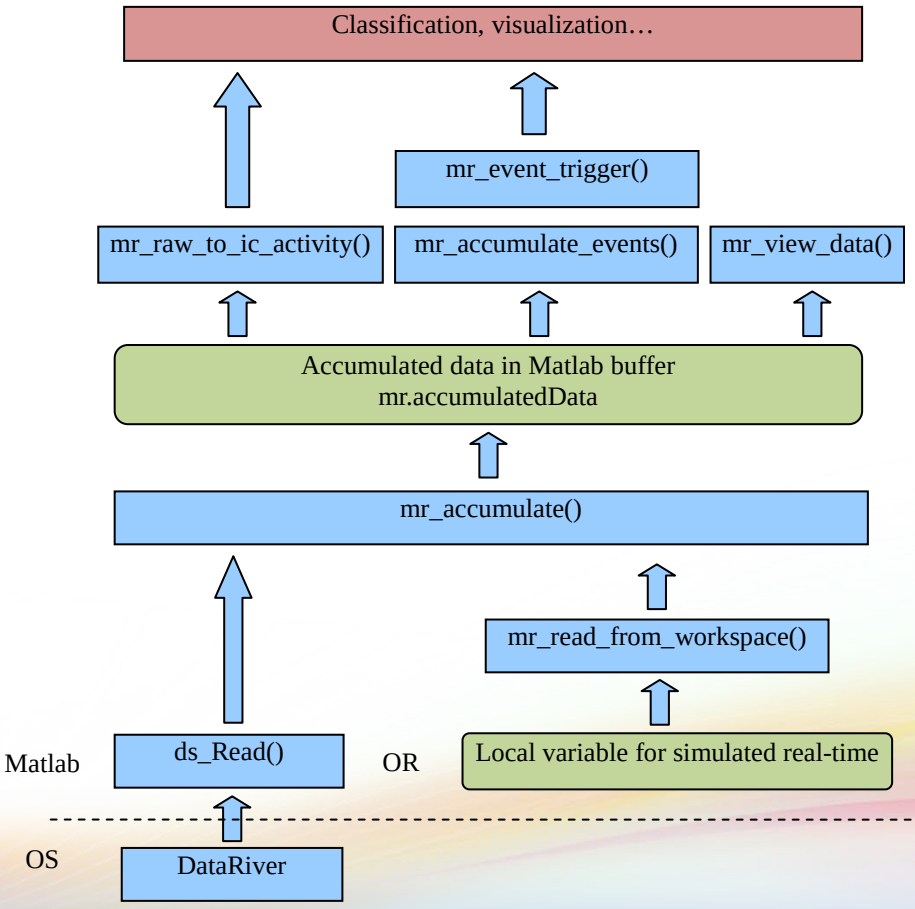


Figure 1. Data flow in MatRiver

Architecture. Fig. 1 above shows the relationship between current components of MatRiver system. Data can be read either from DataRiver system using *ds_Read()* function or a real-time session can be simulated by reading data from a local Matlab variable. In the simulated case, data samples are provided to *mr_accumulate()* function with the same pace as if they were produced with a specific sampling rate.

Since data is already buffered on the local machine outside Matlab by the DataRiver system, MatRiver system does not have to continuously check for newly arrived data. MatRiver can check the buffer through *ds_Read()* function on regular (~0.1 s) intervals by exploiting timer objects and read all the newly arrived samples in one quick step. This feature is essential for real-time operations since it eliminates the chance of sample loss when Matlab process is busy.

Matlab timer objects execute a callback function or script in regular intervals and free up the CPU time between these runs. MatRiver functions are usually invoked by timer objects in specified intervals. At least one timer has to be executed during a real-time session to read the incoming data by running *mr_accumulate()* function. Other MatRiver functions can be executed by the same timer object or in other timer objects to achieve different execution rates. For example, one might decide to read data every 50 ms but visualize it in 100 ms intervals by invoking *mr_view_data()* in another timer object.

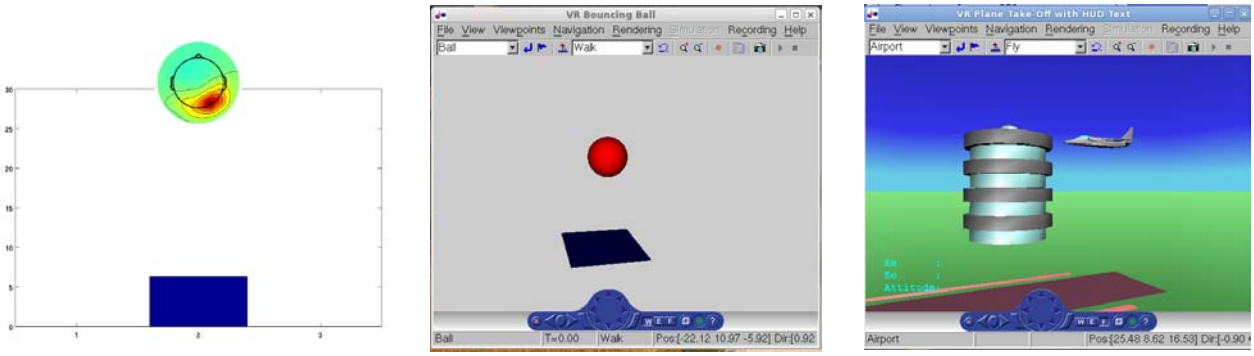
Data read from DataRiver system or simulated through *mr_read_from_workspace()* function is accumulated in a Matlab global variable *mr.accumulatedData* and made available to all MatRiver functions. Each of these functions acts on a subset of accumulated data related a specific modality (EEG data, Event data, Mocap data, …). For example, *mr_raw_to_ic_activity()* function calculates IC activity from EEG channel data or *mr_accumulate_events()* function read events from the event channel and accumulates them in *mr.eeg.event* field.

