

Cerebellar Dynamic State Estimation for a Biomorphic Robot Arm

Christopher Assad

Bio-Inspired Technologies and Systems
Jet Propulsion Laboratory
Pasadena, CA, USA
chris.assad@jpl.nasa.gov

Sanjay Dastoor¹

Mechanical Engineering
UC Berkeley
Berkeley, CA, U.S.A.
sanjayd@berkeley.edu

Salomon Trujillo¹

Mechanical Engineering Dept.
Stanford University
Stanford, CA, USA
sjtrujil@stanford.edu

Ling Xu¹

Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
lingx@andrew.cmu.edu

Abstract - *The cerebellum has been called the brain's "engine of agility". This paper presents a cerebellar-inspired neural network that performs dynamic state estimation and predictive control. The model combines two types of learning within a radial basis function network. Its performance was demonstrated on a 2-link robot arm built with antagonistic pairs of McKibben air muscles. The arm has a gripper end effector to hold and throw a tennis ball. Trajectory data was collected during multiple throwing trials and used to train the model offline. The data were projected onto 2-dimensional state space maps, from which the network learns to estimate state variables and decision boundaries. It successfully learned to trigger the grip release at the proper state for the ball to hit a target. This algorithm should generalize to benefit a wide variety of biomorphic robots.*

Keywords: Cerebellum, biomorphic robotics, dynamic state estimation, McKibben actuators, state space methods

1 Introduction

Animals are very good at maneuvering and manipulating objects in unstructured complex environments, tasks still not achievable with state-of-the-art robotics. They typically possess dynamic, nonlinear and high degree-of-freedom (DOF) bodies that are intractable to conventional control methods. The field of Biomorphic Robotics looks to gain inspiration to build more dexterous, agile robotics by emulating the biomechanics and highly effective nervous structures and nonlinear control algorithms found in biology. Important differences from conventional robotics include (1) exploiting the body's natural dynamics, (2) "springy" compliant actuation for efficient force control and energy recovery, (3) no precision sensors; instead, large arrays of fast and cheap "sloppy" sensors, for over-sensing for kinesthetics and proprioception, (4) no precision machining; instead, reliance on adaptive control to learn dynamics and adapt to

changes over time, and (5) intelligent coordination of feedforward and feedback control strategies.

Research on biomorphic robots has been largely inspired to date by examples from insects, other arthropods, and spinal reflexes in vertebrates. Biomorphic control algorithms include central pattern generators for gait coordination and low-level reflexes such as muscle stretch reflexes and balancing reflexes [1-4]. While these low-level controls may be sufficient for insect locomotion, they do not scale up sufficiently for larger animals or robots, because the dynamics become more important due to greater mass, length, and required forces (e.g., moments of inertia scale as length to the fifth power). Agile control under these conditions requires accurate dynamic state estimation (DSE), a fundamental component of a wide variety of sensor fusion, signal processing and control tasks in engineering.

We take inspiration from the next level of control found in the nervous systems of vertebrate animals. Atop the spinal cord sits the cerebellum, the brain's "engine of agility" [5]. It has a unique neural architecture that appears optimized for learning sensory-motor dynamics and predictive control of high DOF nonlinear systems. Although details of its function are not fully understood, the cerebellum is thought to perform DSE to achieve dexterous, coordinated and dynamically efficient motor output [6]. Our work at JPL has been to develop algorithms to capture the functionality of the cerebellum. We have begun to simulate, implement and deploy these algorithms for dynamic control of biomorphic robots [7,8]. Here we describe the cerebellum model and its application to a biomorphic robot arm built as a demonstration platform. The cerebellar algorithms should be general enough to facilitate a wide range of dynamic robotic systems requiring sensory-driven motor control, such as limbed rovers, legged walkers, UAVs, telerobotic platforms, and even exoskeletons or prostheses.

¹ Authors ST, SD, and LX worked at JPL under the NASA USRP and Caltech SURF summer student research programs.

