BMI CyberWorkstation: a Cyberinfrastructure for Collaborative Experimental Research on Brain-Machine Interfaces

Prapaporn Rattanatamrong, *Student Member*, *IEEE*, Andréa Matsunaga,
and José A. B. Fortes, *Fellow, IEEE*

***Abstract*—This paper describes the design and implementation of an improved version (2.0) of a computational cyber-infrastructure for neuroscience research, called CyberWorkstation (CW). CW can provide to neurophysiology laboratories the following: (1) data storage for large volumes of neural signals, experimental parameters and computational results, (2) integration of necessary experimental equipment, powerful computational resources and robust software mechanisms that enable users to conduct online and offline BMI experiments, (3) a Web-based interface that permits users to conveniently setup, monitor and review their experiments and collaborate with others in analyzing and developing their research findings. The capabilities of the CW in enabling collaborative BMI research are demonstrated using forward models based on neural networks that predict positions of an agent in 2D movement control.**

***Index Terms*—Brain-Machine Interfaces, Collaborative Computing, Cyberinfrastructure, CyberWorkstation.**

I. Introduction

THE objective of Brain-Machine Interface (BMI) research is to understand the mapping from a brain's neural activity to the subject's intention to control an external device. BMIs are a key technology for assisting people who have lost their neurological capabilities (e.g., paraplegics) to regain their abilities by utilizing their thoughts. In addition, BMIs have the potential to enable many kinds of cognitive control applications, e.g., virtual worlds, video games, active car safety systems, and mental text entry systems [1], [2]. Experimental BMI research is also essential for the understanding of how the brain recovers from injury and repairs itself to enable people with severe head injuries to regain their lost neurological abilities.

An infrastructure that provides sufficient computational capacity to facilitate real-time modeling of interactions between multiple brain subsystems, learning, and behaviors must meet several requirements, including timing constraints, highly concurrent execution of system components and operation through simple and intuitive user interfaces.

Our prototyped computational infrastructure for supporting experimental research on BMIs, called CyberWorkstation (CW) 1.0 is described in [3], [4], and [5]. It proves the concept of BMI control schemes, such as Recursive Least Square (RLS) and Reinforcement Learning-based BMI (RLBMI), being implemented and tested in online and offline closed-loop experiments on a campus network setting. One of the CW 1.0 goals was to enable faster computation (and experimentation) than possible in typical neurophysiology labs [4].

Feedback from users of CW 1.0 indicated that major limitations of CW 1.0 are its lack of support for BMI models implemented in languages other than C++ and the tight coupling between client and server sides of CW. In this paper, the design and implementation of CW 2.0 to address these needs are presented. CW 2.0 enables users to rapidly move from concept to real-time experimentation by providing support for BMI models implemented in MATLAB, the most commonly used language by the computational BMI research community. Throughout the rest of this paper, the term "CW" refers to CW 2.0, unless stated otherwise.

Case Study: Research towards the creation of realistic computational models of the sensorimotor system and implementation of *in silico*/biological co-adaptive symbiotic systems is complex, and involves a multidisciplinary collaboration of scientists and engineers from educational institutions across the country. A collaborative research platform is much needed to enable efficient synchronous and asynchronous collaboration among researchers. CW is being designed and implemented to allow participants to share resources and findings, conveniently develop research ideas into sensible computational models, mutually conduct real-time closed-loop BMI experiments combining geographically

Manuscript received August 20, 2010. This work is supported in part by the National Science Foundation under Grant No. CNS-0540304, CNS-0821622 and the Defense Advanced Research Projects Agency (DARPA) Defense Sciences Office under the auspices of Dr. Geoffrey Ling through the Space and Naval Warfare Systems Center, Pacific Contract No. N66001-10-C-2008.The authors also acknowledge the support of the BellSouth Foundation. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the Department of Defense, the NSF or the BellSouth Foundation.

Distribution Statement "A"

P. Rattanatamrong, A. Matsunaga and J.A.B. Fortes are with the Electrical and Computer Engineering Department, University of Florida, Gainesville, FL, 32611 USA (email: {rattanat, ammatsun, fortes}@ufl.edu)

COLLABORATECOM 2010, October 9-12, Chicago, USA

DOI 10.4108/icst.collaboratecom.2010.42

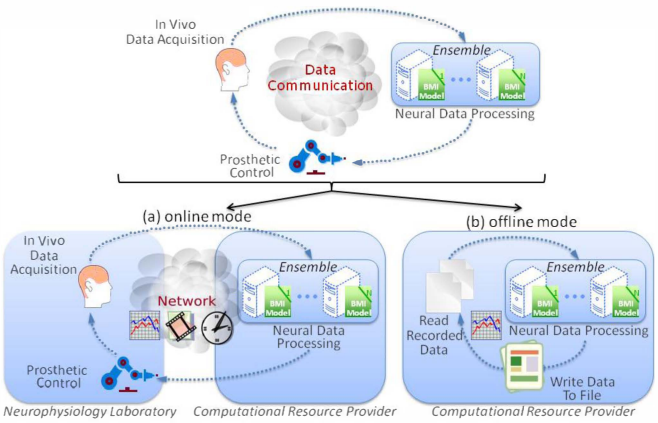

Fig. 1. High-level view of a closed-loop BMI experiment.

distributed resources, store collected neural signals and experimental results in an easy-to-retrieve manner, and perform post-experiment analysis.

The remainder of this paper is organized as follows. In Section II, background on BMI experimentation and related work on experimental research systems are provided. Section III explains the challenges and requirements in providing an efficient cyberinfrastructure for collaborative BMI research. Section IV describes the design and implementation of CW. Section V demonstrates capabilities of CW 2.0 when running an ensemble of forward models. Finally, conclusions are provided in Section V.

II. Background and Related Work

A. Closed-Loop BMI Experimentation

In a typical closed-loop BMI (Fig. 1) experiment, there are three main functional phases: data acquisition, data processing and prosthetic control.

In the data acquisition phase, in-vivo brain signals are sensed from a live subject, and converted from analog signals to digital signals. The signals are then processed by an online digital signal processing (DSP) system that detects spikes, and sorted by an algorithm that uses waveform shapes to distinguish neural activities of one or more neurons from background electrical noise. The complexity of real-time spike sorting can vary quite drastically depending on the types of neural data and research purposes [6]. The fusion of multimodal data may also be necessary if there are multiple sources of neurophysiological data.

The data processing phase turns the sorted spike sequence into the appropriate motor-control commands using a neural decoder that recognizes meaningful patterns of neural data. When studying complex neurological tasks, such as arm and hand movements in 3-dimensional space, an ensemble of concurrent trajectory decoding models may be used instead of a single complex model [7], [8]. Such an approach enables real-time neural decoding by requiring only the computation of a subset of the simple models in the ensemble.