In addition to continuous execution by timer objects, MatRiver function can be invoked or *triggered* by events read from the event channel. This is achieved in *mr_event_trigger()* by defining triggering events and the duration of time after which a trigger callback function, for example a classifier, should be executed.

Visualization performance. Matlab does not offer particularly fast image rendering or the most temporally accurate method of visual presentation. This said, our experiments show that a Matlab process running on the local machine equipped with a mid-level OpenGL capable graphics card can perform most visualization task relevant to EEG analysis with acceptable performance.

In the example in Fig. 2 (below), near-real time changes in alpha power in a posterior component were visualized in three different ways. Processing delay was measured by the difference between the number of data samples in the accumulated data and the number of samples that should have been received at that latency, based on the time the session had started and the data sampling rate. MatRiver was executed in simulated real-time mode using a timer object to perform data acquisition (*mr_accumulate()*), calculating IC activity (*mr_raw_to_ic_activity()*) by vector multiplication, estimating alpha power, and visualizing the alpha power series. Delay was calculated after the visualization function (*mr_record_lag()*). The timer object was set to run its maximum (100 times per second). The actual rate at which timer was executed was dependent on the time it took for its MatRiver callback function to run.

Example: Alpha power feedback



(a) (b) (c)

Figure 2. Three ways to perform real-time visualization of alpha band power *computed from an occipital independent EEG component using MatRiver functions to change the height of: (a) a bar graph (b) a 3-D ball rendered in the Matlab VRML environment (c) an airplane rendered in Matlab VRML environment.*

Table 1 shows the delay experienced using several visualization parameters:

Condition	Median Delay (ms)
Calculating IC activity with no visualization	5.5
Visualizing alpha power by bar graph height (Fig. 2a); zbuffer renderer	45
Visualizing alpha power by bar graph height (Fig. 2a); OpenGL renderer	39
Visualizing alpha power by ball height in Matlab VRML (Fig. 2b)	25
Visualizing alpha power by airplane height in Matlab VRML (Fig. 2c)	25

Table 1. *Processing delay during simulated real time operation of the systems shown in Fig. 2.*

For all these conditions, an acceptable frame rate (above 20 fps) is possible on an unexceptional PC.

MatRiver core functions

1. **mr_init()** initializes connection to local buffers of the DataRiver system.
2. **mr_accumulate()** reads real-time (or simulated real-time) data and accumulates it in the *mr.accumulatedData* field. To save memory and prevent slowdown due to disk-swapping, the length of data accumulated in *mr.accumulatedData* can be limited automatically in this function by setting the *mr.maxNumberOfFramesInAccumulatedData* field to a finite value.
3. **mr_read_simulated()** reads data from mr.simulated.data field and provides it to *mr_accumulate()* to simulate a real-time session.
4. **mr_raw_to_ic_activity()** calculate the activity of select ICs for a portion of data and places it in the *mr.eeg.icActivity* field**.** The advantage of this method is that the activity is calculated only when necessary, minimizing the CPU burden in case this information is requested infrequently (as in an infrequent trigger scenario). Also since filtering is performed on a portion of data, non-

casual filtering can be used to minimize phase distortions (using the Matlab *filtfilt()* function). The execution time for this function increases linearly with the number of channels/components processed and the duration for which the activity is requested.

5. **mr_raw_to_ic_activity_accumulate()** continuously calculates independent component (IC) activities and accumulates them in the *mr.eeg.accumulatedIcActivity* field. Since only newly received data frames are processed, this requires less CPU time in continuous tasks. On the other hand, here ICA is calculated even when it is not used by any MatRiver function, so only casual filters (IR) can be used.
6. **mr_accumulate_events()** reads events from the DataRiver event channel and accumulates them in the *mr.eeg.event* field. Each event in *mr.eeg.event* structure has three fields: *Type*, *Latency* (in frames) and *Triggered*. The latter indicates whether the event has already triggered a callback function.
7. **mr_event_trigger()** goes through events in *mr.eeg.event* and triggers callback functions for the event when necessary. Each trigger is defined by the types of events and the duration of time after these for which the callback has to be executed. For example, a trigger can launch a classifier after 800 ms has passed after events of type 1 or 2.
8. **mr_delete_timers()** stops and clears all timer objects.

MatRiver auxilary functions

1. **mr_view_data()** visualizes the real-time data flowing into the MatRiver system.
2. **mr_record_lag()** records the execution lag. This is the difference between the number of data frames currently in the accumulated data and the number of frames that should have been received based on the time the session started and the data sampling rate. These lags are recorded in the *mr.executionLag* field which can be used to monitor the performance of the system.

MatRiver miscellaneous functions and scripts

1. **mr_test_with_simulated_data()** runs MatRiver in simulated real-time mode.
2. **mr_power_on_cortex()** shows power in a given frequency range on the MNI cortical image.
3. **mr_eeg_ocean()** performs an eegOcean visualization (N. Bigdely–Shamlo) in 2-D.
4. **mr_eeg_ocean_cortex()** performs an eegOcean visualization on the MNI cortical surface.
5. **mr_spectrum_ball()** visualizes alpha power by changes in the height of a 3-D ball (or airplane) in VRML under Matlab.
6. **mr_spectrum_bar()** visualizes alpha power by modulating the height of bar in a Matlab figure.