Massachusetts Institute of Technology

Department of Electrical Engineering and Computer Science

Proposal for Thesis Research in Partial Fulfillment

of the Requirements for the Degree of

Master of Science

Title: Learning Functional Mappings Between Sensorimotor Systems on a Humanoid Robot

Submitted by: Matthew J. Marjanović
MIT Artificial Intelligence Lab
Room NE43-819

(Signature of author)

Date of submission: March 3, 1995

Expected Date of Completion: May 11, 1995

Laboratory where thesis will be done: MIT Artificial Intelligence Laboratory

Brief Statement of the Problem:

The cerebellum has long been known to be associated with the coordination of the human motor system. Contemporary research indicates that this is one product of the cerebellum's true function: the generation of dynamic models of systems both within and without the body. This proposal describes the instantiation of one such model in the humanoid robot Cog, developed at the MIT AI Lab. The model takes the form of an adaptive mapping of head movements into anticipated visual motion. This model is part of a visual subsystem which allows Cog to detect motion in the environment without being confused by movement of its own head. The author hopes that this work will be the first step in the development of a fully-functional artificial cerebellum for Cog.

Supervision Agreement:

The program outlined in this proposal is adequate for a Master's thesis. The supplies and facilities required are available, and I am willing to supervise the research and evaluate the thesis report.

Rodney A. Brooks,
Professor of Computer Science and Engineering

1 Introduction

Cog (Figure 1) is a humanoid robot being developed at the MIT Artificial Intelligence Laboratory. One of the overall goals of the Cog Project is to learn about human cognition by trying to emulate it, guided by the hypothesis that many of the foundations of human cognition are molded and set by the interaction and available methods of interaction of the human body with the rest of the world (Brooks & Stein 1993).

Figure 1: Cog, a humanoid robot in development. Cog won't have legs, but will have two arms and hands in the near future.

In humans, the cerebellum has a well-known, albeit not well-understood, role in the control of movement. Traditionally, that role has been associated with coordination and calibration of motor output. Cerebellar function is typically described in terms of the motor deficiencies associated with cerebellar *dys*function: overshooting and oscillation in voluntary movements, poor muscle tone, decomposition of complex movements into component parts, ataxic gait, and slowed, slurred speech (Dichgans & Diener 1985). These abnormalities don't occur when the cerebellum is correctly assisting the motor cortex.

Mounting evidence suggests that the cerebellum is not just a helpful add-on to the motor cortex, but that it plays a more generalized role of modeling the body's sensory and motor systems. In many species of animals which use electro-location or echo-location, the cerebellum is oversized in comparison to the sophistication of the animal's motor skills. Paulin (1993) thinks that these creatures use the proposed dynamic modeling facilities of the cerebellum primarily to process the complex sensory inputs which they experience. In humans, dynamic models of the body and environment are put to better use by aiding our well-developed motor control.

For my master's thesis, I intend to implement some of this functionality on Cog. I am working on a system which learns a mapping between movement of the head and the resulting apparent visual motion. The system is manifested as a neural network which translates small displacements of the head into displacements of small patches of the visual field. Essentially, this is a model of head kinematics parameterized in terms of the visual system.

This is intended to eventually be only one of many such systems, each of which translates between the task-specific natural parameter spaces of different sensory and motor systems. Such an architecture avoids world models and global coordinate systems, and is a natural candidate for implementation with a distributed computation scheme, such as that built into Cog. In addition, having many models between different systems ensures redundancy and reduces susceptibility to sensor error and noise.

The bulk of this proposal describes the work already completed: creation of a preliminary system which maps two degree-of-freedom (d.o.f.) head motion into two d.o.f. visual motion. I intend to extend this to three d.o.f. head motion, with some incorporation of sensory dynamics. At the end of this proposal is a description of the pending work, and a schedule for its completion.