Next, the results of neural decoding (i.e., control commands) are sent to control the prosthetic device according to the subject's intended actions. Signals that capture the behavior of the prosthetic device (e.g., its trajectory and end position) and, in some cases, resulting environment changes (e.g., displacement of an object) are provided as feedback to the live subject and the ensemble of neural decoders. The feedback information enables the subject to decide on future actions and the brain models to adapt, completing a cycle of the online closed-loop BMI experiment.

Overall, the time taken by each BMI cycle includes the time taken by the above-described three tasks plus the time needed for the communication among tasks. The communication phase of the closed-loop BMI refers to the time required for such data exchanges — between the end of data acquisition phase and beginning of neural data processing, between the end of data processing phase and the beginning of the prosthetic device control phase — to take place (the transfer of information between the prosthetic control and the subject occurs in parallel with the prosthetic control phase). Based on observed acceptable action-to-perception time lags in humans and other mammals, the time spent from the data collection to the subject's feedback must be within 100 ms [4].

Experimentation can take place in two modes: online and offline. An online experiment (Fig. 1a) has high priority – the computation provider must meet all user-specified deadlines to avoid unacceptable delays with a live subject. In addition, users can evaluate the progress of models in real-time during the online experiments. An offline experiment (Fig. 1b) has low priority due to the absence of hard deadlines, and it is typically intended to try new signal processing techniques on existing data.

B. Development of Neural Prosthetic Systems

There exist various commercial and free solutions providing generic frameworks, systems and software toolkits for the development of Brain Computer Interfaces (BCIs), of which BMIs are particular cases. These solutions utilize modularity and abstraction to enable the flexible development of each key component in neural prosthetic systems and the sharing of neural decoding algorithms among researchers. Examples of MATLAB-based solutions include BioSig [9], Virtual Integration Environment (VIE) framework [10], gTec's g.BCIsys [11], and BCILAB [12]. C++-based frameworks that also support execution of MATLAB code include BCI2000 [13], Bio-Feedback Software Development Kit (BF++) [14], BCI++ [15], and OpenVibe [16].

CW offers users a generic framework for the development of BMI ensembles, and a MATLAB-based environment to develop and run customized BMI models on remote resources. Some of the existing comprehensive toolkits that are compatible with CW framework with the respect to model development can also be integrated into CW to expedite the model development even further. In addition, CW goes beyond previous work by allowing parts of the system to be distributed across wide area networks and by providing users the capability to run BMI experiments that uses mixtures of parallel BMI models in a shared pool of resources.

C. Collaborative Experimental Research Systems

Cyberinfrastructures for collaborative experimental research have been previously proposed, especially in the field of biomedical science [17], [18]. The emphasis of these systems is mainly on secure data sharing and data-intensive experiments that integrate applications and database queries. The Biomedical Informatics Research Network (BIRN) [19] promotes large-scale collaborations in brain imaging studies of human neurological disease and associated animal models by providing data sharing, query and analysis tools. The cancer Biomedical Informatics Grid (caBIG) [20] provides access to information infrastructure to share data and analysis algorithm among collaborators in order to build new approaches to detect, diagnose, treat and prevent cancer in patients. The Integrative Biology (IB) [21] project offers a grid infrastructure specifically designed to support collaborative research for heart and cancer modeling.

While our goal is partially similar to the goal of these systems in terms of using cyberinfrastructure to foster collaborative research, to the best of our knowledge, CW is the first cyberinfrastructure specifically designed for BMI research, handling an additional requirement that was not addressed in other collaborative systems – the ability to support online experiments involving closed-loop operation with live subjects and prosthetic devices.

III. Cyberinfrastructure for Collaborative Experimental Research on BMIs

In general, the term "cyberinfrastructure" denotes "an integrated engineering infrastructure with powerful computer resources; well-preserved collections of scientific data; online experimental instruments; convenient software toolkits for modeling and interactive visualization; and support for collaborative work by physically distributed team members using all of these capabilities" [22]. The goal of this work is to build a cyberinfrastructure that leverages and integrates existing resources, services and applications to meet the unique requirements of BMI research. Challenges in building a cyberinfrastructure for collaborative experimental research on BMIs are summarized in the following subsections.

A. Research Data and Model Repository

A variety of BMI models and ensembles have been proposed by distinct research groups. When these groups collaborate, it becomes essential for participants to share not only the models (code) and the ensembles (code combining models), but also the configuration of such experiments (e.g., initial values of model parameter, formats of input data or neuron activity data, expected output, metadata about acquired data, and variables that should be monitored).

To facilitate the sharing of data, foster code reuse, and allow easy reproducibility of experiments, the cyberinfrastructure needs to provide a systematic way for users to publish their models, ensembles and data for use by other researchers. The research data warehouse is responsible for tracking the flow of information among users, possibly offering versioning control features.

B. Customizable BMI Models

Provided that models are reused by different experiments and ensembles, the cyberinfrastructure needs to be generic, supporting the creation of new experiments and ensembles without manual reconfiguration of the underlying infrastructure. Furthermore, interfaces for easy extension and redefinition of models need to be provided.

C. Real-Time Communication of BMI components

In a typical neurophysiological lab setting, the time taken by communication among components of closed-loop BMI experiments (as shown in Section II.A) is negligible since all components are collocated. In a more generic setting, the locations of the data acquisition, data processing and prosthetic control can be distinct and geographically distributed, potentially introducing significant network communication delays, which can violate the real-time requirements of experiments.

In addition to satisfying the real-time requirements of the closed-loop experiments, sequential ordering of data delivery needs to be guaranteed and data loss must be minimized. Most neural decoding algorithms involve feature extraction and translation that require analysis of time-series data; loss of data samples or out-of-order data samples incurred in communication from the data acquisition component to the data processing component can cause undesired effects on the neural decoder. Similarly, commands from the data processing component to the prosthetic control component can be lost, or be delivered out-of-order, which could potentially result in misleading feedback to live subjects or undesired movements of prosthetic devices.

D. Flexible Experiment Composition and Control

The cyberinfrastructure must provide convenient ways for users to compose their BMI experiments using their models and available models from others. Workflow representation can be used to describe and share experiments description among BMI researchers. At the same time, workflow representation can provide detailed specifications for the underlying infrastructure to manage data movement and model execution.

E. Integrated Analysis Platform

When experimenting with new BMI modeling approaches, usually not much is known about the nature of the collected brain-activity signals, and therefore, also not much is known about how to choose model parameters. The cyberinfrastructure should provide parameter visualization in real-time to help experimenters in tuning their parameters and validating the experiments' neurophysiological plausibility.

F. Parallel Processing Capability

There is strong evidence that different brain areas are involved in the computation of motor control commands [23].

Several BMI publications [24]-[26] emphasize the advantage in using ensembles of decoder models in providing superior performance in brain decoding over single-model BMIs.

Since the concurrent execution of many BMI models can require large amounts of computing power and storage capacity, proper allocation of underlying resources and run-time management to ensure timely processing of models are also considered essential aspects of the cyberinfrastructure.