2 Cerebellum model

2.1 Background

The cerebellar cortex has a unique, massively parallel, modular architecture with three layers (Fig. 1): a granule cell layer receiving mossy fiber inputs, a layer of large Purkinje output cells (PC), and a molecular layer where parallel fibers run through the large planar PC dendritic trees. Each parallel fiber contacts up to several hundred PCs along its length, and each PC receives on the order of 100,000 excitatory parallel fiber inputs. Each PC also receives input from a single climbing fiber, which makes multiple synapses on the PC proximal dendrite and ensures the firing of a complex calcium spike in the PC dendrite whenever the climbing fiber fires. The five major cell types (including 3 inhibitory interneurons) are arranged in repeating subunits, often described as crystalline in regularity; therefore this basic microcircuit is considered representative of the entire cortex. Each region of cerebellum monitors its own mix of descending motor commands and ascending sensory and proprioceptive feedback. Synaptic plasticity allows it to learn an implicit model of the body dynamics based on correlations in its inputs; then during replay of learned behaviors, the cerebellum predicts the consequences of motor actions, compares expected states to incoming sensory data, and modulates motor output.

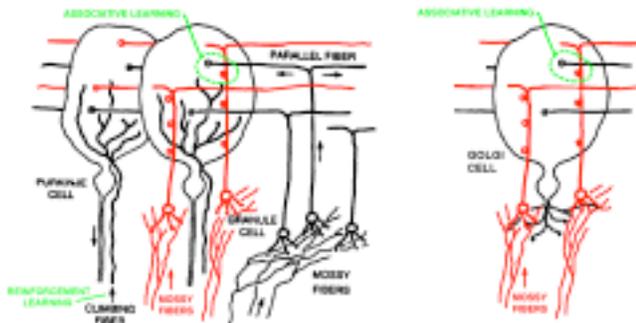


Figure 1. Functional microcircuitry of cerebellar cortex (adapted from [12]).

Most modern models of cerebellar function are descended from those first proposed by Marr in 1969 [9] and Albus in 1971 [10], and still rely on the traditional cerebellar microcircuit framed by those early works. But the wide diversity of current competing models reflects the lack of consensus on how the cerebellum works. Our approach is similar to that described by Keeler [11] for Marr-Albus-Kanerva type cerebellar networks [12], in which the model acts as a sparse distributed associative memory: information from sensory, proprioceptive, and cortical inputs (such as motor commands) provides the context from which to learn to predict future states. However, we are extending the model's function based on new neuroanatomical and neurophysiological evidence for learning mechanisms from cerebellar, vestibular, and electrosensory systems, as well as guidance from control systems analysis [6,7].

2.2 Modeling methods

The neural network model has three groups of neurons, corresponding to an input granule cell layer, an output PC array, and Golgi cell interneurons (Fig. 2). The granule cell layer is implemented as a radial basis function (RBF) network, where each unit responds to stimuli in a localized region of state space. This generates a sparse distributed encoding of the state variables. Because RBFs break the problem domain into many small pieces, their weighted outputs can be used to approximate any nonlinear function. However, the price is in the large number of units needed to cover large workspaces. In typical real systems, only very small subregions of the input space are actually visited where the data needs representation. Current research on RBF networks is exploring adaptive methods to concentrate the receptive fields most efficiently for highest resolution in appropriate subregions of the workspace. However, for simplicity we began with uniformly distributed input layers in 2-dimensional projections of state space (see section 4.2).

Each PC or Golgi cell receives direct information about a particular state variable from the granule cells below it, and a large set of inputs from the parallel fibers with variable delays, representing contextual state information. Correlated activity between direct and contextual inputs causes Hebbian associative learning; i.e., the cell learns to respond the next time it recognizes the same contextual pattern. The Golgi cells provide feedback to the granule cells to filter their activity, in effect weighting the relative contribution of the new sensed input on the implicit system model. The PCs also receive error feedback from climbing fiber inputs. These inputs provide supervisory signals for reinforcement learning, allowing the cell to map patterns of activity into desired motor outputs.

