Flight Control Using an Artificial Neural Network

Gordon Wyeth and Gregg Buskey
Computer Science and Electrical Engineering
University of Queensland
St Lucia, Queensland, 4072
Australia

Jonathan Roberts
CSIRO Manufacturing Science & Technology
P.O. Box 883, Kenmore, Australia. 4069
Australia

Abstract

This paper details the developments to date of an unmanned air vehicle (UAV) based on a standard size 60 model helicopter. The design goal is to have the helicopter achieve stable hover with the aid of an INS and monocular vision. The focus of this paper is on the development of an artificial neural network that makes use of this data to generate hover command, which are used to directly manipulate the flight servos. Current results show that a mapping can be learnt for the training data, but generalisation to unseen data has been poor with current training sets.

1 Introduction

This paper describes the preliminary aspects of the design of an adaptive autonomous helicopter control system, for which the overall design goal is to achieve stable hover under most conditions. The system is equipped with on board monocular camera, inertial navigation system, global positioning system, ultrasonic proximity sensor (used for altitude less than around 1m), and controller computer. At present, the system's stimulus is the INS alone; essentially, the UAV is flying blind.

Previous works by Shakernia et al. [1] and R. Miller & O. Amidi [2] in the area of autonomous air vehicles have taken the approach of hard coding the flight algorithms into an onboard computer that is responsible for integrating flight strategy, making short term adjustments to disturbances, and maintaining actuator control. The approach described herein differs, in that the computer will learn to fly through observation and experience. Since the flight algorithms are not coded directly, the on board computer is required to learn how to mimic the actions of the trainer. Furthermore, the computer needs to be able to respond to situations previously unseen in the training data. Given the characteristic non-linearity of helicopter flight dynamics, this journey into unfamiliar control space can result from any one of a number of external factors. Attempts to use traditional control methods for the construction of inverse control laws for helicopter plants, will inevitably encounter problems when faced with the time-variant, non-linear nature of these systems. Scaled versions suffer even further problems with regard to stability, given their rotor inertia to frame inertia ratio is of the order of six times greater than that of their full size equivalents. The necessity for adaptation, coupled with the need to be able to deal with these non-ideal dynamics, makes this problem particularly suited to neural networks and/or fuzzy logic controllers.

Autonomous helicopters have recently been employed in mapping arctic craters to be used in NASA extra-terrestrial training simulations [1]. This system was developed at Carnegie Mellon University, exhibiting full take-off, landing, and flight path tracking. Further work has been carried out at Berkley University, in which a UAV with a new method of translational and rotational movement estimation using a 4 image point estimation alignment algorithm is to soon be implemented [2].

1.1 Approach Outline

The works described in [1] & [2] have involved using fuzzy controllers, or genetic algorithms (or combinations thereof) to implement an inverse control law for primitive movements, which is then used by a tactical planning computer (among other flight layers) to maintain flight control. This work investigates the process of having an unmanned air vehicle, learn not only how to map its internal aeronautic structure to its desired movement, but also to gain an internal representation through observation, of the task required. Essentially, it uses direct mapping of sensor inputs to actuator control via an artificial neural network, such that the UAV is completely reactive. A feed forward temporal network using the back propagation training regime is employed to learn the INS to actuator control relationship (see Figure 1).

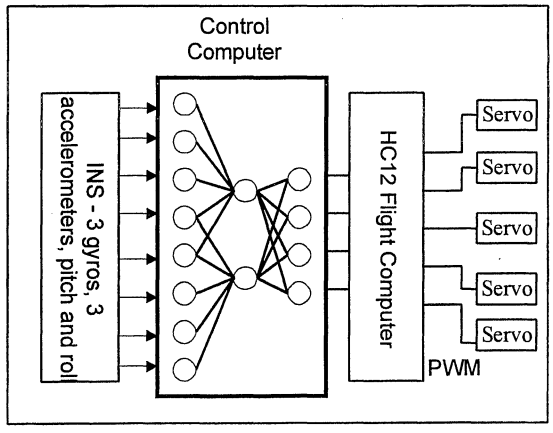

Figure 1 - Autonomous Flight System

Temporal networks, in which some information about the past also serves as input stimulus (sometimes termed partially recurrent networks) have been successfully used in applications in which some form of non-markov process is incorporated; speech reproduction [3] is one such example. While it is true that the network structure and learning algorithm outlined above is not considered to be found in biological organisms, it is nonetheless a useful tool for taking the design of a UAV one step closer to a biological parallel.

