Colorado State University
Brain-Computer Interfaces Lab

CEBL3: A New Software Platform for EEG Analysis and Rapid Prototyping of BCI Technologies

Brainwaves Research Lab

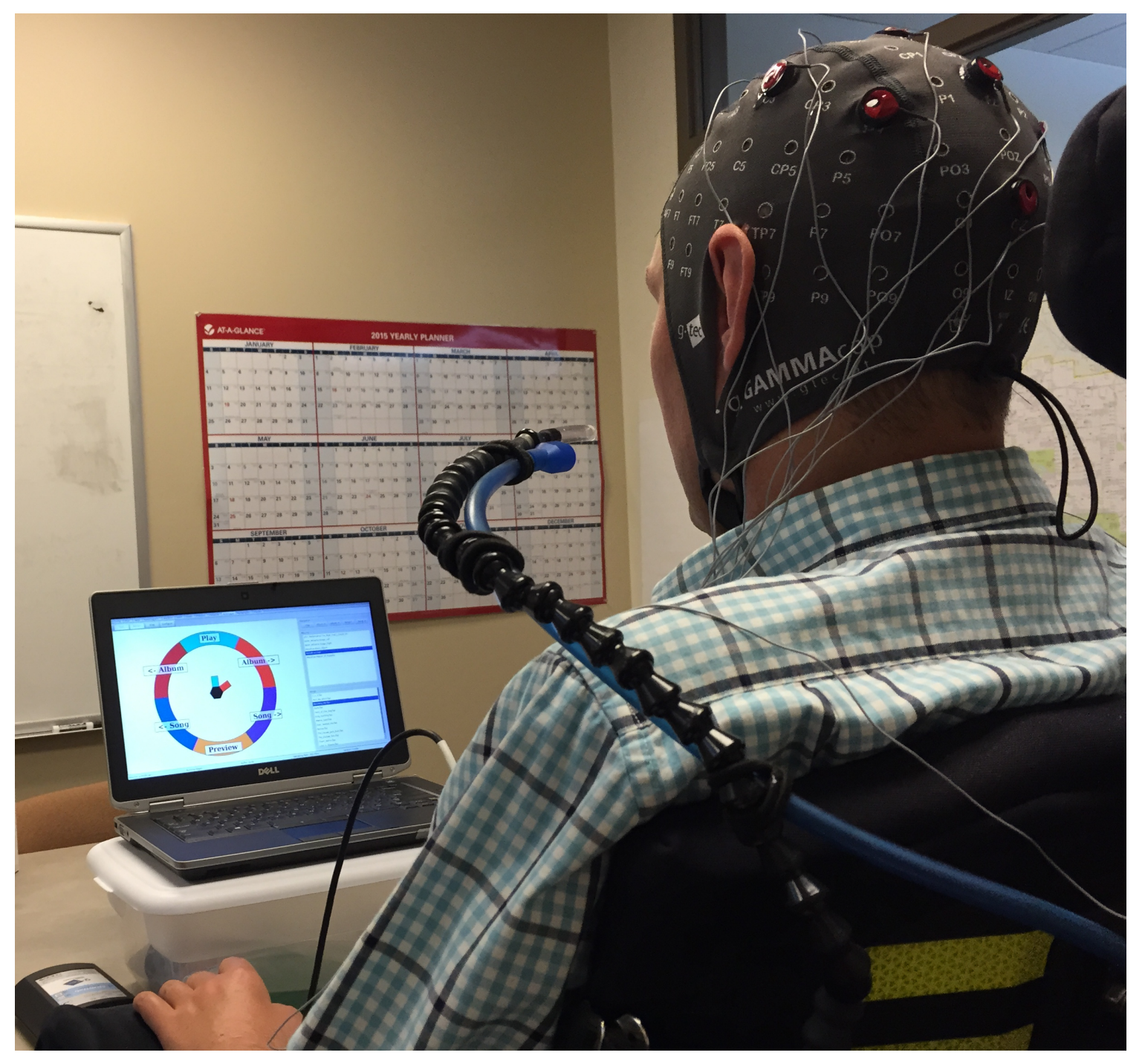
Elliott Forney, Charles Anderson, Igor Ryzhkov, William Gavin, Patricia Davies, Marla Roll, Jewel Crasta, Brittany Taylor, Fereydoon Vafaei

NSF Grant Number 1065513

A New Software Platform for BCI 1

CEBL3

The Colorado Electroencephalography and Brain-Computer Interfaces Laboratory version 3 (CEBL3) is a complete re-write of The CSU BCI Group's flagship software platform.



Most currently available BCI software packages fall short in one of two areas:

1) they are not well-suited for real-time and interactive experiments.

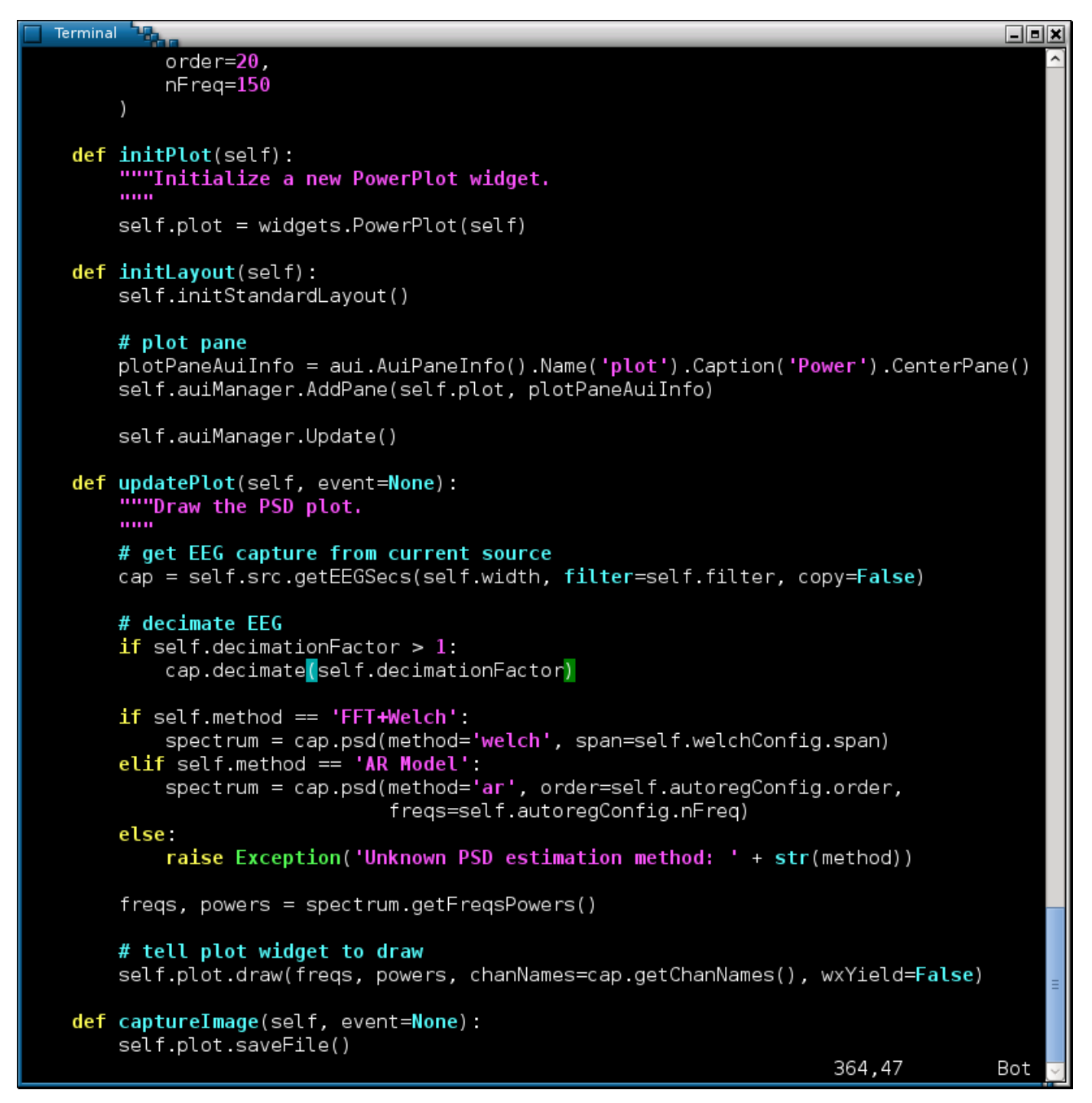
2) *or* they can not be easily and rapidly extended to use new paradigms, methods, algorithms or interfaces.