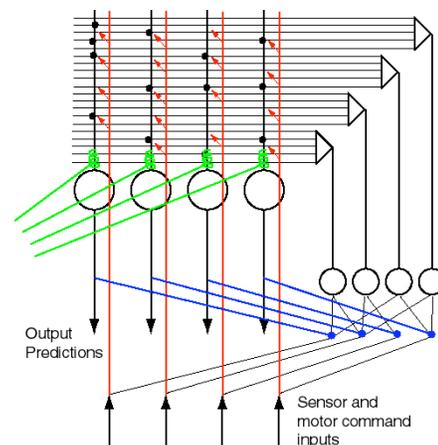


Figure 2. Neural network schematic

3 Biomorphic Arm

Over the past several summers, students in our lab at JPL have built and refined a dynamic two-link planar arm that can throw a tennis ball across the room. We wanted a dynamic platform that allowed exploitation of the

mechanical body dynamics. We chose an arm for several reasons: the cerebellum is known to be involved in multi-joint coordination; results here should generalize to a variety of limbs and body mechanics; research data is available on cerebellar lesion patients in throwing tasks [13,14]; and baseball pitchers have become the poster boys for cerebellar research [5]. The arm is nicknamed “The Arm of Uma” after the Hindu goddess of education.



Figure 3. Early version of the Uma Arm illustrating antagonistic pair of McKibben muscles at shoulder and elbow joints. Here the muscles are partially inflated and the arm has reached an equilibrium position.

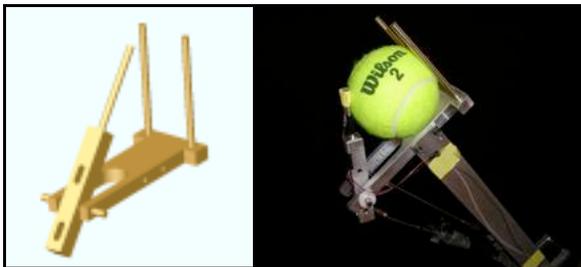


Figure 4. The revised gripper.

3.1 Actuation

Each joint of the arm is actuated with an antagonistic pair of McKibben air muscles. Also known as rubbertuators or braided pneumatic actuators, McKibben’s are fast and strong with muscle-like dynamics but are very difficult to control by conventional means [15]. The device consists of a rubber tube inside a braided mesh. When the tube is pressurized, the mesh contracts along its length. McKibbens have little to no inherent damping, and muscle dynamics are highly nonlinear and change with use, including temperature effects, etc. Several arms have been built with these in the past, but most results from these arms were in tasks of position control [16-18]. McKibbens can be used as variable spring constants to control compliance and mechanical response properties.



Figure 5. McKibben actuators

McKibben actuators are inexpensive to build in the lab. We followed a procedure from Dan Kingsley in the Biorobots lab at Case Western University. Each muscle had two air valves, one for inlet and one for exhaust, so air can be trapped to modify stiffness. The system was run with a compressed air source between 65-75 psi.

3.2 Sensors

Several sensor types were built into the arm to provide feedback for the control algorithms. One potentiometer was attached inline with each joint axle to provide a voltage signal proportional to joint angle. For joint velocities, numerical differentiation of collected angle data proved impractical due to spike noise in the electronics and A/D resolution. (No extra effort was made to limit noise sources in the system, because the algorithm is meant to account for imprecise actuation and sensing.) Therefore we built an analog filter (two opamp circuit) to differentiate the signals in the frequency range of interest and provide a voltage approximately proportional to joint velocity. A force-sensitive resistor on the gripper “thumb” recorded the pressure applied, from which the time of ball release could be determined. The air muscles also have pressure sensors monitoring their inlet tubes, but these were not used for the modeling study presented here.