2 Hardware

The system uses a JR Ergo standard size 60 model helicopter powered by a petrol driven two stroke engine, with a main rotor span of 1535mm and a gross weight of 4.8kg. The onboard system is a combination of conventional and high-tech flight equipment. Conventionally the helicopter is equipped with a traditional radio transmitter and receiver and traditional flight control servos. Unlike most model helicopters, the Ergo is also equipped with a flight computer based on a custom dual HC12 board mounted in the nose of the aircraft, and a control computer (PC104 Plus based AMD K6-2 300MHz) slung beneath the fuselage in an effort to offset the helicopter center of inertia by as little as possible.

The flight computer reads the radio receiver signals which it can send to the control computer for logging. It regenerates the control signals from the original, or as instructed by the control computer, to manipulate the servos in autonomous flight. Individual servo channels can be switched during flight to facilitate forms of partial autonomous operation. There are five servos controlling aileron, elevator, rudder, collective and throttle. Note collective and throttle share the same linkage and hence really need only one control signal between them.

The control computer is also equipped with an Crossbow INS (DMU-VG) provides 6 DOF information incorporating 3 translational and 3 rotational components. The device uses micro-electro-mechanical-systems (MEMS) technology allowing for silicon based tilt sensors and solid state gyros to provide a smaller (475 grams) alternative to their traditionally cumbersome mechanical counterparts. The INS unit also utilises DSP to extract absolute pitch and roll information from the 6 rate sensors, accurate to within ±1°. This paper investigates the possibility of training a system to correlate the INS with the control signals generated by an experienced pilot as shown in Figure 2.

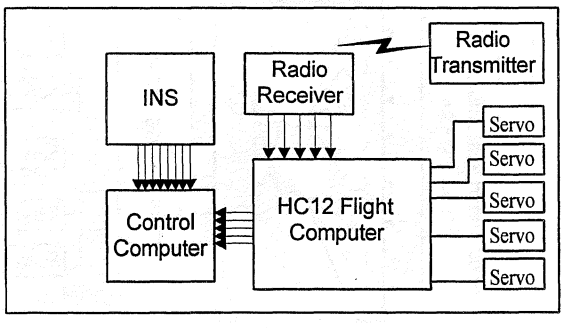

Figure 2 - System Training Setup

The control computer is interfaced to a number of other devices not used in this paper. A single monocular CCD camera is available through a Sensoray frame grabber capable of 50 frames per second. The control computer is also equipped with an ethernet card allowing offline communication with the base computers, and capable of up to 50 frames per second. The helicopter is also equipped with GPS capabilities, although the intention is to rely on this as little as possible, given the terrain dependant degree of reliability experienced in [1].

One hardware module not yet implemented is a closed loop controller allowing precise manipulation of the engine RPM via the throttle servo. This is deemed necessary, based on the throttle to lift variations from flight to flight. This variation is a consequent disadvantage of using a two stroke engine; the advantage being higher power to weight ratio.

3 System Architecture

During autonomous flight, the neural network uses the information supplied by the INS about the helicopter's attitude, to generate an appropriate set of servo responses. The INS information is periodically sampled and sent via the RS-232 link to the control computer. This data is applied to the inputs of the neural network, which is resident on the control computer. The network outputs are then sent to the flight computer, which in turn sends appropriate pulse width modulation signals to the five helicopter servos (aileron, elevator, rudder, collective). The network is trained to generate the correct response to a given helicopter attitude using the back-propagation supervised learning algorithm. During this training phase, both the INS information and the servo inputs generated by the pilot trainer are sampled and logged back to the control computer. These input/output sets are then used to train the neural net on examples of viable responses to a variety of hover situations.