CEBL3 addresses these issues through the use of high-level programming languages and constructs and through a modular design that promotes code-reuse throughout the entire process of developing BCI prototypes.

CEBL3 also includes a number of state-of-the-art signal processing and machine learning algorithms as well as various modules for graphics, visualization and rapid development of user interfaces and communication protocols.

Python, NumPy, SciPy, Matplotlib, wxPython, et al. 2

CEBL3 is written almost entirely in Python.



```
            order=20,
            nFreq=150
        )

    def initPlot(self):
        """Initialize a new PowerPlot widget.
        """
        self.plot = widgets.PowerPlot(self)

    def initLayout(self):
        self.initStandardLayout()

        # plot pane
        plotPaneAuiInfo = aui.AuiPaneInfo().Name('plot').Caption('Power').CenterPane()
        self.auiManager.AddPane(self.plot, plotPaneAuiInfo)

        self.auiManager.Update()

    def updatePlot(self, event=None):
        """Draw the PSD plot.
        """
        # get EEG capture from current source
        cap = self.src.getEEGSecs(self.width, filter=self.filter, copy=False)

        # decimate EEG
        if self.decimationFactor > 1:
            cap.decimate(self.decimationFactor)

        if self.method == 'FFT+Welch':
            spectrum = cap.psd(method='welch', span=self.welchConfig.span)
        elif self.method == 'AR Model':
            spectrum = cap.psd(method='ar', order=self.autoregConfig.order,
                                   freqs=self.autoregConfig.nFreq)
        else:
            raise Exception('Unknown PSD estimation method: ' + str(method))

        freqs, powers = spectrum.getFreqsPowers()

        # tell plot widget to draw
        self.plot.draw(freqs, powers, chanNames=cap.getChanNames(), wxYield=False)

    def captureImage(self, event=None):
        self.plot.saveFile()
                                                                    364,47         Bot
```

Python is a concise programming language with a number of features that aid in the rapid-development of clearly-written code.

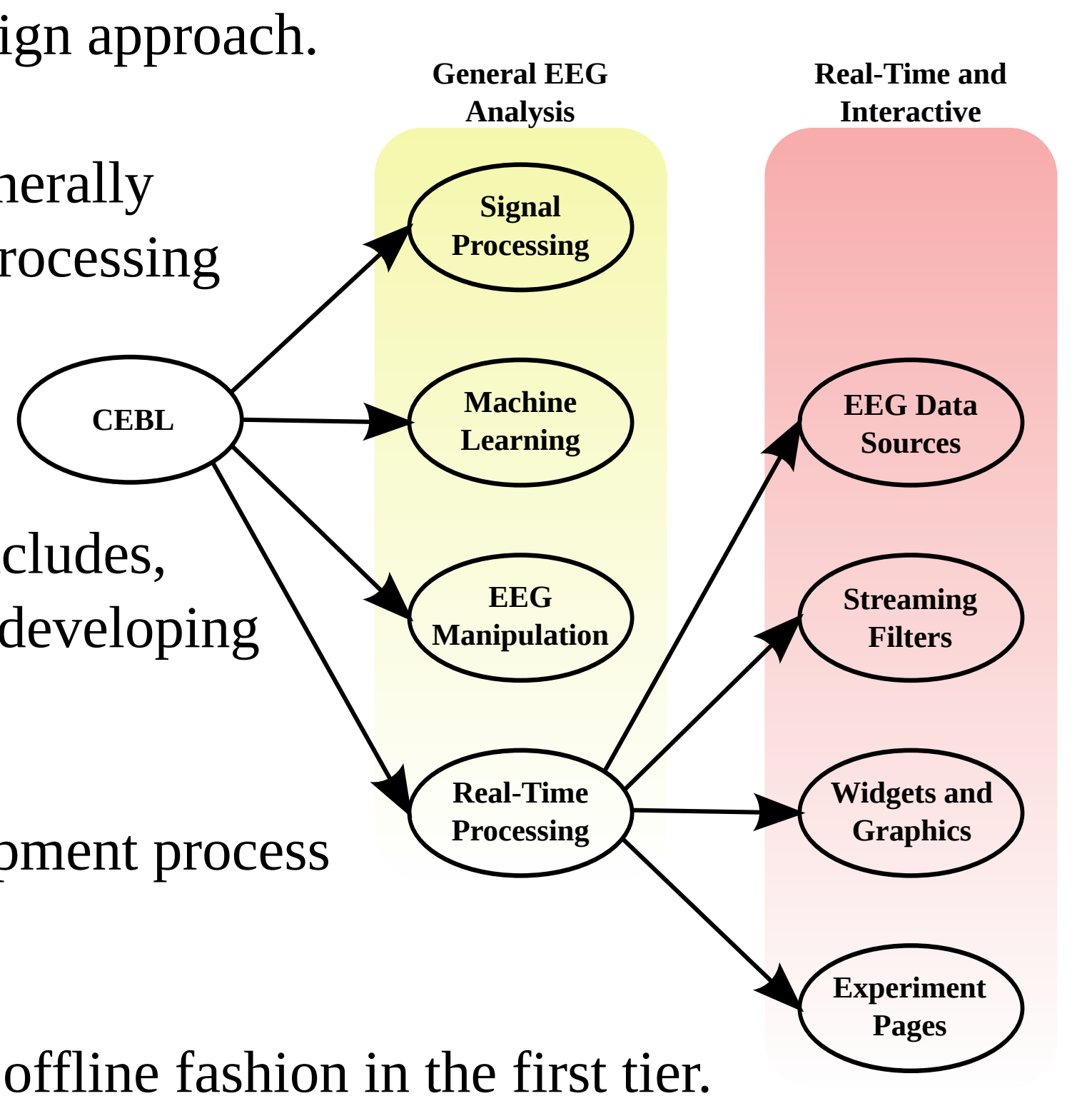
Python has many available packages that allow developers to leverage existing functionality.