G. Collaborative Communication Capabilities

Communication is one of the most important features for any collaborative research environment. Users need to communicate with each other and discuss their ideas to reach a consensus or come up with a solution to an existing problem. In such environment, communication can be divided into synchronous and asynchronous communication, referring to any real-time communication taking place between two parties (e.g., audio/video conferencing, instant messaging) and those with no timing requirement (e.g., calendars, emails, and discussion lists), respectively.

The reproducibility of BMI experiments and analysis are desirable features for collaborative research. Reproducibility allows researchers to reuse BMI techniques, and analysis methods validate each other's hypothesis, provide scientifically similar results for establishing known truths and developing incremental research from the established results. The cyberinfrastructure should provide a unified environment to collect and track all information exchanged among researchers in a variety of locations and forms.

H. Simple and Powerful Graphical User Interfaces (GUIs)

The cyberinfrastructure's GUI must be designed to hide complexity of its underlying mechanisms and present to users easy-to-use interfaces for setting up, conducting, monitoring and reviewing BMI experiments with minimal intervention from CW system administrator. The system should minimize or at least mitigate serious user errors and misunderstandings by providing reasonable default values, tooltips, warnings and proper assistance in usage, and by making common tasks in BMI experimentation simple enough for typical users.

IV. Proposed Design and Implementation

CW was designed to support BMI experiments with resources physically dispersed and connected through the Internet. The motivation for such architecture comes from the strong desire of research laboratories owning different types of expensive resources (e.g., live subjects, computers, and robotic arm) to collaborate and rapidly advance research. In particular, CW considers the collaboration case between Neurophysiology laboratories and computational resource providers. While Neurophysiology laboratories have live subjects (e.g., monkeys and rats), instrumental resources (e.g., electrode implants, sensors, and DSP devices) and prosthetic control devices, computational resource providers have a farm of computing resources (machines, storage, and network) capable of running applications much faster than in Neurophysiology laboratories.

In this context, CW clients run on Neurophysiology laboratories and are responsible for data acquisition and prosthetic control. Through the network, the brain-activity and sensory data collected by a CW client are transferred to the CW server, which is responsible for processing the data and returning the result in a timely fashion. To run on a collection of resources, a series of middleware components need to be designed and integrated (Fig. 2). The CW is divided into a user interface layer, a service layer, and a physical resource layer. The user interface layer is composed of portlets made available through a web portal that allow users to share data, models and ensembles, manage and monitor experiments, visualize results, and communicate with other researchers. The resource layer is composed of computational hardware (processors, network, and storage). The service layer contains modules that allow the use of the computational facility in a controlled manner, namely:

- *Experiment Engine:* launches and manages the execution of BMI models during online and offline experiments.
- *Model/Ensemble Registry:* facilitates the addition of newly developed models and ensembles into CW.
- *User Manager:* maintains information about users and controls access for authenticated and authorized users.
- *Monitor:* provides real-time statistics and information about jobs and resources.
- *Resource Manager and Scheduler:* maintains information about the resources and assigns work to resources, efficiently utilizing the underlying computing resources.
- *CW Network Library:* enables reliable data communication between CW clients and the CW experiment server.
- *Data Manager:* organizes safe and easy-to-retrieve data storage of models, ensembles, and experiments, including experiment results for post-processing.
- *Machine virtualization:* all components of CW are deployed in virtual machines to facilitate replication and migration of the services.
- *Shared File system:* all virtual machines have access to a single file system to facilitate data access from various services.

In the following subsections, the design and implementation of the main aspects of the architecture are presented.

A. Modular Model Development

CW middleware components are designed to be reused by different types of BMI experiments without the overhead of rebuilding the software infrastructure for every BMI research experiment. The key is to allow flexible and efficient reconfiguration of the CW so that different models can be easily "plugged in" for new BMI ensembles and experiments. CW offers a "plug-and-play" experiment engine and enables the generalization of models by using a BMI-model template.

The BMI model template, defined in MATLAB, allows users to implement different BMI models, which may require different inputs and outputs, in a standard manner. The

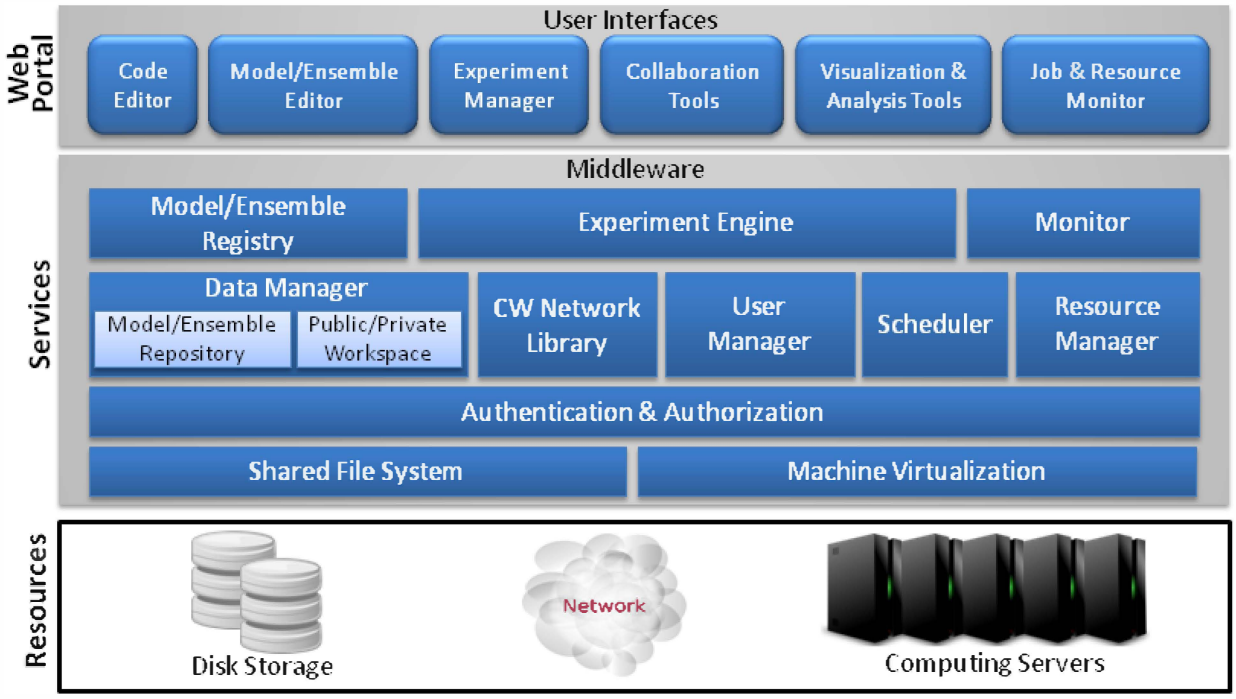

Fig. 2. Conceptual architecture of the server-side BMI CyberWorkstation. CW portal and services, hosted by computational resources providers, offer a collection of user interfaces and middleware modules that allow the parallel execution of MATLAB-based BMI ensembles for online and offline experiments on a set of shared computational resources. CW users develop BMI models, define ensembles, control experiments, analyze data, and collaborate with other researchers through the CW web portal.

template provides hooks for the user to implement initialization, algorithm and clean up routines. The template also promotes the specification of model-specific parameters to be separated from the model code, in parameter files, to facilitate the comparison of BMI model performance under different parameter settings. Similar to CW 1.0, these models can be easily combined as an ensemble in CW by defining the ensemble according to a generic abstraction. In addition, CW 2.0 provides a web interface for users to define the ensemble and does not require the server engine to be manually rebuilt every time a new model is created.