3.1 Data Preprocessing

Before the data is presented to the network, however, a certain degree of pre-processing is required. Firstly, it should be noted that the sampling rates of the INS and servo information differ by around 20%, with sampling periods of 30ms and 22ms respectively. Naturally the two need to be aligned to a common sampling frequency before meaningful training sets can be constructed. Using DSP interpolation to upsample both signals to a common 1kHz sampling frequency is not an option when using accelerometers. This is because acceleration is a second derivative term, which is often noisy and hence plagued by high frequency components. Therefore, there is no guarantee that the data will satisfy the Nyquist criterion for successful signal upsampling, which states that the highest spectrum frequency must be less than half the original sampling frequency. Therefore, it was decided to treat the INS sampling rate as the system sampling rate, and simply match each INS sample to the nearest following servo sample. The INS was chosen to be common simply because it ultimately determines the state update rate of the system during autonomous flight.

The next stage of the pre-processing is incorporated to make the task of learning, and making generalizations about

the input to output mapping by the net more manageable. Each of the eight input and four output scalars are expanded into 10 point vectors calculated to each take the form of a non-normalized Gaussian distribution. The result is the formation of a nearest neighbour association between like valued input and output sets. More specifically, each vector bin is assigned a scalar quantization. The value βi of each bin for a given scalar input α is decided by the numerical distance of its quantization value χi rom the scalar as follows:

$$\beta_i = e^{\frac{(x_i-\alpha)^2}{\sigma^2}}$$

In essence, each input and output looks like a sliding Gauss function. This concept is illustrated in Figure 3.

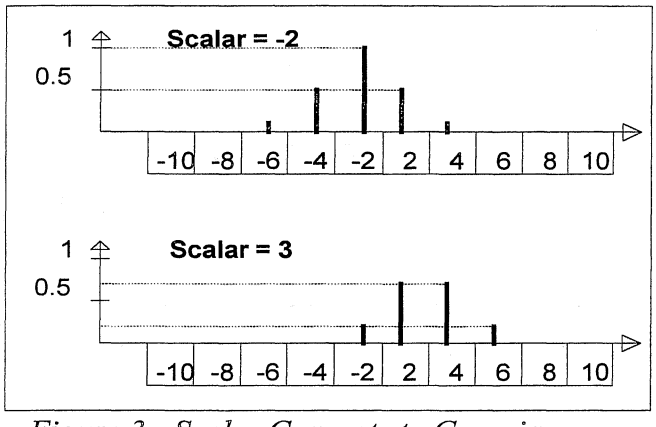

Figure 3 - Scalar Converts to Gaussian vector

The decision regarding the degree of nearest neighbour association (Gaussian variance), while not arbitrary, is certainly not based on any solid theory. The design choice of σ equaling 0.84 sets up vectors with peak bin values of one having left and right neighbouring bins equaling 0.5; bins two degrees removed are for the most part zero. This of course, occurs only if the presented scalar matches a vector quantization bin exactly, however the degree of association for other scalar values is very similar.

Perhaps more important is the choice of the actual bin quantization values. Given that outliers will always exist in the training data, using a linear scale that includes the entire range for a given input or output is unsatisfactory for two reasons. The first is that it facilitates no room for more extreme outliers during autonomous flight. The second is that the outliers will stretch the vector such that most of the flight time is contained within the middle one or two vector bin quantizations. The solution to both inadequacies comes in the form of using the statistical distribution for each input and output to calculate their associated vector structure. More specifically, the middle two bins of the vector associated with the *x*-gyro for example, should have quantizations that envelope the region about the *x*-gyro average that covers 20% of the flight time. The second two from the middle should envelope a total 40% flight time, with the outer two incorporating virtually the full 100%.

Note that with this method, scalars and quantization bin values need to be normalized to their standardised random variable equivalents [5]. This structure results in increased resolution in the region of most stable hover, while still allowing for statistically outlying regions of control space to be represented. The resolution increase improves the network's ability to interpret the intricate movements associated with stable hover.

3.2 Network Architecture

The network itself, is a temporal network using a time window of ten samples (300ms), resulting in an input size of 100 nodes for each of the eight scalar inputs (800). The outputs are treated only in their static form (40). This temporal nature allows the helicopter to react not only to direct stimulus, but also to the context within which it is presented. The choice of the actual internal architecture of the network, including the type of neuron units (excitory / inhibitory or pure excitory) selected, is like any feed-forward neural network trained using back-propagation; very much application dependant. Typically, a large number of hidden units allows for better values of convergence between the net outputs and the target outputs compared during training. However, a smaller number of hidden units results in better generalisation when exposed to unseen situations during testing, so like most engineering problems, a trade off exists between the two terms of merit [6].