CEBL3 relies heavily on NumPy and SciPy, which are highly optimized for numerical computations. These libraries, along with a few custom C and Cython modules, allow CEBL3 to achieve the levels of performance required for developing real-time BCIs.

wxPython is used for the CEBL3 graphical interface. wxPython is portable, extensible, straight-forward to use and includes many features and widgets.

Modular Design 3

CEBL3 follows a modular, two-tiered design approach.



The first tier contains modules that are generally useful for EEG analysis. Such as signal processing and machine learning algorithms.

The second tier contains modules that are specific to real-time functionality. This includes, hardware drivers, widgets and classes for developing BCI modules, called "pages."

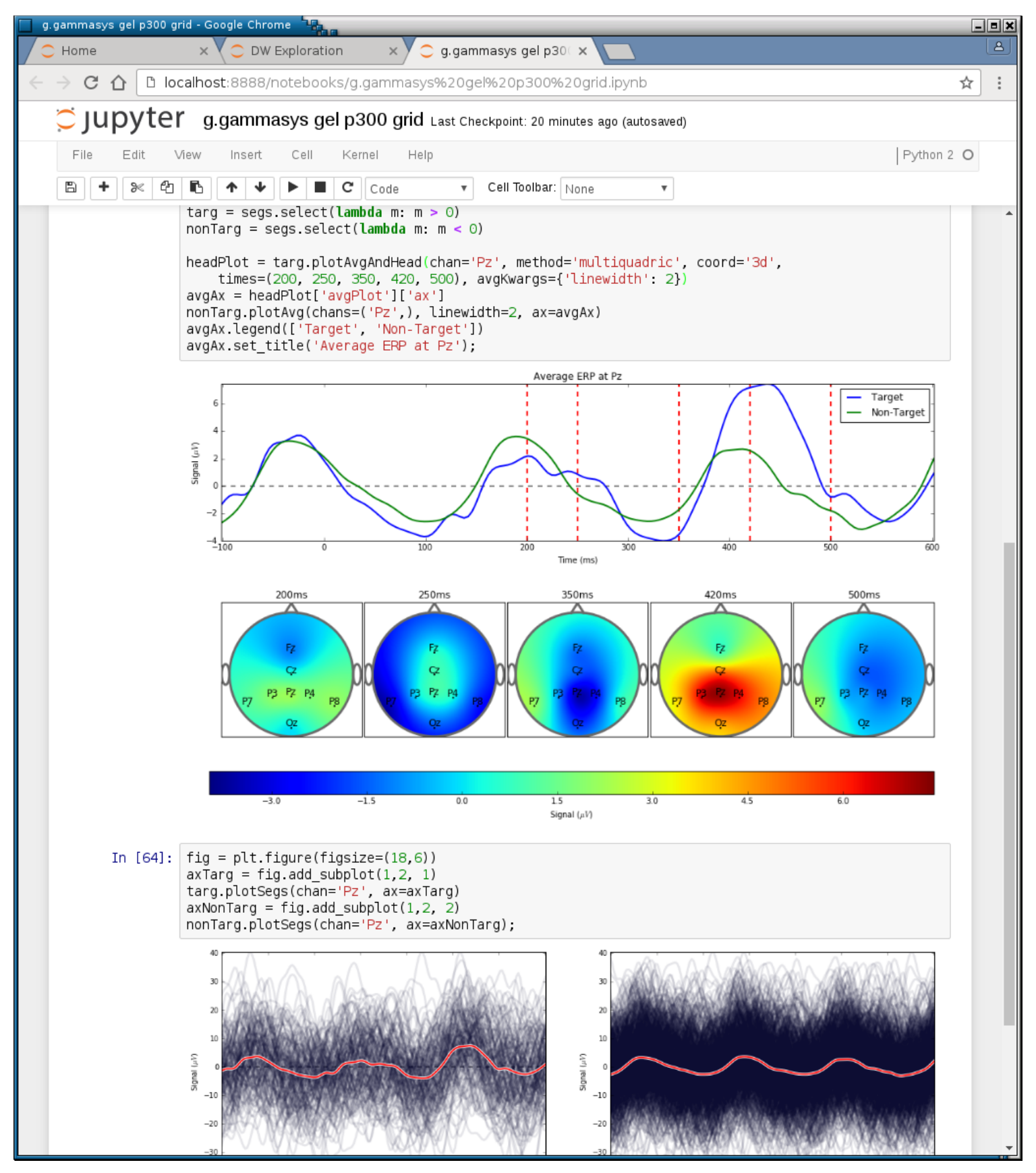
This two-tiered design follows the development process from beginning to end:

A researcher can first test new ideas in an offline fashion in the first tier.

Successful experiments can then be promoted to real-time pages in the second tier.

Offline Analysis in Jupyter Notebooks 4

Offline analysis can be performed using Jupyter Notebooks.


g.gammasys gel p300 grid - Google Chrome
Home
DW Exploration
localhost:8888/notebooks/g.gammasys%20gel%20p300%20grid.ipynb
jupyter g.gammasys gel p300 grid Last Checkpoint: 20 minutes ago (autosaved)
File Edit View Insert Cell Kernel Help
Python 2
Code
Cell Toolbar: None
Average ERP at Pz
Target
Non-Target
Signal (μV)
Time (ms)
200ms
250ms
350ms
420ms
500ms
Signal (μV)

```
targ = segs.select(lambda m: m > 0)
nonTarg = segs.select(lambda m: m < 0)

headPlot = targ.plotAvgAndHead(chan='Pz', method='multiquadric', coord='3d',
    times=(200, 250, 350, 420, 500), avgKwargs={'linewidth': 2})
avgAx = headPlot['avgPlot']['ax']
nonTarg.plotAvg(chans=('Pz',), linewidth=2, ax=avgAx)
avgAx.legend(['Target', 'Non-Target'])
avgAx.set_title('Average ERP at Pz');
```

```
In [64]: fig = plt.figure(figsize=(18,6))
         axTarg = fig.add_subplot(1,2, 1)
         targ.plotSegs(chan='Pz', ax=axTarg)
         axNonTarg = fig.add_subplot(1,2, 2)
         nonTarg.plotSegs(chan='Pz', ax=axNonTarg);
```

Jupyter Notebooks provide a graphical, browser-based environment for performing data analysis in Python.