2 The Robot Cog

A quick description of Cog is in order. Cog is the focus of a project led by Rodney Brooks and Lynn Stein at the MIT AI Lab. Mechanically, Cog is an approximation of a human from the waist up. Although bolted to a sturdy machine base, Cog has two degrees of freedom (d.o.f.) in its hips, and another d.o.f. for twisting its shoulders. Cog's neck has a full three degrees of freedom. Cog has two independent 2 d.o.f. eyes. Each is composed of two black and white cameras: one with a wide-angle lens to handle full-field peripheral vision and one with a narrow-angle lens for foveal vision. Cog will soon have two 6 d.o.f. arms (Williamson & Pratt 1995) and two many-d.o.f. hands (Matsuoka 1995). One of each has been built, but have yet to be installed. All of Cog's actuators are driven by electric motors, each with its own microprocessor-based servo-controller.

Cog's brain is an open architecture multiprocessor system (Kapogiannis 1994). The current setup has room for 256 processors, each of which can communicate with up to six other processors via dual-ported RAM (DPRAM), and with up to 8 motor or sensor controllers via high-speed serial lines. The processors in use are Motorola 68332's — a 16 MHz, 16-bit variant of the 68020 — which run L, a multitasking dialect of Lisp (Brooks 1994). Each processor runs its own L-image; there is no shared memory in the system except for that used to link individual pairs of processors. A Macintosh PC is used as a front-end to provide tty access and file service for the processors.

At this moment, Cog's senses are limited to the eyes and joints. Images from the cameras are fed to the processors in real-time via DPRAM from custom built frame-grabbers. The motor controllers provide Cog with sensing of joint position and torque. More senses are on the way, though: ears for Cog are under development (Irie 1995), as is tactile sensing and a vestibular system.

3 The Present System

The root goal for this particular project — the justification for developing the head-to-vision modelling system – is a single, basic task: reliable motion detection. The goal is to create a visual subsystem which does nothing more than to report where motion is occurring in the visual field and what that motion is. However, the system has to be smart enough to distinguish between real motion in the world and the apparent motion caused by a moving eye. Doing that requires that the visual subsystem receive knowledge of what the head's motor systems are doing. In the biological world, this is known as "efference copy" (Gallistel 1980, p.166f).

In the simplest case, efference copy can refer to an inhibition signal: when the head or eyes are moving, motion detection just shuts down, knowing only that it's a bad time to look. In a more complicated and more useful case, the efference copy really is a copy of the motor control signals, from which the visual system determines where and how the head will move, and adjusts its motion detecting accordingly. This is a much more useful approach, since Cog will not go blind to motion when it twists to look over its shoulder. It is also a much more complicated approach, requiring the visual system to use some dynamic model of the head's sensorimotor systems.

The system I have currently implemented is a primitive form of the latter approach. I only use one of Cog's cameras, which is restrained so that it cannot move with respect to the head. I also restrict movement on Cog to only the pan and tilt motions of the neck. Furthermore, I assume that position reports from the neck controllers are accurate and synchronized with the receipt of video frames. Thus, the system is only concerned with a total of four degrees of freedom (Figure 2), and the timing and dynamics of the motor control are not taken into account.

With all this in mind, the task breaks down into three parts: moving the head, detecting motion, and learning how to correct for head motion.