CW allows developers to share not only BMI models and ensembles, but also subroutines and toolboxes that can be reused across several experiments. For example, users could reuse previously developed signal processing algorithms and toolboxes. Details about the experiment engine can be found in [5].

B. CW Network Libra y (CNL)

To support communication between CW clients and server, data formats were defined and a software library called CW Network Library was developed. The goal of these data formats is to decouple the implementation of network communication APIs from the BMI models allowing faster integration of new models into the system. The CNL, developed as a MATLAB class, provides mechanisms for transferring data over networks through wrapper functions of the published TCP/UDP/IP toolbox [27]. In addition, the library handles necessary data marshalling and demarshalling from one format to another. An example provided later in this section gives an overview of the communication performed in CW. More details about this library can be found in [28].

In the client program and in the model processing code residing in CW server, the models' input and output are represented in the Internal Representation (IR) formats. A Mashalling/Demarshalling Procedure (MDP) carried out by CNL is responsible for transforming the input and output structures of Brain-Machine Interfaces (BMI) models in any specific IR format to an External Data Representation (EDR) suitable for transmission at the sender end and reversing this transformation for reconstructing data in the IR formats required by the receiving end (e.g., models and instruments). Each MDP is equipped with a Marshalling/Demarshalling Procedure Format (MDPF) to indicate how the MDP should perform data marshalling or packet demarshalling. The CW can automatically create basic input MDPF (iMDPF) and output MDPF (oMDPF) for users. On the other hand, the user can specify custom MDPFs which allow the MDP to achieve more efficient data communication by removing unnecessary repetitive data transfers for parameters that are shared by multiple BMI models.

Fig. 3 provides an example scenario of how CNL can be used for communication between a CW client and the CW server. The Model Collection Specification(MCS) file contains information about the input and output of models available in CW; there are five BMI models available in this example (as shown in the 'Total' line). The users develop a client program to implement their experiment using two existing models in CW. This client program calls the constructor method of the CNL's class to setup necessary network connection to the CW

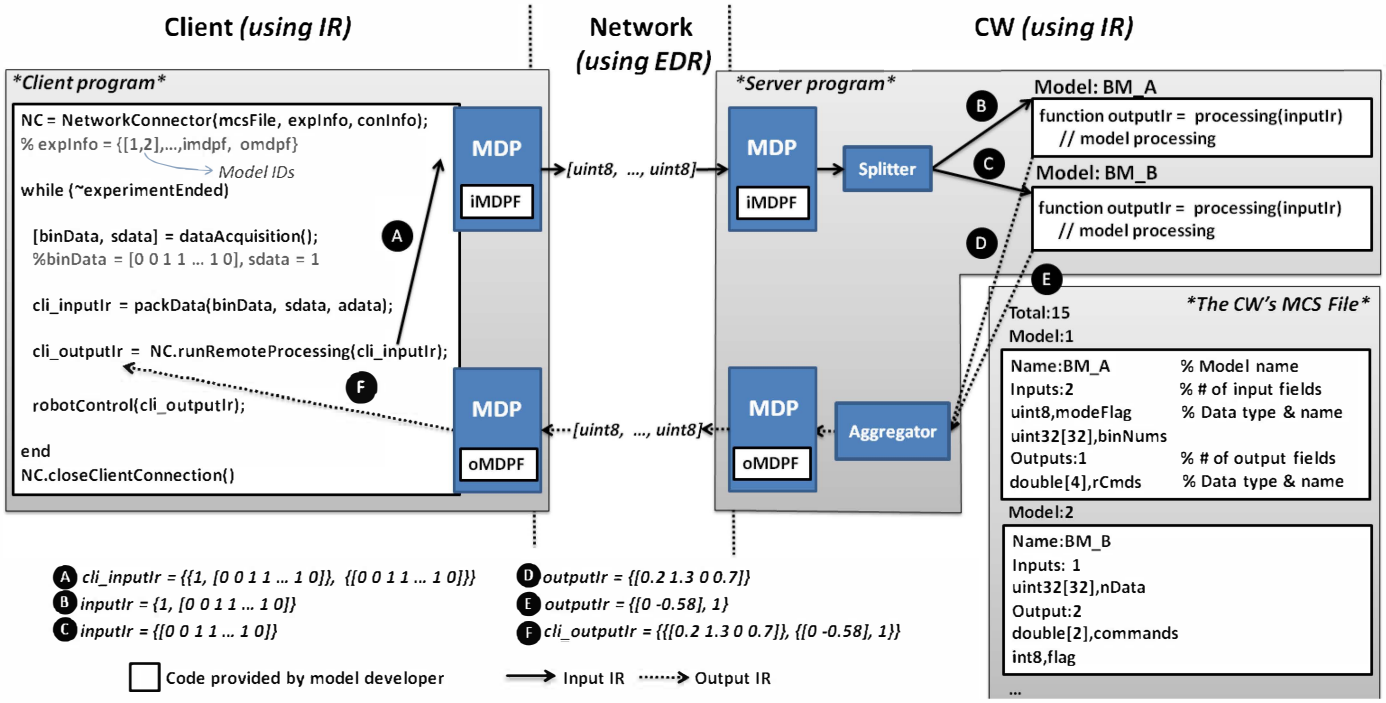

Fig. 3. CW Network Library usage for typical BMI closed loop experiment. A client program uses an Internal Representation (IR) to format acquired data and submits them to be processed remotely (A). The Mashalling/Demarshalling Procedure (MDP) packages/unpackages data as needed to be transported in the network. The CW server splits data for processing by different BMI models (B, C) as specified by ensembles created Model Collection Specification (MCS) file. Result from models are aggregated (D, E) by the CW server and returned to the client.

server and configure an experiment consisting of the BMI models 1 and 2.

During the experiment loop, the client program uses the runRemoteProcessing method of the library to send the input data (i.e., *cli_inputIr*) in the users' choice of IR formats. Four types of IR formats were designed to accommodate CW's basic to advanced usage, namely full-format, name-value, value-only and MDPF-based IR. The full IR allows users to specify the experiment input or output data fields with full flexibility; there is no restriction in the order of data fields. With the name-value IR, users have flexibility in specifying the data field within each model in any order, but have to specify each model' s data according to the specified order of models in the experiment configuration. The value-only IR provides less flexibility in terms of data field order, but it offers a concise and straightforward format for users to use. The last format, MDPF-based IR, presents another simple and condensed format, but requires a strict order of data fields according to the associated MDPF.