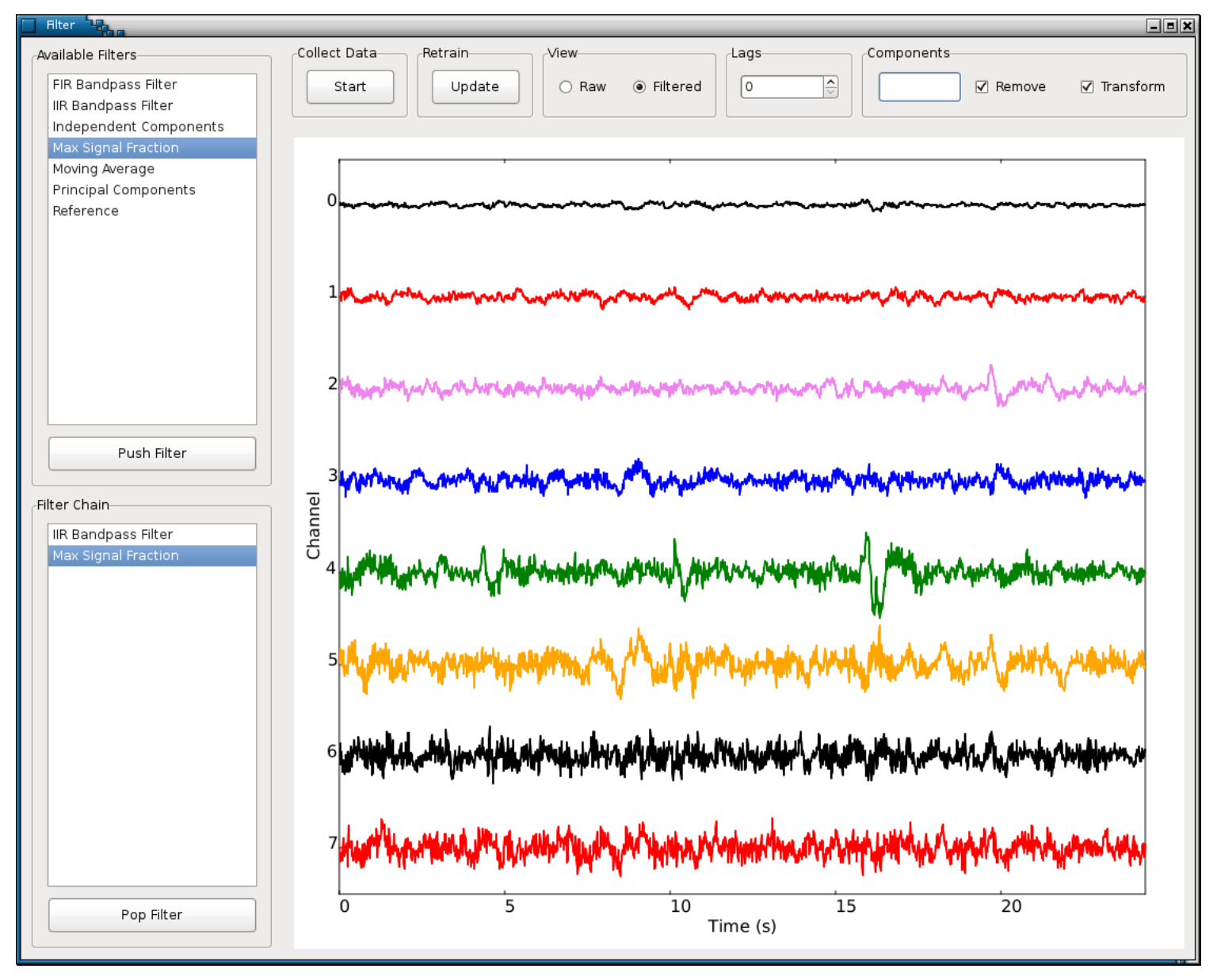
The cebl package can be imported in order to take advantage of all the first-tier features.

Many of these routines provide convenient plotting and manipulation routines.

Much of this code can then be re-used when creating real-time BCI pages.

Signal Processing 5

CEBL3 currently includes a number of standard and advanced signal processing algorithms, including:



IIR and FIR bandpass filters.

Moving averages and wiener filters.

Resampling and temporal interpolation.

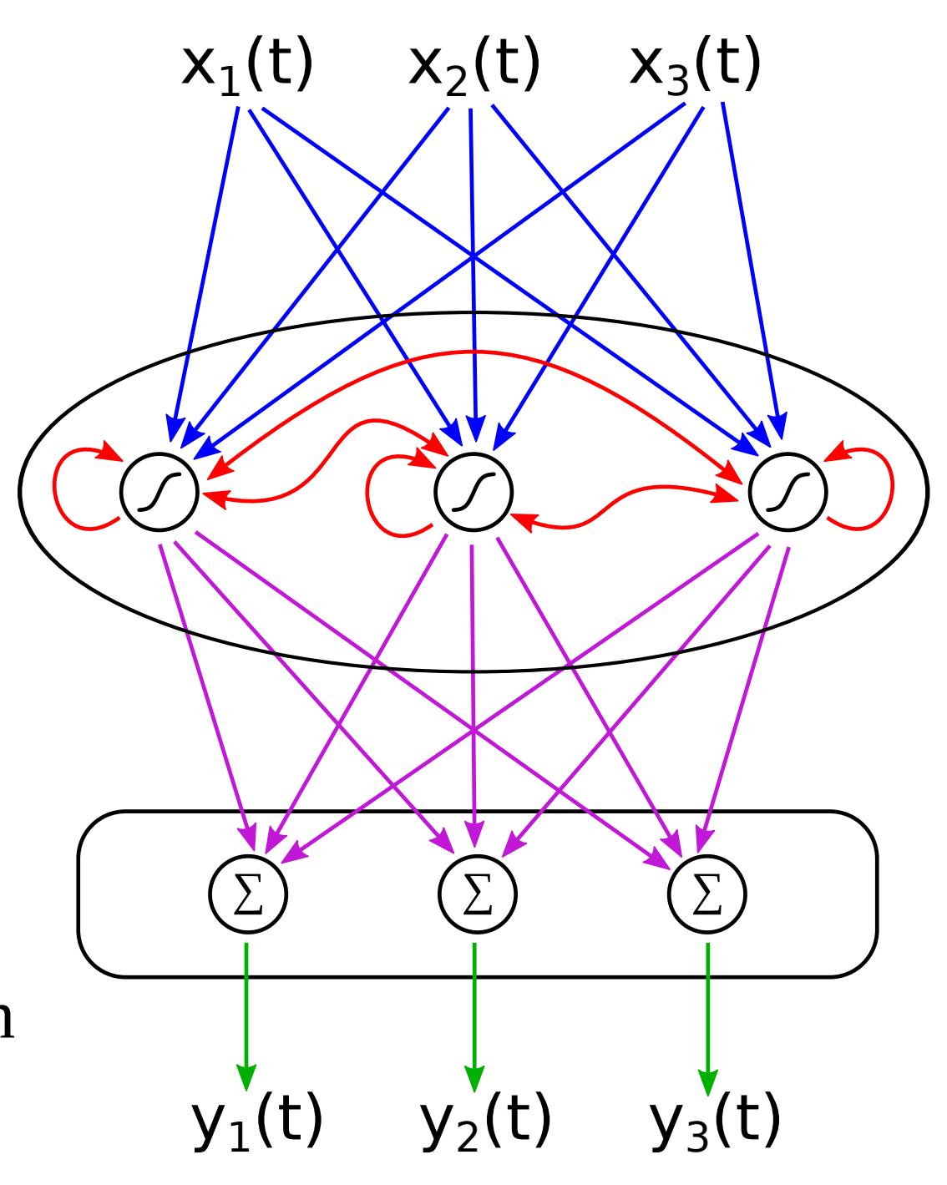
Spatial interpolation in 2d, 3d and spherical coordinates using linear, cubic, nearest-neighbor or radial basis function approaches, including splines.