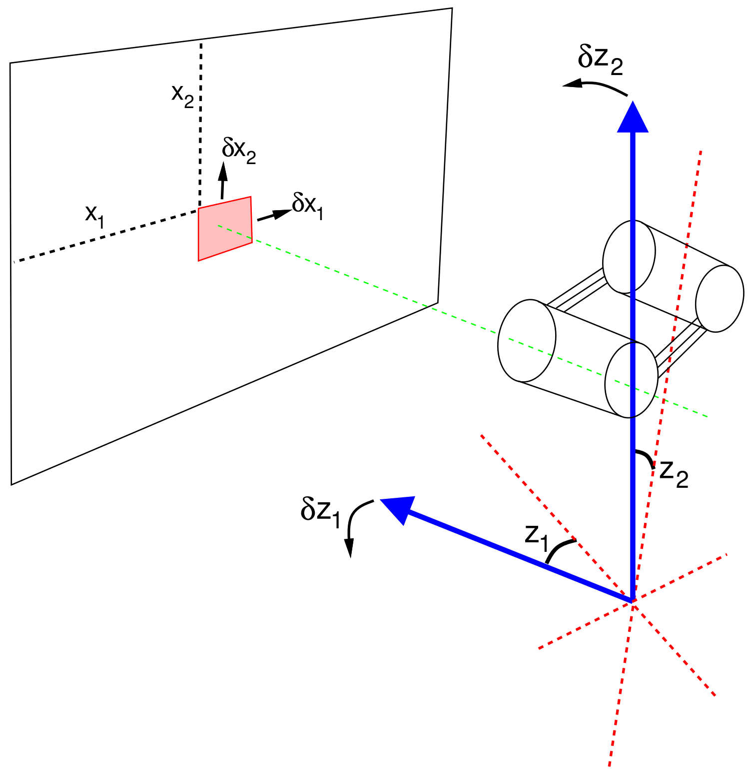

Figure 2: Parameterization of the motion detection task. $\vec{x}$ is a coordinate in the visual field; $\vec{z}$ represents the position of the head.

3.1 Head Movement

Eventually, when more systems are running on Cog, some combination of processes will motivate Cog to move its head around. In the meantime, for the purposes of experimenting with motion detection, Cog's neck motors are motivated by a random number generator. A random coordinate of pan and tilt positions is picked from a range of 80% of the full range of motion, Cog's head smoothly tracks to that target, and then a new target is picked and the head tracks again. This ensures both continuous motion of the head, so that there is always something to learn and correct for, and it ensures that the head visits the entirety of the 2 d.o.f. position space.

3.2 Motion Detection

The motion detection system is composed of a number of independent detector processes, each of which operate on a small region of the full visual field. These regions are distributed across the visual field. Cog's full visual field is a 128x128 array of pixels; each motion detector operates on a 16x16 pixel subarray. Each detector reports a motion vector for its particular subarray, which combines with all the other results to create a motion vector field over the whole image. The regions of interest are currently fixed in place, but they do not need to be. One alternative is to allow detectors to lock onto movement they detect, and then follow and track it.

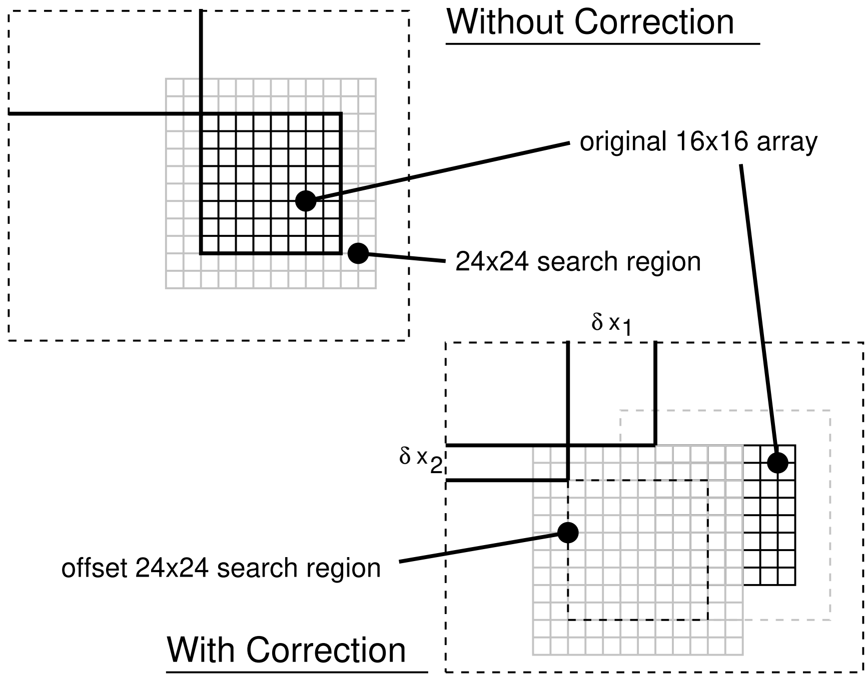

Figure 3: Illustration of the arrangement of pixel arrays used in motion detection.