One method of minimizing the number of synaptic connections is to try and build some a priori knowledge of the system dynamics into the model [6]. The approach taken for this system is to use a form of partially connected network, the nature of which is described in sections 4 and 5. Other parameters such as learning rate are also responsible for a trade off between speed of convergence, and location of more optimal solutions; these are also discussed in the following section.

4 Training Algorithm

The Back Propagation training algorithm is a method of supervised neural network learning. During training, the network is presented with a large number of input patterns (training set). The desired corresponding outputs are then compared to the actual network output nodes. The error between the desired and actual response is used to update the weights of the network inter-connections. This update is performed after each pattern presentation. One run through the entire pattern set is termed an epoch. The training process continues for multiple epochs, until a satisfactorily small error is produced. The test phase uses a different set if input patterns (test set). The corresponding outputs are again compared to a set of desired outputs. This error is used to evaluate the networks ability to generalize. Usually, the training set and/or architecture needs to be assessed at this point.

There are several parameters that can be adjusted to improve training convergence. The two most relevant are momentum α and learning rate η. If the error is considered to lie over a multi-dimension energy surface, η is considered to designate the step sizes across this surface to reach a

global minimum. Momentum, on the other hand, tries to push the process through any local minima. A learning rate too large results in the global minima being stepped across; too small and the convergence time is prohibitively large. With the momentum, too large a value sees the process oscillate, while too small can result in local minima entrapment. The actual shape (stretch) of the network neuron transfer functions have also been found to affect convergence [4]. As previously mentioned, the number of hidden units, and the level of connectivity, can have a large effect on training speed, however, a trade off between training and testing success exists.

4.1 Performance Evaluation

This success is most often gauged on the mean square error between the ANN's response to the training or testing set inputs, and the corresponding targets. While this is certainly relevant for classification systems, the measure of merit for some control system ANN applications can be quite different. Given that the system described in this paper must learn to develop a generic trajectory approximation based on input stimulus, it makes more sense to evaluate the closeness between the target sequence and the network output sequence over time rather than consider each output error separately. This is not to say that MSE should be ignored; only that it needs to be put in context as a gauge of the network's ability to produce reasonable sequences. One suggested method for evaluating such a network's success is proposed in [7], in which merit is based on the correlation between the actual and target time sequences. At this point, visual evaluation of plots between the two have been used as an exercise of benching certain MSE ranges, which is essentially the same process; formal mathematical correlation may be incorporated in the final testing regime if deemed necessary.

5 Experiments

Exhaustive experiments were initially carried out using excitory/inhibitory (bipolar) neurons. While the results are not included, it was found that no significant improvement in training success resulted from varying any or all of the parameters previously discussed. Changing to pure exhitory units (unipolar) resulted in immediate improvements. The experiments detailed herein use only unipolar neurons.

The experiments to date have focussed on deduction of best connectivity, activation unit type, and hidden unit number. These investigations were carried out using several training sets of varying sizes. Little has been investigated into the effect of varying initial weight ranges, as [4] found for largely connected networks, such variations have little effect on the rate of training convergence. The experiments of interest carried out thus far are as follows:

- First a network with a large number of hidden units (40) was trained with learning rate of 0.02, momentum of 0.95 and architecture fully connected. This was an attempt to converge the network to a very low MSE. The intention was to then plot the network output on seen data to put MSE values into some context.
- Architecture was then changed to 10, 6 and 2 hidden units, so that a possibly worse MSE could be gauged visually, in order to test for possible ranges of acceptable MSE.
- A partially connected network, in which x-gyro, y-accelerometer and roll were connected to one set of hidden units, y-gyro, x-accelerometer and pitch to another, and z-gyro and z-accelermoter connected. This was to look for any synaptic redundancy within the network
- Test data simulations were run at five epoch training intervals, in an attempt to understand where, if at any point, the influence of over training began to manifest itself in the generalization performance.
- Increasing the stretch factor for the hidden units in order to look for evidence of premature saturation.