From Fig. 3, the CW client MDP uses an iMDPF to serialize data values from the experiment's input IR into EDR, which is subsequently packaged as part of the network packet. On the server side, the MDP transforms the content of received network packet from the EDR format into the experiment input IR expected by CW server. The server acts as a splitter which distributes data in the input IR to each participating model. Once model processing is finished, the server aggregates the output data from all models into the output IR. By using the specified oMDPF, the experiment's output IR can be packaged into and extracted from the network packet using procedures similar to those used for input data. User-specified code can also be called by the aggregator when fusion of the output data fields or complex decision algorithm is needed. After the output packet is received by the client program, MDP restores data in the packet into the IR format required by the robotControl function.

The example in Fig. 3 uses basic MDPFs and, duplicated copies of binData are used in the data transfer from the client to the server. If the users specify their own input MDPF, the second copy can be avoided and results in more efficient data transfer. Similarly, the basic oMDPF packages every output data from all participated models back to the client program (i.e., cli_outputIr). Users are allowed to leave out some data fields or combine multiple models' outputs when they specify their own oMDPFs and aggregation functions.

C. *Parallel Execution of BMI Experiments*

As shown in Fig. 4, users can create and configure an experiment through the CW portal, which subsequently creates a proper job submission script and a directory to hold related files of the newly created experiment. These files include input files (model data files for offline experiments and static or dynamic parameter files), output files and an experiment configuration file.

CW uses virtualization technology to provide the execution environment of BMI models and supports concurrent execution of BMI models using the Message Passing Interface (MPI). Each experiment consists of multiple MPI processes, each of which executes an individual BMI model. These MPI processes are managed via a cluster management system [29] in the experiment submission server. The cluster management

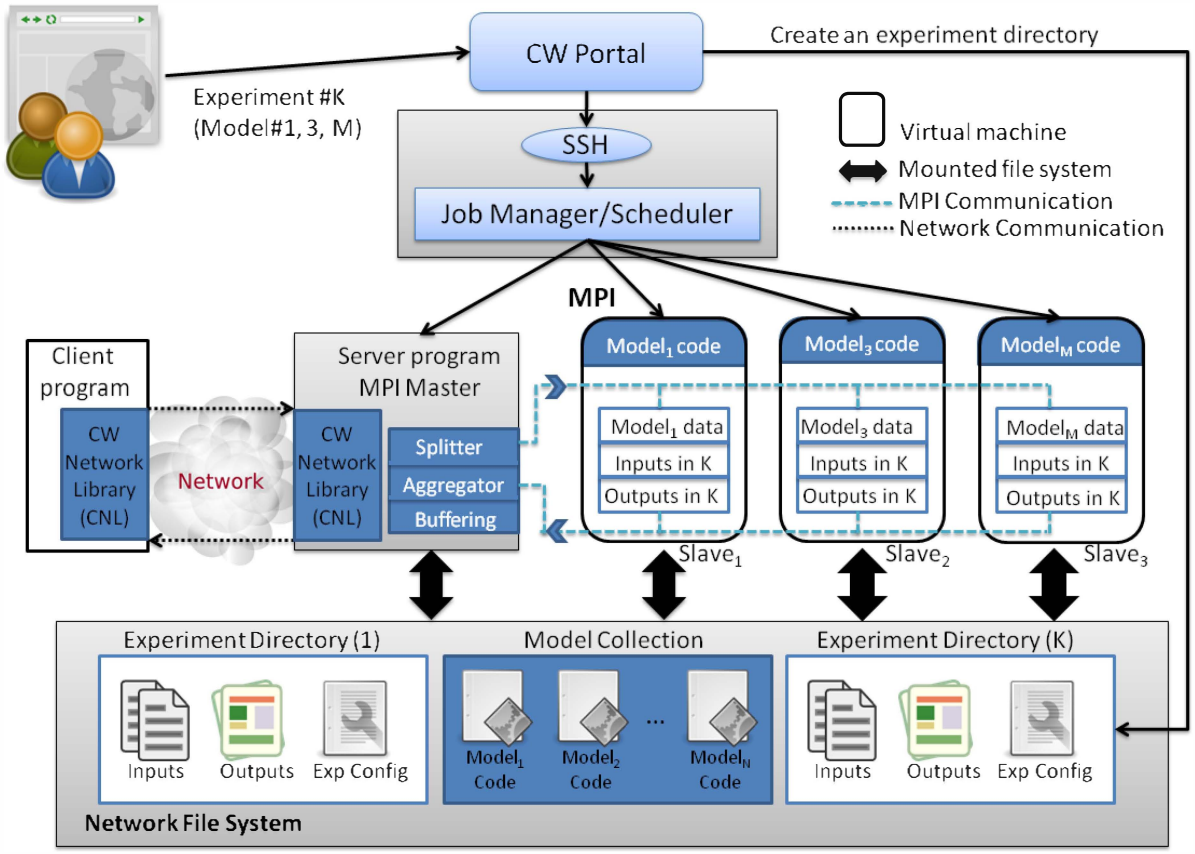

Fig. 4. Parallel execution of BMI ensembles. Users initiate the execution of an ensemble of BMI models in parallel through CW portal, which in turn requests reservation of a pool of resources using a resource manager (Torque) and a scheduler (Maui). Once the ensemble starts to run, master and workers communicate through MPI, model parameters and outputs are stored on a network file system, and online neural data is exchanged using the CW Network Library (CNL).

system provides queuing and scheduling of job executions on a cluster of virtual machines. All worker nodes are mounted on a single Network File System (NFS) holding model codes and experiment directories.

Both modes of operation (online and offline, see Fig. 1) are supported in CW. For online experiments, the neural data is sent from the client to the server using calls to functions in CNL. The master node implements a splitter, which distributes required data to each model, and an aggregator, which combines results from models, as mentioned in Section IV.B. Since an online experiment has strict timing requirements for the model computation, a set of computational resources is reserved in advance to prepare a cluster of virtual machines for the online computation. For offline experiments, virtual machines allow the model computation to share resources with other jobs in an isolated manner. The master node loads the experiment data from the model data files and buffers it in memory before the experiment begins. The experiment data is then distributed in the same manner as in the online. Priorities of VMs participating in online and offline experiments can be manually controlled to give preference to online experiments.

D. Authentication and Access Control

Distributed systems require a complex security subsystem to grant access only to authorized users. Each system component may use different authentication and authorization mechanisms making the integration a challenging task. Security mechanisms also consume resources and processing time, which is in many cases not affordable when real-time requirements need to be met. CW 1.0 made the following assumptions to simplify its security design and implementation: (1) users are registered in the portal, which authenticates users using username/password pairs; (2) users only interact with the portal, and system components communicate on behalf of users when necessary; (3) resources and data are shared among all registered users, requiring users to trust each other, and (4) cost of authentication and data encryption during online closed-loop experiments is high.