The detection scheme is a maximal-correlation search technique, illustrated in Figure 3. During one video frame, the 16x16 array of interest for a detector is grabbed and stored. That array is then shifted around within a 24x24 pixel array from the succeeding video frame to find the offset which yields the best match. The 24x24 array is by default centered over the original 16x16 array. The offset coordinates of the best match then give the motion of the 16x16 region between frames. Since the search region is only 8 pixels wider and taller than the original array, each component of the motion vector output is an integral value of ± 4 pixels.

To add a correction input to this scheme, the 24x24 array can be grabbed from an offset location. Thus, instead of searching for the original subimage where it used to be, the detector searches where it expects it to be, knowing that the head has moved.

The correlation between 16x16 arrays used to find the best match is computed by using a logical XOR between pixels. The 8-bit greyscale pixels in the 16x16 and 24x24 arrays are actually first converted to 1-bit values by thresholding with the average of the original 16x16 array. This allows an entire row to be packed into one machine word and XOR-ed by a single machine instruction, which immensely speeds up processing. The method is crude and simple, but it works well.

3.3 Learning to Correct for Head Movement

To correct for head movement, the motion detection system needs to learn a transform from head motion to visual motion. For this learning to be biologically plausible and computationally feasible, the training technique needs to be an on-line algorithm. It should also operate continuously, so that the system adapts to changes in the robot over its lifetime. A learning rate which takes hours or days is acceptable, though, since Cog is supposed to adjust and adapt to the world with human-like responsiveness.

In this implementation, the transform is in the form of a network which takes neck motion $\delta\vec{z}$ as an input and returns the corresponding apparent visual motion $\delta\vec{x}$ as the output. The motion detectors use this to generate a corrected motion vector

$$\Delta\vec{x}_c = \Delta\vec{x}_d - \delta\vec{x}, \tag{1}$$

where $\Delta\vec{x}_d$ is the raw detected visual motion.

The system trains the network by assuming that, for the most part, the real world is still. If there is a large number of motion detectors distributed around the visual field, but only a small number of moving objects, then at any one time most of the detectors should register zero motion. Any motion that is detected is due to head movement and error in the correction network. Thus, the error signal for training the network is simply the final motion output of the detectors.

Now, if something does move in the environment and this is picked up by a detector, the output will cause mistraining of the network. However, if the learning rate is small, it will have little effect. Motion of objects in the world is not, for the most part, correlated with head motion, so the mistraining will manifest itself as a low-level of noise in the network.

3.4 Transforming Head Motion into Visual Motion

Ostensibly, there exists some function $\vec{f}$ of neck kinematics and camera optics, such that given a visual coordinate $\vec{x}_1$ and neck position $\vec{z}_1$ at some time, and a neck position $\vec{z}_2$ at some later time, the output is $\vec{x}_2 = \vec{f}(\vec{x}_1, \vec{z}_1, \vec{z}_2)$ — where the original visual coordinate has moved. Applying $\vec{f}$ to an image of the world in Cog's eye will yield the image it sees after moving its head. This function is hopefully differentiable, but probably has a very unbecoming analytic form, so it makes sense to let some network go off and learn the function for itself.