Spectral estimation using Discrete Fourier Transforms, Welch's Method, Continuous Wavelet Transforms and Autoregressive Models.

Source separation using Maximum Signal Fraction, Independent Components Analysis, Common Spatial Patterns and Principal Components Analysis.

Machine Learning 6

CEBL3 also includes a number of machine learning algorithms that are useful for signal classification in Brain-Computer Interfaces, such as:



Linear and Quadratic Discriminant Analysis with covariance shrinkage regularization

Linear Logistic Regression with L2 and L1 norm regularization

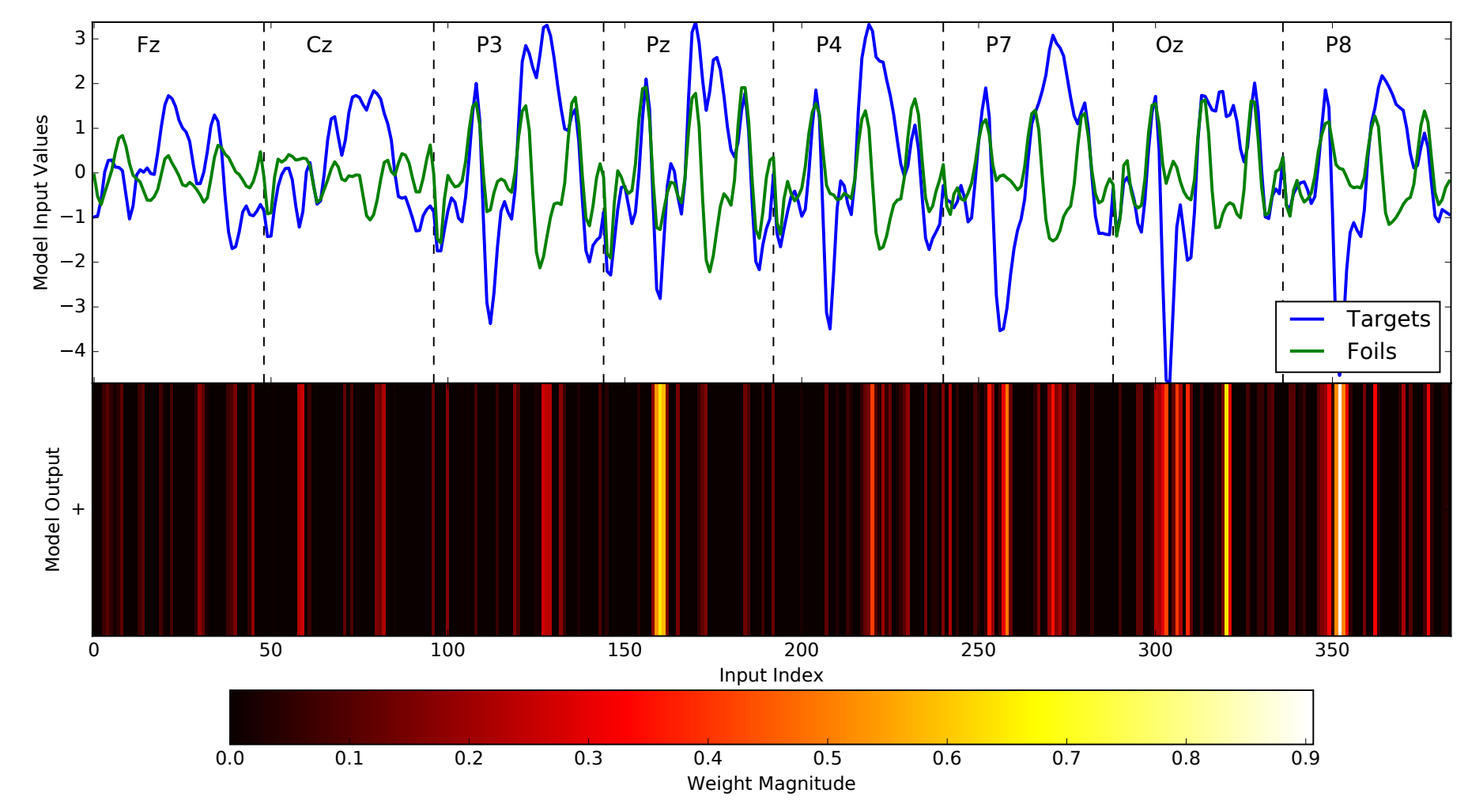
Feedforward, Recurrent and Convolutional Artificial Neural Networks with L2 and L1 norm regularization