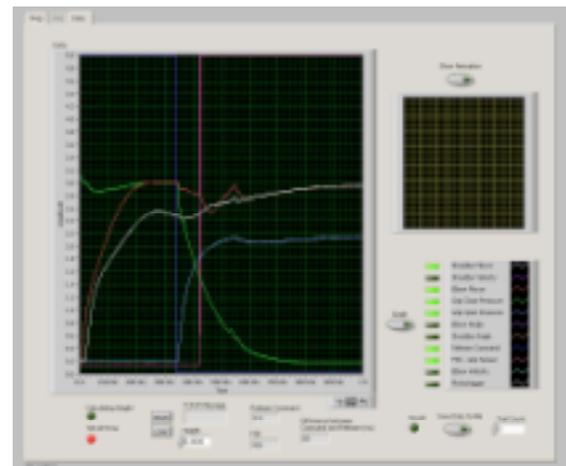


Figure 6. Sample sensor data collected in Labview.

3.3 Computer interface and control

We constructed a Labview interface to control valve sequencing and data collection. To generate a trajectory appropriate for throwing, the air valve timing for the shoulder and elbow flexors was chosen after a few trial and error throws (Fig. 7). During each throw, an A/D card collects joint angle, joint velocity, and grip sensor data at 1 ms intervals. A spare A/D channel was used to monitor the release command where it reached the air valve bank. At the end of each throw, the height at which the ball reached the target plane had to be determined to provide feedback for the reinforcement learning. A USB webcam imaged multiple frames across the plane of the target during the throw. The images were processed in MATLAB to determine height with about 1 cm resolution.

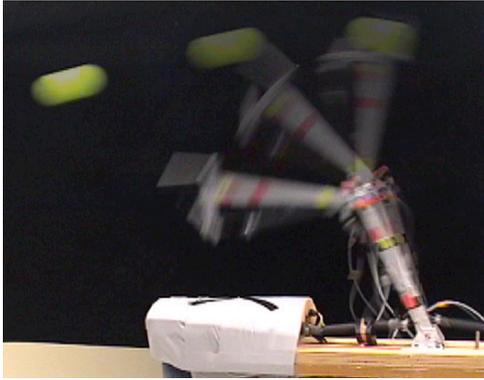


Figure 7. Three frames (50 ms apart) at time of release.

4 Learning to throw with accuracy

4.1 Problem Statement

Conventional robotic controllers typically work with kinematic or dynamic system models. For a multi-link manipulator, the torque equations take the following form [19], where θ is the vector of joint angles, M includes inertial terms, V velocity terms (e.g., Coriolis and centrifugal forces), G includes gravity terms, and F denotes frictional forces:

$$\tau = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + G(\theta) - F(\theta, \dot{\theta}) \quad (1)$$

For revolute joints these terms are nonlinear, and it can be difficult to accurately model parameters for friction, actuator dynamics, sensor noise and delays in feedback.

Controlling such a dynamic system requires knowledge of the system's state and its response to new motor commands, a task that naturally lends itself to state space methods of solution. The cerebellum can orchestrate dexterous, agile movements by learning: (1) to estimate and predict trajectories through state space, i.e., modeling the system dynamics, (2) decision boundaries around regions of state space in which to initiate actions to achieve desired goals, and (3) to modulate motor commands to redirect the trajectory as needed (Fig. 8). The cerebellum

estimates the current state by combining incoming sensor measurements and the implicit learned model of system dynamics, predicts the trajectory, and then can initiate actions in appropriate regions of the space.

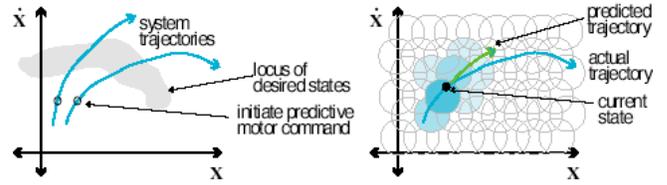


Figure 8. Problem description in state space.