The locality of the motion detection scheme allows us to simplify things a little bit by taking a first order approximation of $\vec{f}$ in the neighborhood of some initial point $\{\vec{x}_0, \vec{z}_0\}$:

$$\vec{x}_1 = \vec{f}(\vec{x}_0, \vec{z}_0, \vec{z}_0) + (\vec{z}_1 - \vec{z}_0) \left.\frac{\partial \vec{f}}{\partial \vec{z}_2}\right|_{(\vec{x}_0, \vec{z}_0, \vec{z}_0)} \tag{2}$$

Since $\vec{f}(\vec{x}_0, \vec{z}_0, \vec{z}_0) = \vec{x}_0$ (the case of no head movement),

$$\begin{aligned} \vec{x}_1 - \vec{x}_0 &= (\vec{z}_1 - \vec{z}_0) \left.\frac{\partial \vec{f}}{\partial \vec{z}_2}\right|_{(\vec{x}_0, \vec{z}_0, \vec{z}_0)} \\ \delta\vec{x} &= \frac{\partial \vec{f}}{\partial \vec{z}_2}\delta\vec{z} \\ \delta\vec{x} &= \mathsf{A}\delta\vec{z} \end{aligned} \tag{3}$$

This has reduced the problem to learning the matrix A, a linear transformation from head position space to visual position space which is perfectly suited for use with the regional motion detectors. Given a visual patch at $\vec{x}$ and an initial neck position $\vec{z}$, the matrix $\mathsf{A}(\vec{x}, \vec{z})$ will transform the small movement of the neck between video frames into the resulting anticipated visual motion of the patch. A is not necessarily simple, though. In the current problem, with two degrees of freedom each in the neck and in the visual field, A is a function which maps $\mathcal{R}^4 \rightarrow \mathcal{R}^4$.

3.5 Network Implementation

A hybrid neural network is used to learn an interpolated, piecewise linear approximation of A (Figure 4). This network, similar to one used by Stokbro et al. (1990), is a type of *mixture of experts model*, composed of a set of output nodes $\{u_{pqi}\}$ and a set of gating nodes $\{g_i\}$.

Each output node u_{pqi} is a linear function of $\vec{z}$ and $\vec{x}$, corresponding to an element A_{pq} of the matrix A:

$$u_{pqi} = \sum_j a_{pqij} x_j + \sum_j b_{pqij} z_j + c_{pqi} \tag{4}$$

Each gating node g_i — one for every set of output nodes — is a radial gaussian unit characterized by a mean $\vec{\mu}_i$. The receptive field of a node g_i determines a region of expertise for its associated output nodes u_{pqi}. That is, for some input $\{\vec{x}, \vec{z}\}$, the weight given to the linear approximation u_{pqi} in the final output is:

$$h_i = \frac{g_i}{\sum_j g_j}, \quad g_j = e^{-(\{\vec{x}, \vec{z}\} - \vec{\mu_j})^2} \tag{5}$$

The final output of the network is the weighted sum of all the linear approximations contained therein:

$$\mathsf{A}_{pq} = \sum_i h_i u_{pqi} \tag{6}$$

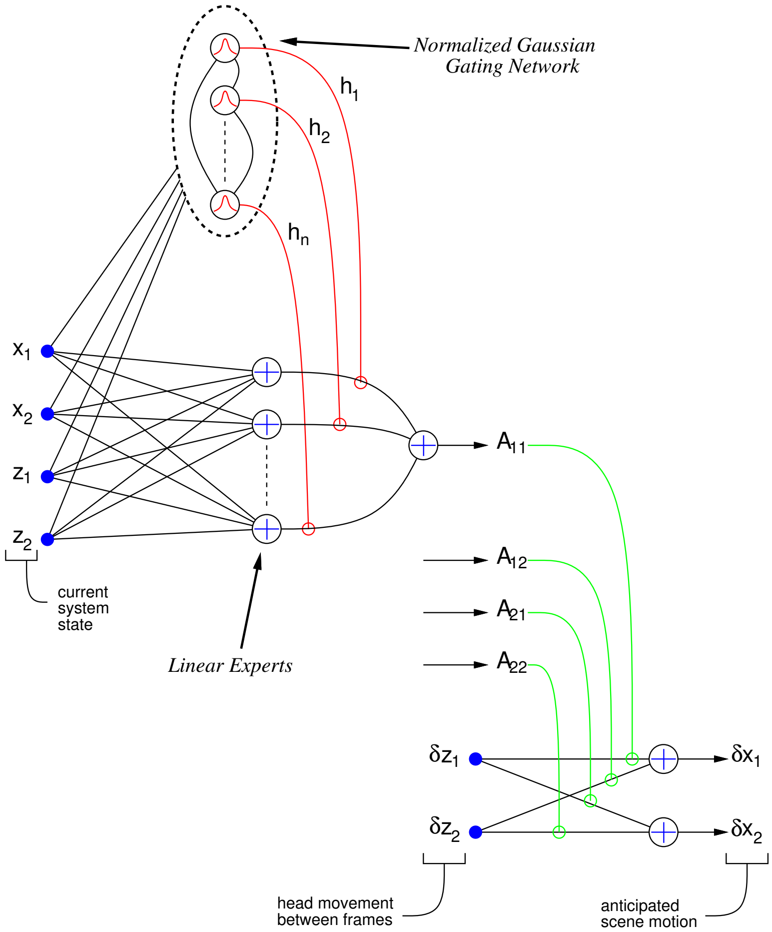

Figure 4: A schematic diagram of the network used to approximate the transform A.

Each output node u_{pqi} is thus an expert for inputs in a neighborhood about the mean $\vec{\mu}_i$; the output of the network is the sum of the experts' opinions, weighted by their expertise.

This network is trained in two phases. First, competitive learning — an unsupervised learning technique — is used on the gating network. A new input $\{\vec{x}, \vec{z}\}$ is presented to the gating nodes and the "fittest" node — the one with maximal h_i — is jostled closer to the input point via:

$$\Delta\vec{\mu}_i = \rho(\{\vec{x}, \vec{z}\} - \vec{\mu}_i), \tag{7}$$

where ρ is an appropriately chosen (*i.e.* small) learning rate constant. The overall effect of this jostling is to distribute the gating nodes' receptive fields over the input space. Although the input vectors are represented as members of some real space $\mathcal{R}^n$ (in this case, $\mathcal{R}^4$), they may lie in a lesser dimensional manifold or be restricted to some bounded subset. With competitive learning, the gating nodes find this subspace, concentrating the efforts of the linear expert nodes where they are actually needed.

In the second phase of training, the output nodes $\{u_{pqi}\}$ are trained using gradient descent with a least-mean-squares (LMS) learning rule, a standard supervised learning technique (Hertz, Krogh & Palmer 1991). As described earlier, the error signal is simply $\Delta\vec{x}_c$, the compensated output from a motion detector, since the system is being trained to report zero motion. After propagating $\Delta\vec{x}_c$ back through a sum-of-square-error cost function, weight updates take the form:

$$\Delta a_{pqij} = \eta h_i(\delta z_q)(\Delta x_{cp})x_j \tag{8}$$

$$\Delta b_{pqij} = \eta h_i(\delta z_q)(\Delta x_{cp})z_j \tag{9}$$

$$\Delta c_{pqi} = \eta h_i(\delta z_q)(\Delta x_{cp}) \tag{10}$$

where η is another learning rate constant.

Both learning phases are applied to the network with every new data point, collected in real-time from the motion detectors. Since each datum is a fresh, new piece of information from the world-at-large, there is no danger of overtraining the network, as there might be if the network were being trained from some fixed sample data set.

4 Preliminary Results

The system described above has been implemented on Cog in almost its full glory. Instead of running many motion detection processes distributed around the visual field, only one is currently in service; the single 68332 processor used for experiments does not have enough muscle to run multiple detectors at an acceptable speed. However, the single motion detector does move around. Every two seconds, its region of interest is moved to a random location in the visual field, to ensure that the entire $\vec{x}$ space iss traversed.