While the listed assumptions allowed for a simple implementation of CW 1.0, early users experience raised the need for a more elaborate authentication scheme. For example, different groups need isolated environments for their experiments, calling for improvements in CW.

CW places all resources in a private network, making only certain services available to the public. Most CW services are provided to users through portlets as part of a web portal. The portal authenticates users and authorization is defined on a per portlet basis. For example, the file manager and editor portlet enables the sharing of BMI ensembles and models, neural activity data, experimental setup, and MATLAB toolboxes using role-based access control, while the experiment portlet allows all users to run and manage the execution of BMI

experiments. Trade-offs between security and processing latency in online experiments are still under investigation.

E. Web Portal

CW provides a user-friendly interface via a portlet-based portal. Portlet technology allows modular development of Web interfaces and it makes it easy for the user to customize the portal layout by selecting only the web interfaces that are pertinent to a particular use case of CW. The Liferay portal framework [30] was selected amongst the open source portal frameworks available because of its clean architecture based on Java 2 Platform Enterprise Edition. Asynchronous JavaScript and XML (AJAX) technologies are used in CW's portlets to provide asynchronous and independent content updates. This technology allows for example to display balloon-style messages when defining BMI models that provide tooltip information for new users.

The experiment management portlet enables users to dynamically configure and control online experiments as well as offline studies. It interacts with other components of CW to manage the entire closed loop experiment setup. Users can monitor the status of their BMI jobs and check the availability of resources through the job and resource monitoring portlet.

The code editor portlet provides a simple web-based MATLAB code editor that allows users to quickly implement their models, possibly starting from a BMI model template, by automatically formatting code. Model developers may also include existing MATLAB routines from the public code library into their model implementation.

Various electronic communications are offered by CW portal. For asynchronous communication, each user has a personal message box for communication with peers. Additionally, users share a storage space and a forum that allows the exchange of information and data. Sharing of experiment results is another CW feature to promote sharing of findings among researchers. For synchronous communication, CW provides instant messenger portlet for real-time chatting while working through the portal.

F. Timing in Virtualized Environment

Accurate timekeeping in virtualized environments is known to be problematic due to virtualization of timing devices and the need for virtual machine monitors to serve several VMs in a single physical host [31], especially in fully-virtualized environments (e.g., VMware products). Due to the time sensitive nature of online experiments, applications need accurate time readings. When VMs can access a hardware time source without device virtualization overheads, it is possible to get time accuracy comparable to that of physical machines. An example of such time source is the Time Stamp Counter (TSC), a 64-bit register incremented every CPU clock cycle. VMs running tick-counting kernels depend on interrupt-based timer devices, and applications cannot depend on timing system calls – instead, they need to be modified to use the TSC. VMs running tickless kernels use TSC for timekeeping, and applications can run unmodified. CW uses tickless kernels whenever possible, and run modified applications, reading from TSC when the execution environment requires tick-counting kernels.

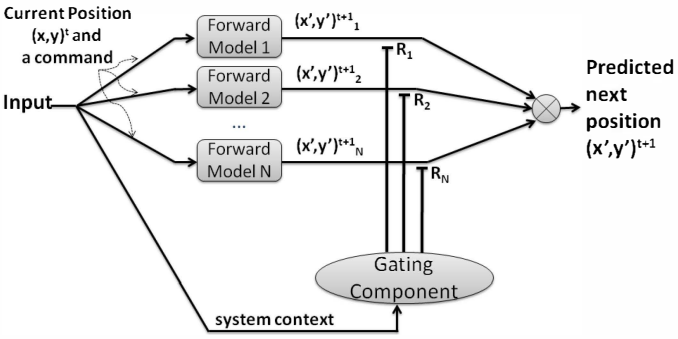

Fig. 5. Example experiment where an ensemble of forward models predicts next positions of an agent in a 2D agent movement control application.

```
function outputIr = processing(PI, inputIr)
% Get input data out from the input IR
[mode, curPos, command, trainPos, stepSize] = inputIr{1:end};
if (mode == 0)
    % Training mode
    % Store a new training sample in a batch window
    PI.storeSampleWnd(curPos, command, trainPos, -1);
    if (PI.trainWndSize() == PI.nTrainSamples)
        % When the window is full, train the model's weights
        [expertInput, desiredOutput] = PI.getTrainDataWnd();
        PI.train(expertInput, desiredOutput);
        PI.clearSampleWnd();
    end
    % Create the output IR
    outputIr = {trainPos};
else
    % Testing mode (with online adaptation)
    if (PI.trainWndSize() == 1)
        % Adapt the weight per sample using delta rule
        [expertInput, expertOutput, step] =
PI.getTrainDataWnd();
        PI.adapt(step, expertInput, expertOutput, trainPos);
        PI.clearSampleWnd();
    end
    % Calculate predicted agent's next position
    expertInput = [curPos command];
    nextPos = PI.calOutput(expertInput);
    % Create the output IR
    outputIr = {nextPos};
    % Save the data sample for next adaptation
    PI.storeSampleWnd(curPos, command, nextPos, stepSize);
end
```

Fig. 6. The forward model function used in the example experiment. The input data sent by client programs is processed by a forward model and predictions are returned to the client.

V. CW Evaluation through an Example Experiment

This section presents an example of typical experiment in BMI research where an ensemble of similar neural-network-based models with different parameters/weights is used for studying neural decoding in neurophysiological tasks. This example demonstrates CW's abilities to run and visualize BMI experiments collaboratively.

A. Example Experiment

Without loss of generality, consider an example in which users need to run collaboratively an experiment with an ensemble of 10 forward models in an application of 2D agent movement control. This experiment is based on the concept from the motor control architecture in [32]. The ensemble is a mixture-of-expert system [1] where experts implement