The task we chose was to learn the release timing for the arm to throw a ball at a given target. In overhand throws in humans, the crucial control variable for target accuracy is the timing of the release with respect to the arm trajectory, and the cerebellum has been directly implicated in studies of patients with cerebellar lesions [13, 14]. Because there is a motor delay on the order of 50-100 ms (in both humans and with our air muscles), the motor command has to be predicted and sent ahead of the actual release. This process is better described as learning the proper state rather than the “timing” for the action.

4.2 Learning an idealized trajectory

Our simulations have demonstrated that this cerebellar model can learn temporal and spatial correlations within its stream of sensory and motor command inputs. We built relatively small cerebellar models in MATLAB, consisting of several principal cells, 4-6 state variable inputs, and a few thousand RBF units that uniformly covered 2-dimensional projections of the input state space. The projections were chosen to match pairs of joint angles and velocities found in the torque equations. To test the algorithm we simulated the throwing task using a virtual planar arm with one link or two links. For the two link case, trajectory state data was collected from the robotic arm during throwing trials (Fig. 9). An ideal trajectory was spline smoothed and used to train the network model offline. The results showed successful learning to predict the state variables along the trajectory, and to trigger release of the ball to hit the target (Figs. 10, 11).

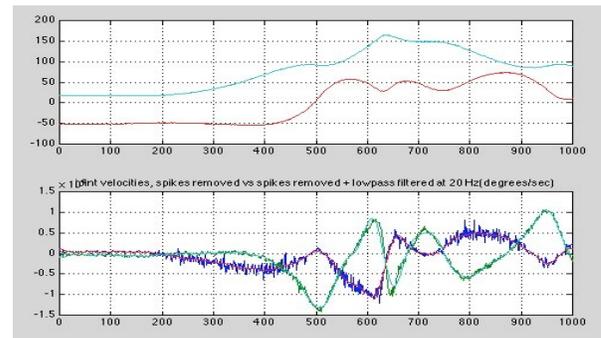


Figure 9. Joint angles (top) and velocities (bottom) from a sample throw.

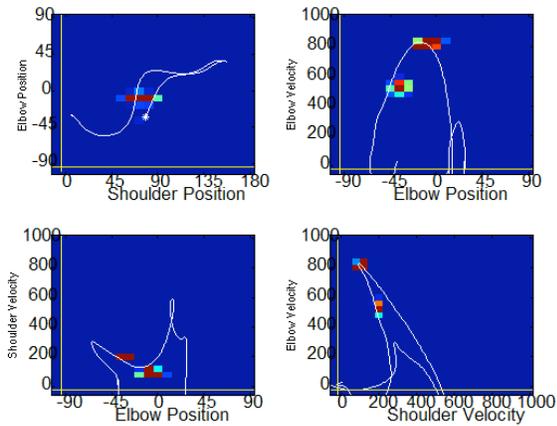


Figure 10. Trajectories in state space (white lines) and weights of RBF units after learning (colored blocks). Two solutions were found on the trajectory, corresponding to a high-arching slow pitch and a straight fastball.

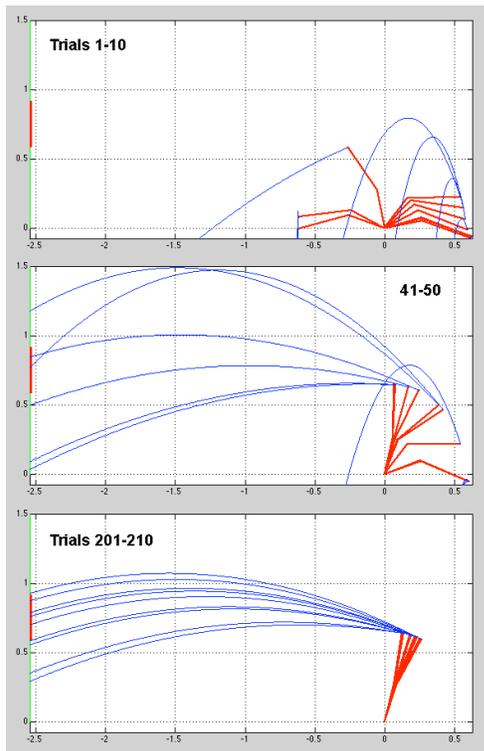


Figure 11. Simulated throw results during training on idealized trajectory.