The approximation network uses 17 linear experts. This number is an *ad hoc* choice, based on the number of vertices of a 4-dimensional hypercube. This seems to be the minimum number of linear pieces necessary to approximate a convex smooth function over a 4-dimensional input space. The number is kept small to speed up processing; even 17 experts results in 408 parameters to be trained.

When running, the system processes $\sim$4 frames per second, a seventh of the base video input rate, and about two orders of magnitude too slow to be useful to Cog for interacting with the world. With the future introduction of floating point coprocessors, and some clever machine-level coding, it should be possible to bring the system up to speed.

In a typical run, Cog is left mostly alone for 6-8 hours. Learning rates are set high enough to yield results in that amount of time, and to avoid underflowing the six digits of precision currently implemented in L. Results from such a run are shown in Figure 5, a graph of root-mean-square error vs. run-time. The error is averaged over every 1000 video frames processed, and is measured over the entire $\{\vec{x}, \vec{z}\}$ input space.

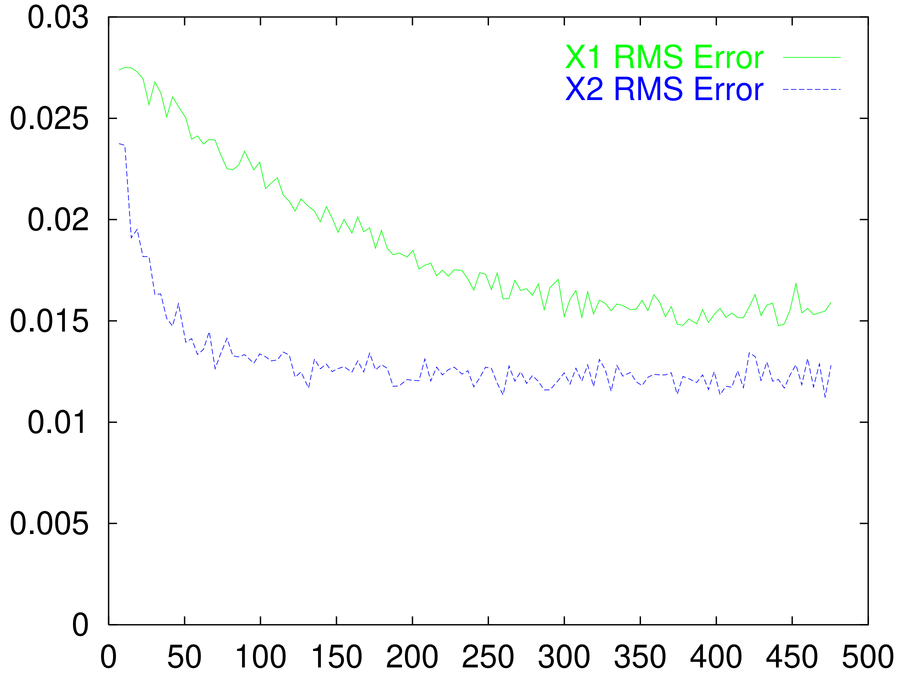

Figure 5: RMS error vs. run time of the correction network for a typical run. Time is in minutes; error is in fractions of the total visual field ($0.008 \sim 1$ pixel).

Unfortunately, the error doesn't drop dramatically to zero; it only drops to half of the error in a non-corrected system. The current network isn't all that it was hoped to be.

The residual error probably comes from three main sources. First, 17 linear experts only marginally cover the input space of the problem. Second, the dynamic range and precision of the motion detection is low, since the output is an integer quantity in the range $[-4, +4]$. This makes the system very susceptible to quantization noise. Third, the detection system does not take into account any of the dynamics of the sensory systems. It assumes that information on head position is timely and accurate with respect to video framing, which is definitely not the case. Many time delays lurk along the pathways between the neck motor controllers and the motion detection processes.

Nonetheless, the system is on the right track.