K-Nearest Neighbors

Self-Organizing Maps

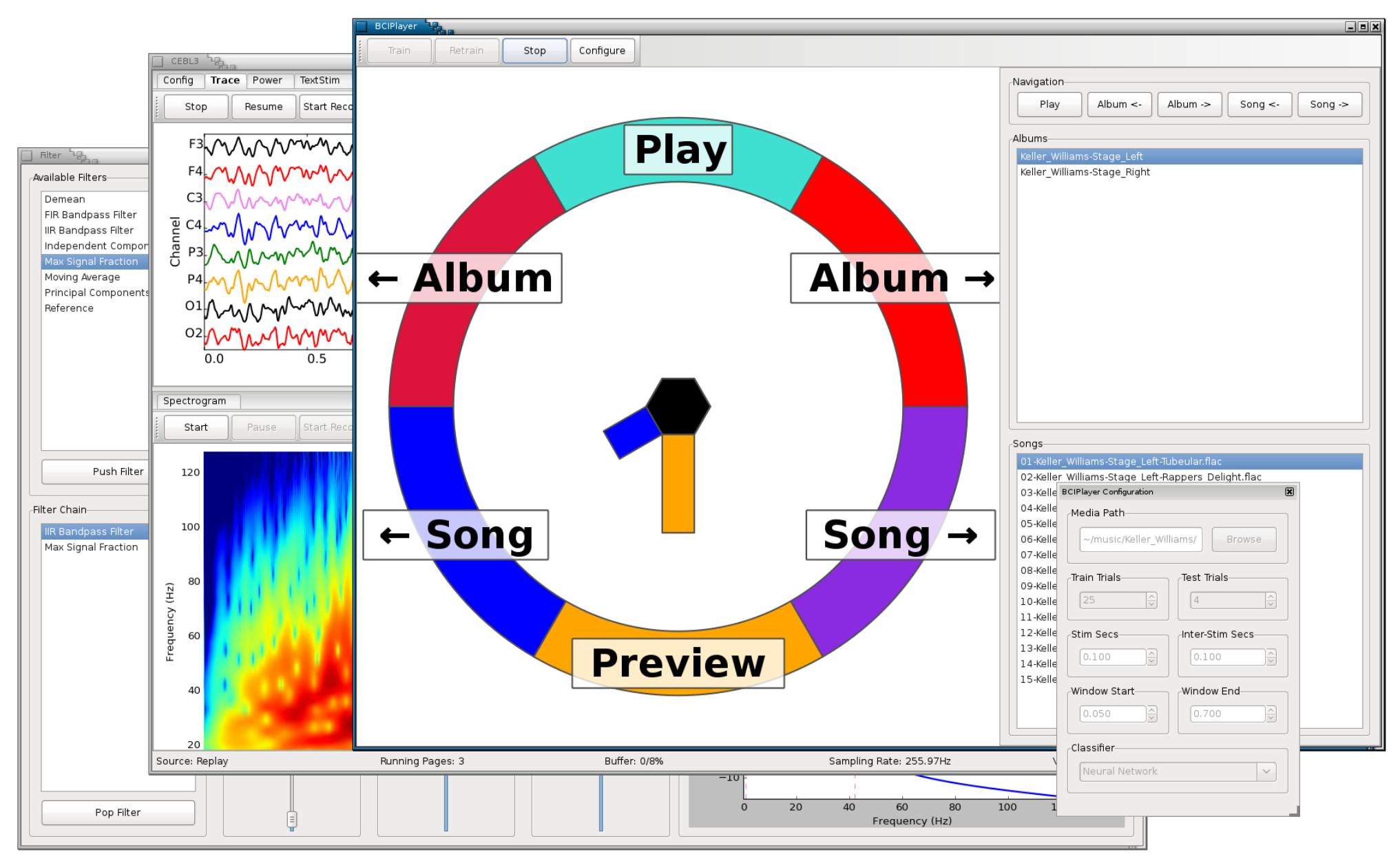
Various optimization routines

General framework for linear, non-linear and recurrent autoregressive models.



Real-Time Experiments with the CEBL3 7

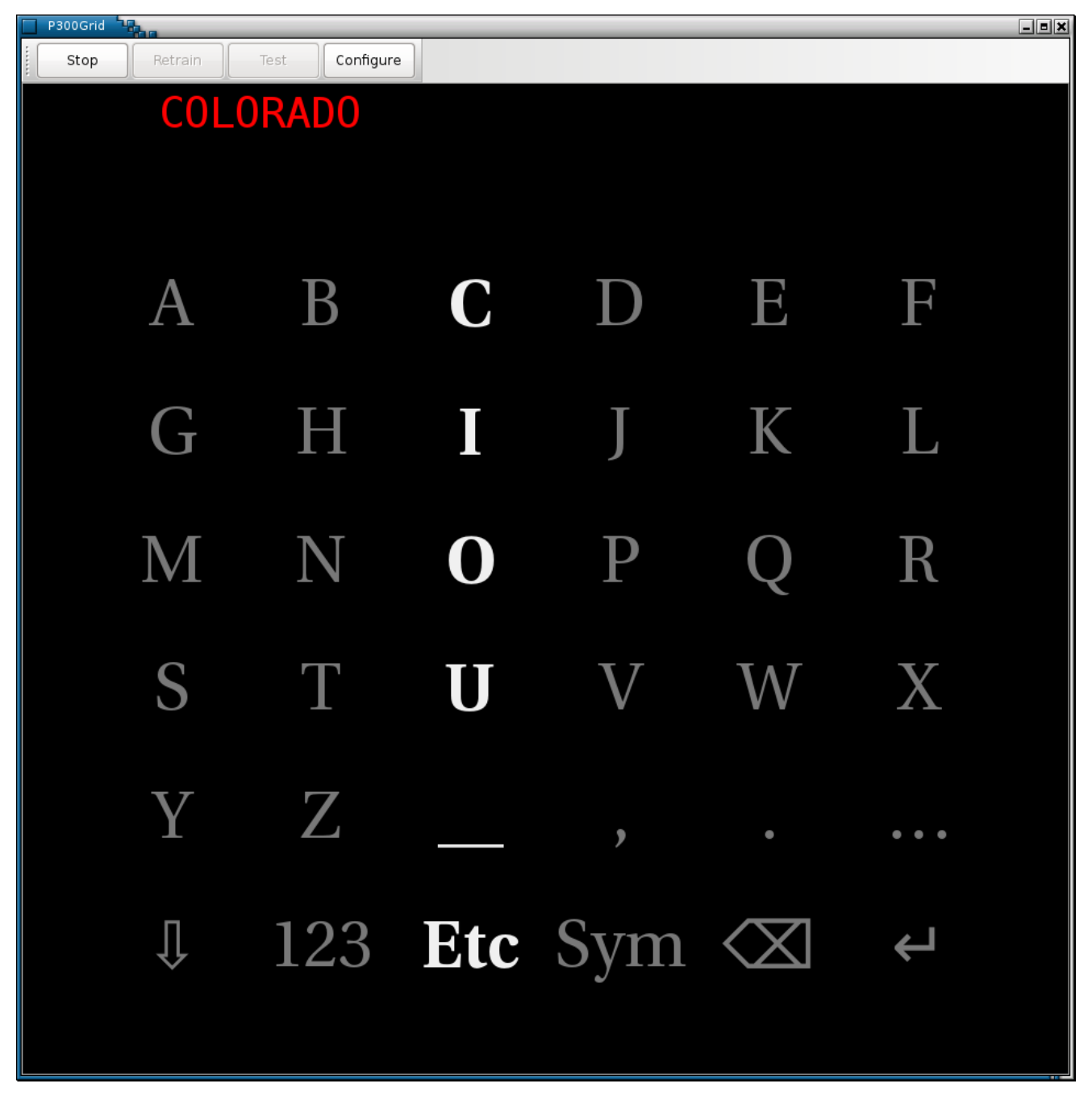
The CEBL3 Graphical User Interface takes the form of multiple tabbed pages.