4.3 Learning on raw trajectories and closing the loop with cerebellar control

We extended these results to the Uma Arm throwing a tennis ball at a target (distance: 2.5 m; target height 0.8 ± 0.15 m). The cerebellar network was first trained offline to initiate the throw release based on trajectory and target error data from 200 throwing trials. The learned weights were then used to approximate a cerebellar lookup table that

could be inserted into the Labview control loop. As the throw progressed, the sensor feedback was used to index into the lookup table, and an output value was read and compared to a threshold. When the output passed the threshold the release command would be initiated (Fig. 12). Figure 13 compares our results to throwing accuracy in humans. (Note that we did not measure horizontal spread, only vertical height. The robot data are spread horizontally in the figure only for viewing purposes.) These preliminary results indicate the cerebellar model reduced the standard deviation in throw error by about 50% compared to open loop control with fixed time of release.

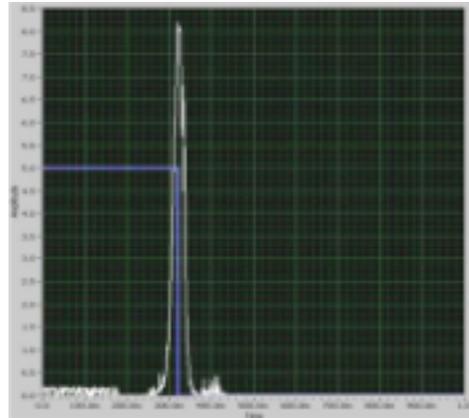


Figure 12. Example output from a Purkinje cell trained to generate the release command. Blue line indicates the release command triggered at time = 320 ms.

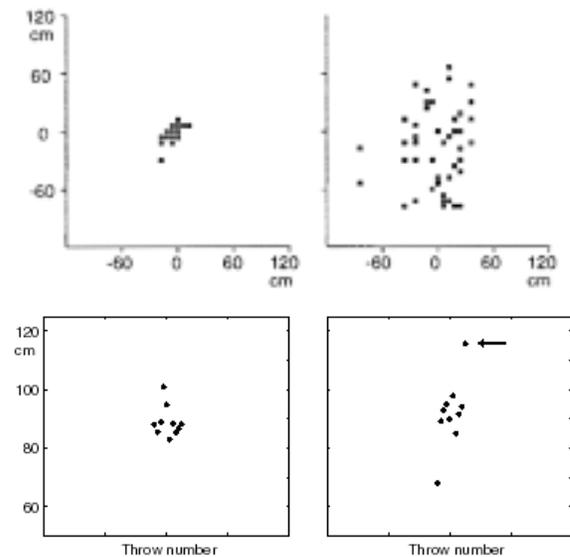


Figure 13. Scatter plot results compared to human throwing. Top left: normal human subject throwing at a target 3 m away. Top right: human patient with cerebellar lesion (from [13]). Bottom left: results from cerebellar model controlling release of the Uma Arm (note difference in vertical scale). Bottom right: open loop throwing by Uma Arm (fixed time of release, prechosen by operator). Arrow indicates a throw sampled after a delay of 15 minutes, showing a typical open-loop drift over time.

5 Conclusions

We have presented a model of cerebellar function that performs DSE by learning temporal and spatial correlations within its stream of sensory and motor command inputs. The model was applied to a throwing task on a dynamic robot arm, where it learned to estimate the state in which to trigger the grip release to accurately hit a target. We are currently continuing training on our biomorphic Uma Arm platform to measure significance of these results, and extending the model to explore generalization to varying target heights and projectile mass. Because the cerebellar model learns an implicit representation of the system dynamics, in principle it could be applied to any number of mechanical systems requiring DSE. In particular, this method should prove efficient for learning dynamic trajectories in high DOF nonlinear biomorphic robots.