5 Pending Work

Table 5 gives a schedule of remaining work. As it stands, this system only accounts for two degrees of freedom in the neck, and none in the eyes. I plan to extend the system to deal with the extra degree of freedom in the neck; the eyes will remain fixed for the time being. Eventually (i.e. after the master's thesis), I would also like to use a similar system to model eye kinematics in terms of visual motion. Hooking the two up would allow Cog's eyes to

Task	Date
Optimize motion detection	March 10, 1995
Optimize network learning	March 17, 1995
Add third d.o.f. and dynamics	March 24, 1995
Develop tracking task	April 7, 1995
Finish testing, begin writing	April 21, 1995
Submit thesis	May 11, 1995

Table 1: Schedule of work from present until completion of thesis.

compensate for retinal slip. With a vestibular apparatus is in place, the same system could be used to implement a vestibulo-ocular reflex (VOR).

Another necessary refinement is a rework of the network to incorporate the dynamics of the head and of the sensory systems on Cog. One possibility is fashioning it into a Smith Predictor — a control system which contains a model of plant dynamics as well as a model of time delays in the control loop. The Smith Predictor has been regarded as a candidate for the basic functional block in the cerebellum (Miall, Weir et al. 1993).

Once the head-to-vision model in place, I would like to use it for some other task in addition to motion detection. The model could be used in the feedback loop of a tracking system which directs the head to keep the eyes fixated on a moving object.

The vision here is to find a function approximator with enough versatility that it can be used to model any one sensorimotor system in the parameter space of another. A phalanx of these models could be put to use by control systems and processes to allow these systems to effectively cooperate with each other. This approach avoids the need for, and the fallacy of, trying to build a single, canonical model of the world. Instead, Cog's brain will see the world through many small, task-specific models, linked together by the various sensorimotor systems and their interactions with the world.

6 Acknowledgments

I would like to thank Rod Brooks for making the Cog Project possible, Cynthia Ferrell for helping immeasurably to make the project a reality, and Matthew Williamson for the rich conversations which inspired the work in this proposal and the upcoming thesis.

References

Brooks, R. A. (1994), *L*, IS Robotics. reference manual.

Brooks, R. A. & Stein, L. A. (1993), Building brains for bodies, AI Lab Memo 1439, MIT Artificial Intelligence Lab.

Dichgans, J. & Diener, H. C. (1985), Clinical evidence for functional compartmentalization of the cerebellum, *in* J. R. Bloedel et al., eds, 'Cerebellar Function', Springer–Verlag, New York, pp. 127–147.

Gallistel, C. R. (1980), *The Organization of Action*, L. Erlbaum Associates, Hillsdale, N.J.

Hertz, J., Krogh, A. & Palmer, R. G. (1991), *Introduction to the Theory of Neural Computation*, Addison–Wesley, Redwood City, CA.

Irie, R. (1995), "Robust sound localization: an application of an auditory system for a humanoid robot", submitted to the International Joint Conference on Artificial Intelligence, Montreal, Canada.

Kapogiannis, E. (1994), Design of a large scale MIMD computer, Master's thesis, Massachusetts Institute of Technology.

Matsuoka, Y. (1995), Masters thesis proposal in progress, Master's thesis, Massachusetts Institute of Technology.

Miall, R. C., Weir, D. J. et al. (1993), 'Is the cerebellum a Smith Predictor?', *Journal of Motor Behavior* **25**(3), 203–216.

Paulin, M. G. (1993), 'The role of the cerebellum in motor control and perception', *Brain Behav Evol* **41**, 39–50.

Stokbro, K. et al. (1990), 'Exploiting neurons with localized receptive fields to learn chaos', *Complex Systems* **4**, 603–622.

Williamson, M. & Pratt, G. (1995), "Design of a Naturally Compliant Arm for a Humanoid Robot", submitted to the International Joint Conference on Artificial Intelligence, Montreal, Canada.