The first few pages control the CEBL3 configuration, including hardware parameters, channel montages and real-time filters.

Additional pages constitute real-time BCI modules.

In Python, each page extends one of several base classes that provide common functionality.



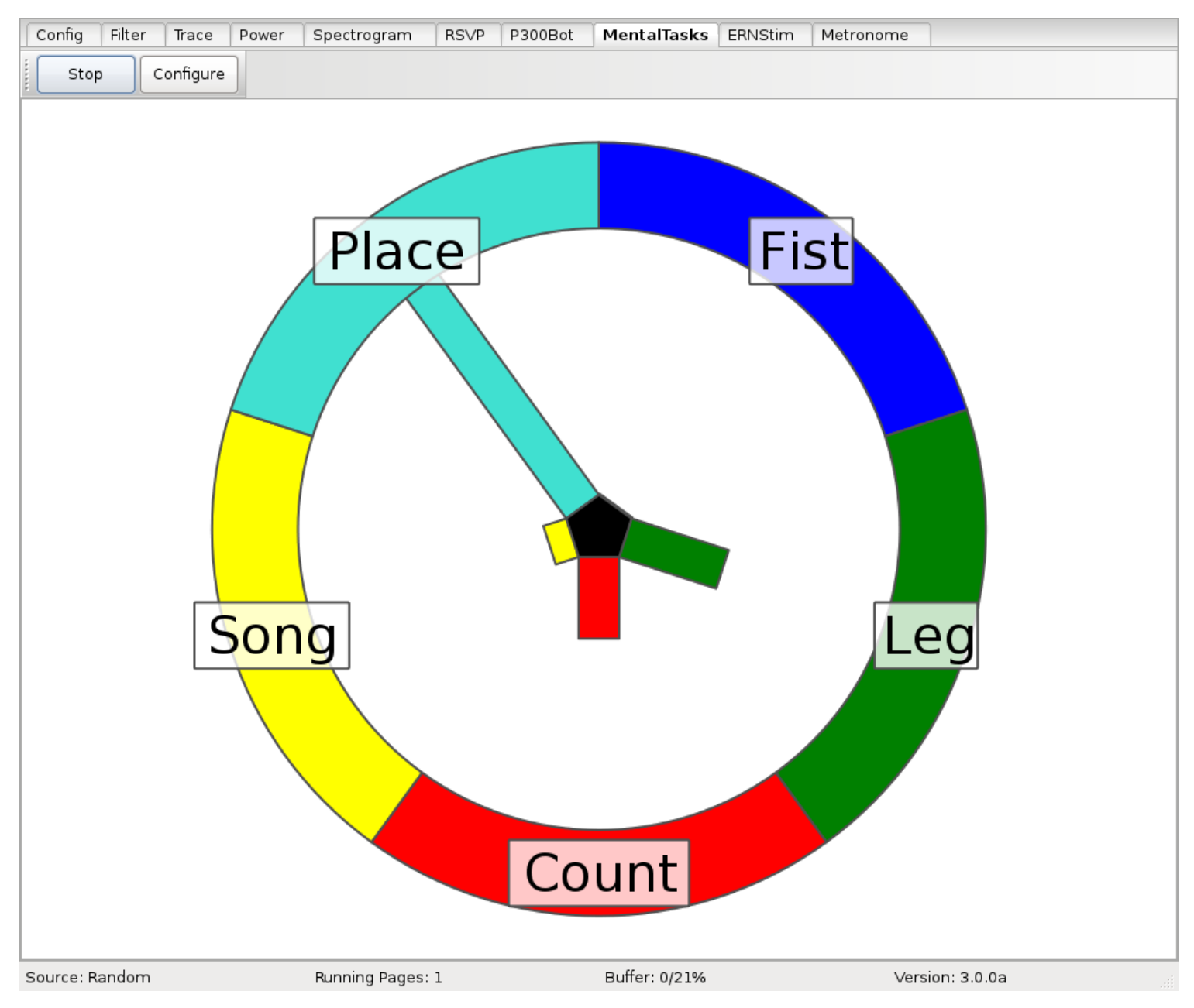
Code from offline experiments can be reused.

Various custom widgets are also included, such as a grid-speller, pie-menu and multimedia player.

Real-time data collection is currently supported from the g.tec g.MOBILab+ and g.Nautilus, the NeuroPulse Mindset24R and various replay and artificial sources. Support for theBiosemi ActiveTwo is currently in progress.