Acknowledgements

Thanks to Nathaniel Chan and Dan Kingsley for building the original Uma Arm, Michael Rizk and Emily Fox for improvements in the arm and the original computer interface, and Mitra Hartmann and Mike Paulin for discussions on cerebellar algorithms. This work was supported by NASA's CICT/ITSR Revolutionary Computing program area.

References

- [1] F. Delcomyn, "Walking robots and the central and peripheral control of locomotion in insects," *Autonomous Robots* 7:259-270, 1999.
- [2] M.A. Lewis, R. Etienne-Cummings, M.J. Hartmann, Z.R. Xu, A.H. Cohen, "An in silico central pattern generator: silicon oscillator, coupling, entrainment, and physical computation," *Biological Cybernetics* 88(2): 137-151, Feb 2003.
- [3] P. Arena, L. Fortuna, M. Frasca, G. Sicurella, "An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion," *IEEE Transactions On Systems Man And Cybernetics Part B-Cybernetics* 34 (4): 1823-1837, Aug 2004.
- [4] F. Delcomyn, "Insect walking and robotics," *Annual Review Of Entomology* 49: 51-70, 2004.
- [5] I. Wickelgren, "The Cerebellum: The brain's engine of agility," *Science* 281:1588-1590, 1998.
- [6] M.G. Paulin, L.F. Hoffman, C. Assad, "A model of cerebellar computations for dynamical state estimation," *Autonomous Robots* 11:279-284, 2001.
- [7] C. Assad, "An hypothesis for a novel learning mechanism in cerebellar cortex," *Autonomous Robots*, Vol. 11, No. 3, pp. 285-290, 2001.
- [8] C. Assad, M.J. Hartmann, and M.G. Paulin, "Control of a simulated arm using a novel combination of cerebellar learning mechanisms," *Neurocomputing*, Vol. 44-46, pp. 275-283, 2002.
- [9] D. Marr, "A theory of cerebellar cortex," *J. Physiology*, Vol. 202, pp. 437-470, 1969.
- [10] J.S. Albus, "A theory of cerebellar function," *Mathematical Biosciences* 10:25-61, 1971.
- [11] J.D. Keeler, "A dynamical system view of cerebellar function," *Physica D*, Vol. 42, pp. 396-410, 1990.
- [12] P. Kanerva, *Sparse Distributed Memory*, MIT Press, Cambridge, 1988.
- [13] D. Timmann, S. Watts, and J. Hore, "Failure of cerebellar patients to time finger opening precisely causes ball high-low inaccuracy in overarm throws," *J. Neurophysiology* 82(1): 103-114, Jul 1999.
- [14] J. Hore, S. Watts, J. Martin, and B. Miller, "Timing of finger opening and ball release in fast and accurate overarm throws," *Exp. Brain Research* 103:277-286, 1995.
- [15] G.K. Klute, J.M. Czerniecki, B. Hannaford, "Artificial muscles: Actuators for biorobotic systems," *International Journal Of Robotics Research* 21 (4): 295-309 APR 2002.
- [16] C.P. Chou, B. Hannaford, "Study of human forearm posture maintenance with a physiologically based robotic arm and spinal level neural controller," *Biological Cybernetics* 76 (4): 285-298, APR 1997.
- [17] P.P. van der Smagt, F.C.A Groen, and K. Schulten, "Analysis and control of a rubber-tuator arm," *Biological Cybernetics*, 75(5):433-440, 1996.
- [18] T. Hesselroth, K. Sarkar, P.P. van der Smagt, and K. Schulten, "Neural network control of a pneumatic robot arm," *IEEE Transactions on Systems, Man, and Cybernetics*, 24(1):28-38, January 1994.
- [19] J.J. Craig, *Introduction to Robotics: Mechanics and Control*, Addison-Wesley Publishing, Reading MA, 1986.