5.1 Results

The relationship between number of hidden units and MSE on training data is shown in Table 1 below. It also shows a comparison between partially and fully connected units. These results were compiled using a training set comprising 2.5 minutes flight time.

Hidden Units	MSE Partial ($\times 10^{-4}$)	MSE Full ($\times 10^{-4}$)
40	0.36	0.25
10	1.11	1.13
6	1.89	1.52
2	2.21	2.24

Table 1 – Varied Architecture MSE's after 250 epochs.

Figure 4 shows the test performance at 5 epoch training intervals for a network trained on the same data set. This is for a varying number of hidden units, while using a constant η of 0.02. This learning rate was found to give optimal convergence speed vs final MSE, however any learning rate within the range of 0.01 to 0.04 gave comparable results. The upper four plots are test MSE; the lower four are

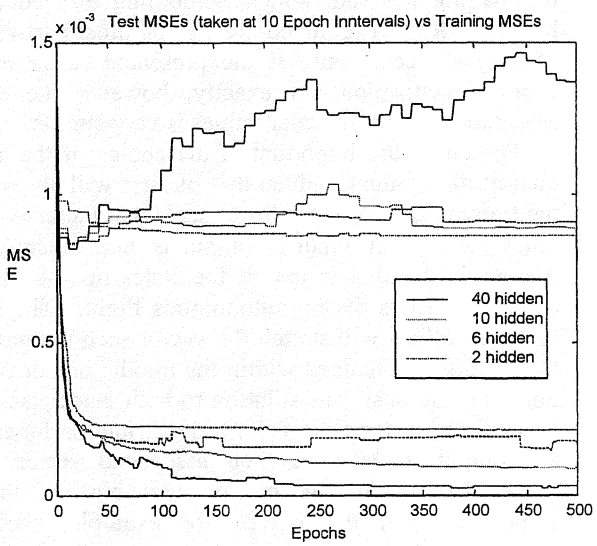

Figure 4 - MSE 's for Varying Hidden Number

training MSE. It was also found that increasing the hidden stretch factor, improved generalisation, however for the 2.5 minute data set, acceptable MSE was never achieved.

Figure 5 shows the training and test MSE for a network with the same parameters trained on a flight lasting more than 6 minutes. Figure 6 compares the elevator sequences produced by the neural net (--) with that

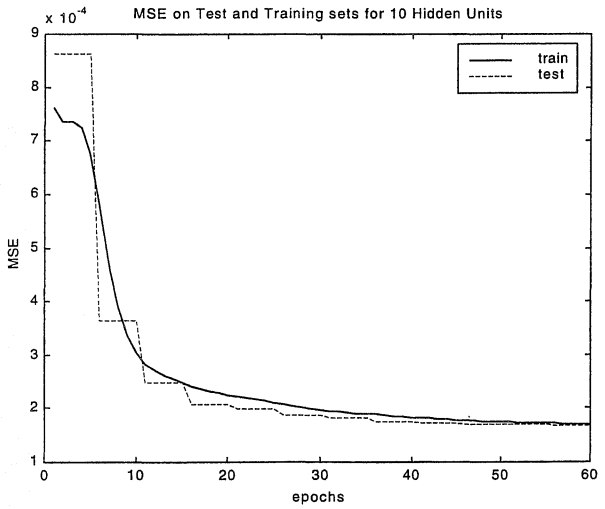

Figure 5 - MSE on 6 Minute Training Sets

of the pilot (full lines). It should be noted that variation of those parameters described for the smaller data set produced no considerable difference in performance when applied to training on the larger set.