Current BCI Pages 8

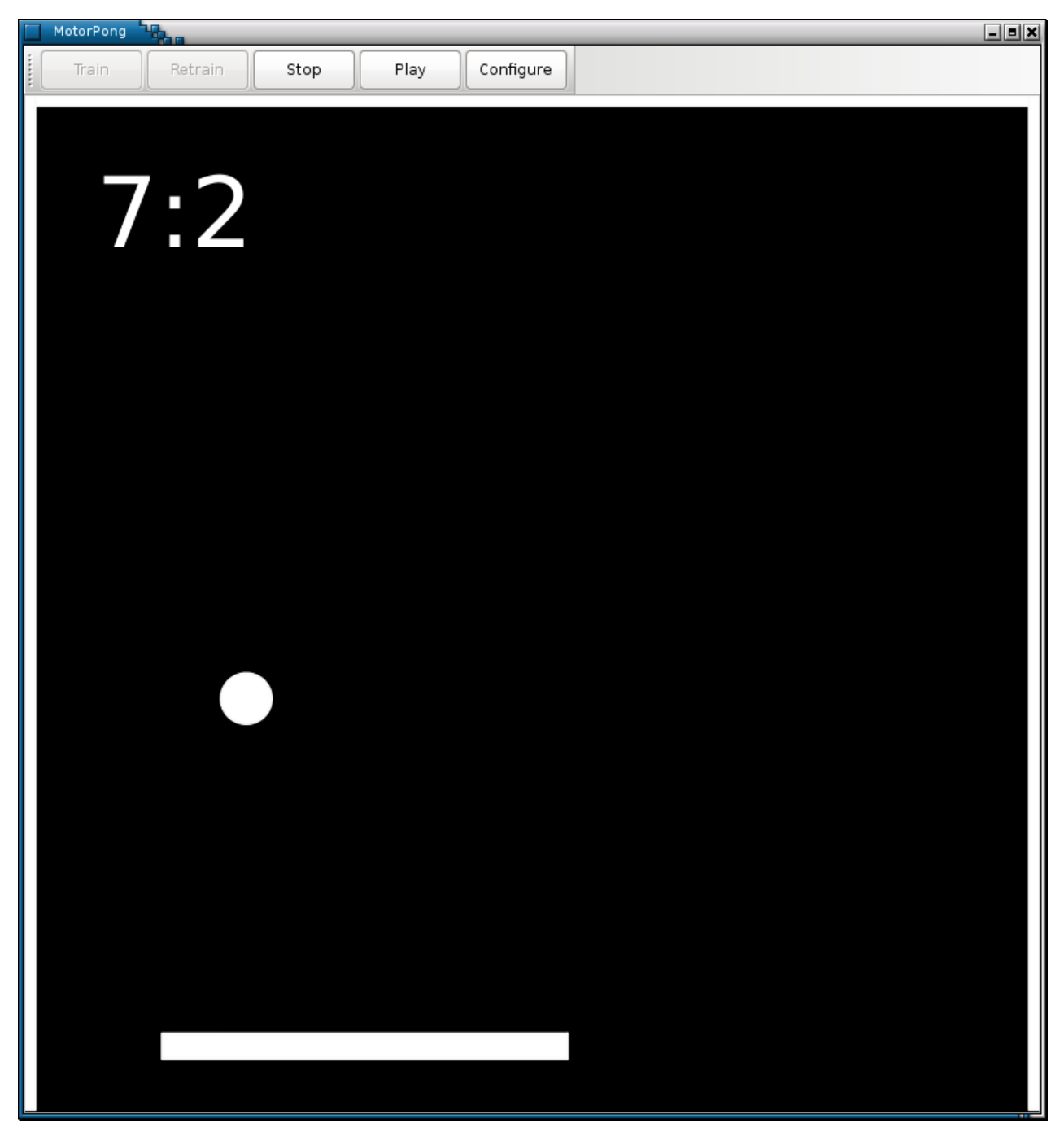
CEBL3 was designed to be versatile enough to support a number of BCI paradigms.



We have implemented a number of standard and novel BCI pages.

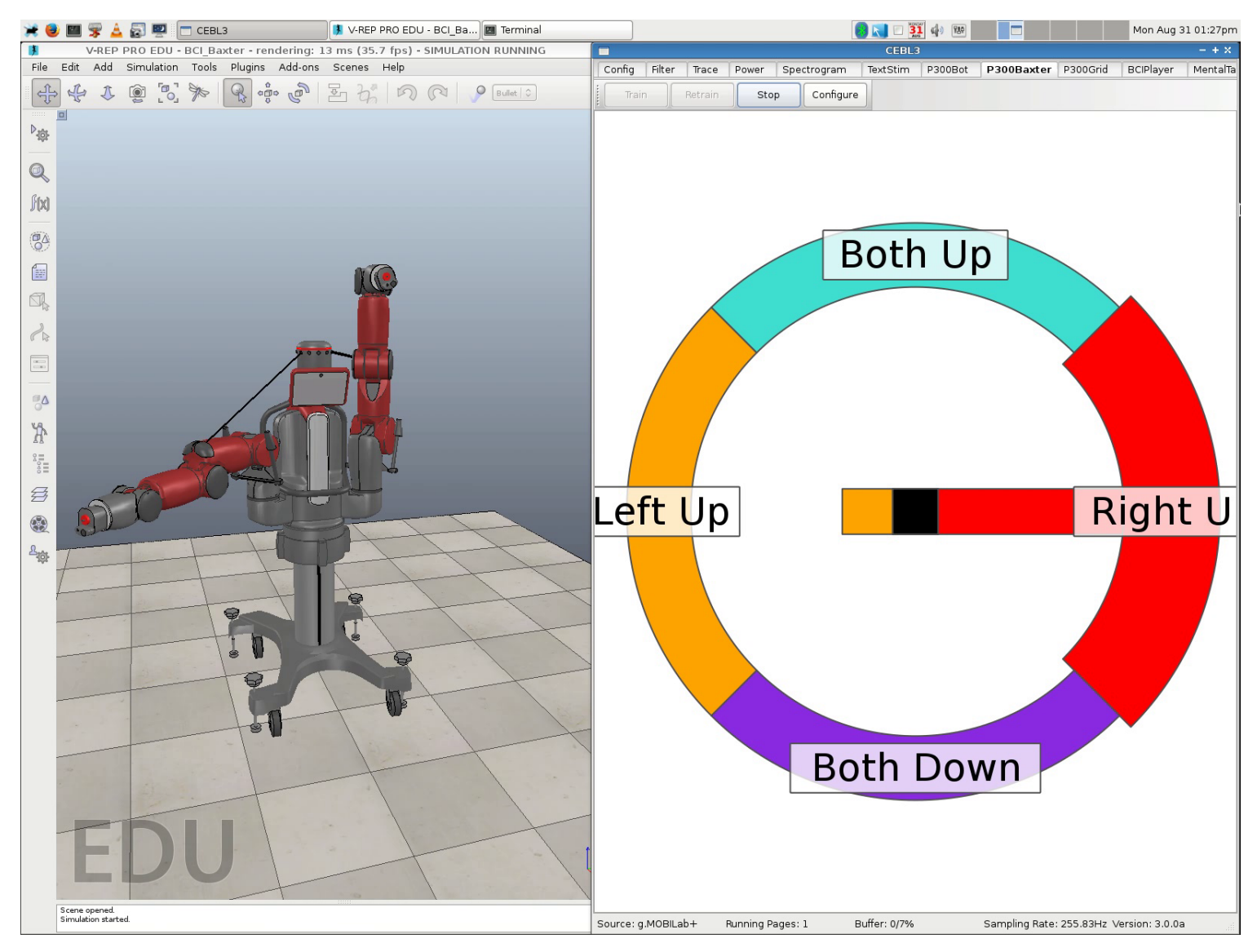
A P300 Grid Speller follows a layout similar to model smart-phone keyboards and includes options for adjusting timing, classifiers and color scheme. This provides flexibility for those with vision and cognitive impairments.

A Pie-Menu interface can be controlled using mental tasks or a P300-style paradigm. Bars grow toward a user's selection to provide immediate feedback.



After learning to control a motor imagery task using a pie-menu interface, users may attempt to control a version of the popular "pong" video game.

We have also developed a multi-media player that can be controlled using a P300-style pie-menu. A user is able to select their choice of music using only a few degrees of control. This requires minimal concentration and may be more tolerable than flashing rows and columns.



Using a similar idea, we have also constructed interfaces for controlling mobile robotics platforms, including an ER1 robotic cart and a Baxter robot using the VREP robotics simulator.

By leveraging more goal-directed behavior, we hope that we will eventually be able to control various types of robotics and prostheses.