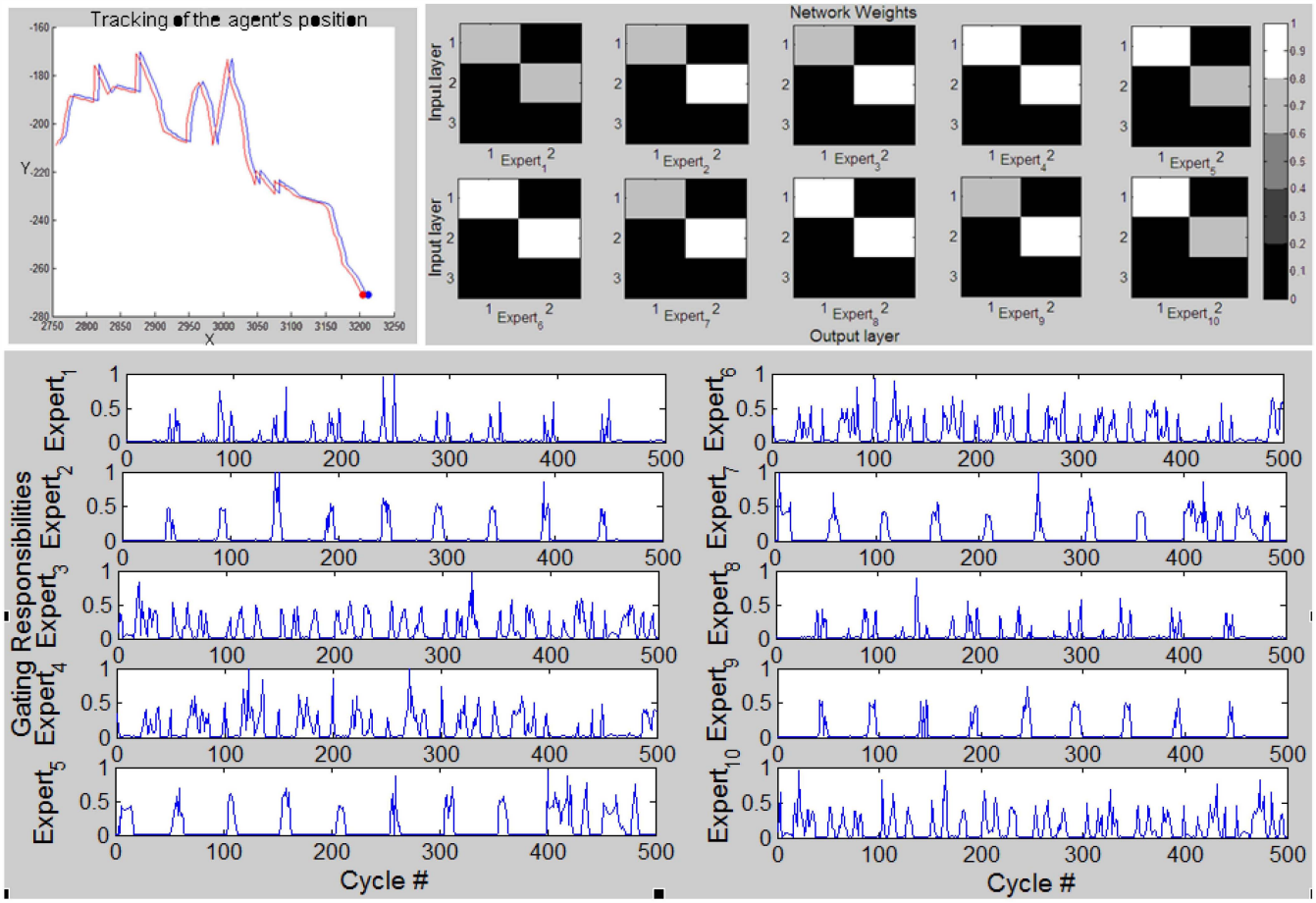

Fig. 7. Example experiment data visualization. Users can monitor the agent's next positions predicted by the ensemble and compare with the agent's actual positions [top left], view grayscale maps of the forward models' neural-network weights between a 3-neuron input layer and a 2-neuron output layer [top right], and observe a trace of gating responsibilities of all forward models [bottom] while running the experiment.

forward models responsible for predicting the agent's next position when the system is in different contexts. A gating component assigns responsibilities to these experts based on the current context of the system. The system state is defined as an (x, y) coordinate of the agent's current position. The predictions from experts with the top-2 responsibilities are then weighted combined to generate the ensemble's prediction of the agent's next position in the 2D space (Fig. 5).

B. *Model Development Utility*

For enabling the execution of the described experiment in CW, a user needs to provide an implementation of the forward model following the BMI model template (introduced in Section IV.A) and register it in CW's repository. Fig. 6 shows the implementation of the forward model's processing function in this format. This processing function takes inputs in the value-only IR format, processes them to produce the next-state prediction, and returns its result in the value-only IR format. The experiment engine calls this function when users run experiments that include an instance of this model. The CW's programming interface defined for the implementation of the processing function is simple and flexible enough to allow model developers to implement their own models without requiring manual intervention of CW administrators.

C. *Experimentation in CW*

Once the forward model is added to CW's Model Repository, a user can create a new ensemble consisting of 10 forward models with the same initial network weights. Different client programs can be developed, each with its own specific purposes (e.g., solving the next-state prediction problem under different system behaviors). To develop these clients, users simply need to make use of the CW Network Library (in Section IV.B) to communicate input and output data with the CW experiment engine. As an example, the client program can train these forward models to behave in different ways according to the context they were associated with. The ensemble can also be saved or registered as an ensemble template, which can later be selected by other researchers in the group to create a new experiment.

While running experiments, users can monitor progression of their experiments (as shown in Fig. 7) via web browsers from their workstations. Collaboration tools are also available in CW for users to discuss experiment results and share ideas to improve their findings in the experiments.

VI. Conclusion

This paper discusses the design and implementation of CyberWorkstation that serves as an elaborate multidisciplinary

research testbed for acquiring essential understanding in building highly-dependent BMIs. The procedure in setting up, conducting and analyzing online BMI experiments was generalized and a modular architecture was designed to enable the automation of this procedure by CW's middleware using virtualization technology and other adaptive management mechanisms.

The modular design of CW allows users who are not cyber-infrastructure experts to easily integrate and share new BMI models. This significantly reduces the research and development cycle by relieving the time consuming task of manually integrating new BMI models into the system.

Another unique feature of CW is to allow experiments to use resources distributed over wide-area networks. It allows state-of-the-art equipment owned by different institutions (and fields of study) to be combined into a powerful BMI system.

ACKNOWLEDGMENT

This work is supported by the grants listed on the first page. In addition, the authors would like to thank the students Diego Mesa and Pooja Raiturkar for their contributions to the web portal and to the unit testing, respectively.

REFERENCES

[1] S. G. Mason, R. Bohringer, J. F. Borisoff, and G. E. Birch, "Realtime control of a video game with a direct brain–computer interface." Journal of clinical neurophysiology : official publication of the American Electroencephalographic Society, vol. 21, no. 6, pp. 404–408, 2004.

[2] B. Blankertz, G. Dornhege, S. Lemm, M. Krauledat, G. Curio, and K. R. Muller, "The berlin brain-computer interface: machine learning based detection of user specific brain states," Journal of Universal Computer Science, vol. 12, 2006.

[3] M. Zhao, P. Rattanatamrong, J. DiGiovanna, B. Mahmoudi, R. J. Figueiredo, J. C. Sanchez, J. C. Principe and J. A.B. Fortes, "BMI CyberWorkstation: enabling dynamic data-driven Brain-Machine Interface research through cyberinfrastructure", Proc. Of the 30th Annual International IEEE EMBS Conference, pp. 646-649, August, 2008.

