The Databaser© Data Managment System

Jason J. Kutch, Ph.D.

July 22, 2012

1 Introduction

When I started my post-doctoral research at USC in the fall of 2008, I realized that it was all too easy for data in human neurophysiology experiments to get scattered in a large number of text, MATLAB, and Excel files. I wanted a general tool to manage my data, and it needed to meet three requirements. First, it should be easy to develop and not turn into a huge time-sink. The Matlab gui environment was very useful for this. Second, it needed to allow me to coalesce data collected on multiple systems (EMG, force recording, motion capture) into a standard MATLAB format. Third, it needed to allow me to write scripts that could be quickly applied to subsets of my data to produce tabular or publication-ready figures with minimum effort. Later, a module was added to control experiments.

This document, which was generated in LaTeX and is fully searchable, describes the all functions of the Databaser to the level of detail needed to operate the system.

2 Experiment and Session

The Experiment and Session are nothing more than folders where the Trial folders are stored. They are necessary for the "Build Database" and "Analyze" functions, but not the "Collect Data" or "Publish!" functions. The "Experiment" pull-down menu can be used to create a new experiment folder, or open an existing one. Likewise, the "Session" pull-down can be used to create a new session folder within the experiment folder, or open an existing session folder.

Trials will be stored in the folder "Active Experiment"/ "Active Session".

3 Collect Data

The functions in the "Collect Data" box are intended to facilitate the generation of text files needed for experiment control.

3.1 Make Path for Input Files

This push button sets a folder where Input files will be stored. An Input file, for example, may be a text file that contains a list of commands to be sent to a controller. If the radio button "Set to FTP?" is checked, clicking "Make Path for Input Files" will allow the user to navigate an FTP server (useful if the command files need to be written to a remote controller.

3.2 Notebook

During the course of running an experiment, it is often very useful to carefully document events that occurred at specific times. This can be accomplished, semi-automatically, using the Notebook feature of the Databaser. Clicking "New" or "Open" will generate or open, respectively, a .mat file that stores the notebook. The name of the current notebook file will then be displayed under "Current Notebook". When you want to add a notebook entry, click "Add Entry", and you will be prompted to enter a line of text. This line is stored in the notebook, along with a time-stamp.

3.3 Trial Inputs

The listbox under "Trial Inputs" contains a list of Trials.

4 Build Database

The "Build Database" panel was designed to aid in combining separate data streams into one common MATLAB .mat file containing the data for the trial.

4.1 Setting up

As an example, imaging you have raw data in a folder called /Users/Jason/RawData/2010Jul01Data. In this folder, you have motion capture data files and LabView data files:

Vicon Files	**LabView Files**
Trial01Vicon.csv	Trial01LabView.txt
Trial02Vicon.csv	Trial02LabView.txt
Trial03Vicon.csv	Trial03LabView.txt
⋮	⋮

The first step in incorporting these files into a database is to click "Set Path for Add Trial(s)", and choose "/Users/Jason/RawData/2010Jul01Data". We're going to call the Vicon files "Input 1" and the LabView files "Input 2", for convenience (this labeling doesn't matter as long as you are consistent, as we will see below). In the list box for "File(s) to Incorporate", we want a list of the Vicon files when "Input" at the bottom of the list box is set to 1, and a corresponding list of LabView files when "Input" is set to 2. There are 2 ways to populate this list: either use the "+" button, or use the "Search and Select" textbox. The search and select will search the folder "/Users/Jason/RawData/2010Jul01Data" for files specified by wildcards. For instance, if you want to put all the Vicon files in "File(s) to Incorporate" list box, simply type "*Vicon*" in the "Search and Select" field. Then, switch to Input 2 and type "*Lab*" to get all of the LabView files into their proper place. Notice that the files will go in the right order because they are numbered sequentially.

If trials already exist in the session, you can add these to the corresponding "... into Trial ..." list (in the order corresponding to the "File(s) to Incorporate List"). If you want additional processing to append current trial data, click the "Append?" check box.

4.2 Readers

Readers are m-files designed to take the files to incorporate, process them, and write them into a single .mat file that contains all data for the trial. To make a new reader, click the "+" button.

Readers can be divided into classes to keep them better organized. All readers are in the folder "DatabaserProject/Readers", and classes are subfolders within this directory. If you make a new reader, it will show up in the list, and you can right click it to open it in the editor.

When you press "Build", the reader will sequentially work through each trial.

When you open a newly-generated reader, you will find some help comments already written. ReaderInput is a struct with three fields. ReaderInput.Files contains the names (including path) of the files in the "File(s) to Incorporate" list box. For instance, when the reader runs on the first trial, ReaderInput.Files1 will be "/Users/Jason/RawData/2010Jul01Data/Trial01Vicon.csv" and ReaderInput.Files2 will be "/Users/Jason/RawData/2010Jul01Data/Trial01LabView.csv". When the reader finishes the first trial and moves to the next, these will be updated to 02 and so on. ReaderInput.HelperFunctions is not currently used. ReaderInput.ProgramStatusIndicator is a handle to the "Program Status" bar on the front panel. You can use it to update the status of the readers for the user. For instance

```
set(ReaderInput.ProgramStatusIndicator,'string','Hi. Still working. 5% complete');
```

will write "Hi. Still working. 5% complete" to the status indicator. This string can be created dynamically during a for loop to keep the user informed of progress.