[4] J. DiGiovanna, P. Rattanatamrong, M. Zhao, B. Mahmoudi, R. J. Figueiredo, J. C. Sanchez, J. C. Principe and J. A.B. Fortes, "CyberWorkstation architecture for computational neuroscience", Frontiers in Neuroengineering, 2009.

[5] P. Rattanatamrong, A. Matsunaga, P. Raiturkar, D. Mesa, M. Zhao, B. Mahmoudi, J. DiGiovanna, J. C. Principe, R. J. Figueiredo, J. C. Sanchez and J. A.B. Fortes, "Model development, testing and experimentation in CyberWorkstation for Brain-Machine Interface research", Proc. Of the 32nd Annual International IEEE EMBS Conference, 2010.

[6] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials." Network (Bristol, England), vol. 9, no. 4, November 1998.

[7] B. M. Yu, C. Kemere, G. Santhanam, A. Afshar, S. I. Ryu, T. H. Meng, M. Sahani, and K. V. Shenoy, "Mixture of trajectory models for neural decoding of goal-directed movements," J Neurophysiol, vol. 97, no. 5, pp. 3763–3780, May 2007.

[8] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control", Neural Networks, vol. 11, issue 7-8, October 1998.

[9] A. Schlögl and C. Brunner, "BioSig: a free and open source software library for BCI research", IEEE Computer, vol. 41, issue 10, pp. 44-50, October, 2008.

[10] W. Bishop, R. Armiger, J. Burck, M. Bridges, M. Hauschild, K. Englehart, E. Scheme, R. J. Vogelstein, J. Beaty and S. Harshbarger, "A real-time virtual integration environment for the design and development of neural prosthetic systems", Proc. of the 30th Annual International Conference IEEE EMBS, 2008.

[11] g.BCIsys: The g.tec BCI research platform http://www.gtec.at/products/g.BCIsys/bci.htm.

[12] A. Delorme, C. Kothe, A. Vankov, N. Bigdely-Shamlo, R. Oostenveld, T. O. Zander, S. Makeig, "MATLAB-based tools for BCI research", Brain-Computer Interfaces, pp. 241-259, 2010.

[13] BCI2000 http://www.bci2000.org/BCI2000/Home.html.

[14] L. Bianchi, F. Babiloni, F. Cincotti, D. Mattia, M. G. Marciani, "Developing wearable bio-feedback systems: the BF++ framework approach", Proc. Of the 1st International Conference on Neural Engineering, March, 2003.

[15] L. Maggi, S. Parini, P. Perego, and G. Andreoni, "BCI++: an object-oriented BCI prototyping framework," in *Proceedings of the 4th International Brain-Computer Interface Workshop and Training Course*, Graz, Austria, September 2008.

[16] Y. Renard, F. Lotte, G. Gibert, M. Congedo, E. Maby, V. Delannoy, O. Bertrand and A. Lécuyer, "Openvibe: an open-source software platform to design, test, and use brain-computer interfaces in real and virtual environments", Presence Teleoperators & Virtual Environments, vol. 19, no. 1, pp. 35-53, 2010.

[17] L. D. Stein, "Towards a cyberinfrastructure for the biological sciences: progress, visions and challenges", Nature Reviews Genetics, vol. 9, no. 9, pp. 678-688, September, 2008.

[18] K. H. Buetow, "Cyberinfrastructure: empowering a third way", Science Magazine in Biomedical Research, vol. 308, pp. 821-824, 2005.

[19] J. S. Grethe, C. Baru, A. Gupta, M. James, B. Ludaescher, M. E. Martone, P. M. Papadopoulos, S. T. Peltier, A. Rajasekar, S. Santini, I. N. Zaslavsky and M. H. Ellisman, "Biomedical informatics research network: building a national collaborator to hasten the derivation of new understanding and treatment of disease", Proc. Of HealthGrid, 2005.

[20] A. C. von Eschenbach and K. Buetow, "Cancer informatics vision: caBIG", Cancer Inform, vol. 2, pp.22-24, February, 2006.

[21] S. Lloyd, D. Gavaghan, A. Simpson, M. Mascord, C. Seneurine, G. Williams, J. Pitt-Francis, D. Boyd, D. M. Randal, L. Sastry, S. Nagella, K. Weeks, R. Fowler, D. Hanlon, J. Handley and G. d. Fabritiis, "Integrative biology – the challenges of developing a collaborative research environment for heart and cancer modeling", Future Generation Computer Systems, vol. 23, no. 3, pp. 457-465, 2007.

[22] D. E. Atkins, K. K. Droegemeier, S. I. Feldman, H. Garcia-molina, M. L. Klein, D. G. Messerschmitt, P. Messina, J. P. Ostriker and M. H. Wright, "Revolunizing science and engineering through cyberinfrastructure", Report of the National Science Foundation, Blue Ribbon Advisory Panel on Cyberinfrastructure, 2003.

[23] O. Hikosaka, K. Nakamura, K. Sakai and H. Nakahara, "Central mechanisms of motor skill learning", Current Opinion in Neurobiology, vol. 12, issue 2, April, 2002.

[24] J. C. Sanchez, J. M. Carmena, M. A. Lebedev, M. A.L. Nicolelis, J. G. Harris and J. C. Principe, "Ascertaining the importance of neurons to develop better Brain-Machine Interfaces", IEEE Transactions on Biomedical Engineering, vol. 51, no. 6, June, 2004.

[25] N. Hatsopoulos, J. Joshi and J. G. O'Leary, "Decoding continuous and discrete motor behaviors using motor and premotor cortical ensembles", Journal of Neurophysiology, vol. 92, pp. 1165-1174, 2004.

[26] S. Kim, J. C. Sanchez, D. Erdogmus, Y. N. Rao, J. Wessberg, J. C. Principe and M. Nicolelis, "Divide-and-conquer approach for brain machine interfaces: nonlinear mixture of competitive linear models", Neural Network, vol. 16, issue 5-6, pp. 865-871, June, 2003.

[27] P. Rydesäter, MATLAB Central File Exchange - TCP/UDP/IP Toolbox 2.0.6, Copyright ©1997-2009.

[28] P. Rattanatamrong, A. Matsunaga and J. A.B. Fortes, "Data formats for data marshalling and demarshalling in the BMI CyberWorkstation 2.0", ACIS Laboratory, University of Florida. TR-ACIS-10-001, 2010.

[29] Torque Resource Manager. URL: http://www.clusterresources.com/pages/products.php.

[30] Liferay portal framework. [Online]. http://www.liferay.com.

[31] VMware, Inc. Timekeeping in VMware Virtual Machines. White paper, 2008. URL: http://www.vmware.com/pdf/vmware_timekeeping.pdf.

[32] D. M. Wolpert and M. Kawato, "Multiple-paired forward and inverse models for motor control", Neural Networks, 11, pp. 1317-1329, 1998.

[33] R. A. Jacobs, M. I. Jordan, S. J. Nowlan and G. E. Hinton, "Adaptive mixture of local experts", Neural Computation, 3, pp.79-87, 1991.