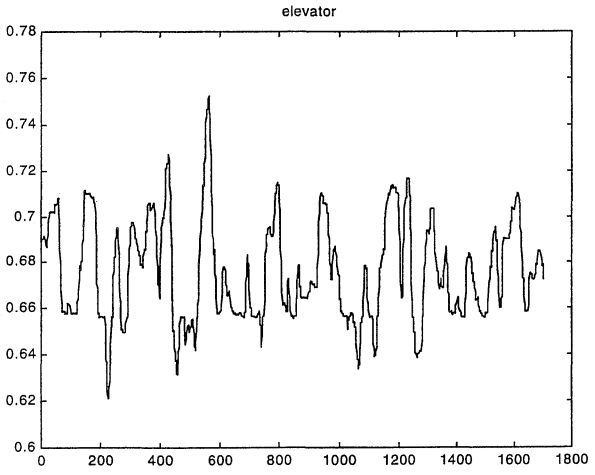

Figure 6 - Elevator Demand vs Time

The only parameter not varied for the larger set was hidden unit number; this was not necessary since acceptable generalisation has seemingly been achieved with a 10 hidden unit architecture. As with the smaller training set, training on the larger set exhibited little performance variation between partially and fully connected networks.

6 Discussion of Results

When considering training merit as a degree of correlation between desired and actual response sequences, rather than as a measure of classification accuracy on a pattern by pattern basis, it can be seen from Figures 5, 6 & 7, that a network with a final MSE of 0.00018 yields a reasonable level of performance.

The results in Table 1 give a fair indication as to the minimum number of hidden units required before considerable MSE degradation occurs. While these were logged from training on the small data set, the choice of architecture to be used when training on the larger data set was made based on these results. The clear relationship between the fully and partially connected network "MSE pairs" (Table 1) furthers this focus on network pruning, demonstrating a degree of synaptic redundancy within a fully connected architecture. The degree of redundancy was found to be even greater when training was carried out using the larger data set. This again allows interconnection minimization with virtually no compromise in network training convergence.

Figures 4 & 5 provide some interesting insight into the training requirements of the system. While the intention of the associated experiments in Figure 4 was to find the optimal number of training epochs before over-learning began, the results in fact indicated that the training regime was suffering from a data deficiency. This conclusion is based on the lack of variation shown for the span of epochs for varying architectures in Figure 4. This is not overly surprising, given a training set size of 2400 samples (around 75 seconds), and a connection number of the order of 5000. A generic approximation for the number of required training patterns is suggested in [6] to be equal to the total number of connections divided by the acceptable error level; this indicates around 40000 samples for a 10 hidden unit architecture aiming for 80% accuracy. This is achievable with 22 minutes of flying time. However, the results illustrated in Figure 5 show the system trains acceptably on data sets far smaller than that estimated by the generic approximation (16000 patterns). This is somewhat in contrast to expected behaviour given that the order of pattern presentation in a temporal network cannot be randomized, as is suggested in [3] & [6] for network classification improvement. This better than expected performance on such a small training set, coupled with the system's relative performance independence to common back propagation parameter variation implies that the hover data, as pre-processed and presented to the network, formulates a relatively well behaved optimization system.

The results indicate that the optimal system architecture is an 800 input, 40 output, 10 hidden partially connected unit network trained on a six minute hover flight.

7 Controller Implementation

The optimal neural network architecture described in the preceding section will comprise the bulk of the controller software residing on the Controller Computer (PC). Interface procedures provide services to the neural net through which it can both access INS input data (state

information), and send controlling demands to the flight computer (HC12). These demands are then forwarded to the servos in PWM form (See Figure 1).

The neural net receives helicopter state updates every 7.6ms. The net processes the updated information in a feed-forward manner in order to produce updated servo demands. While the servos have their own associated closed loop controllers, the overall system is not considered to be in a closed loop configuration in the classical sense. Rather, the control software uses a "cause and effect" approach, such that it's output demands are gauged on the states fed back via the INS in a fuzzy manner. Due to the frequency limitations of the flight computer demand generation resulting from the PWM nature of the controlling signals, the rate of demand update at the actual servos is limited to 22ms. This is of no real concern to the neural net, as it was trained on data logged during a piloted flight subjected to the same hardware restrictions.

The testing regime for the controller will involve the pilot taking the helicopter into a stable state of hover. The controller will then be given control one axis at a time until full autonomous hover is achieved. While preliminary testing will relinquish aileron and elevator control only, it is expected by the end of the year that the other two controls will also be transferred.

8 Conclusions and Future Work

This paper details the preliminary design of an ANN to be used as a purely reactive autonomous helicopter controller, using INS information as its only input stimulus. Particular attention was paid to the preprocessing methodology and the neural network architecture. It is found that a partially connected neural network, in which x-gyro, roll and y-accelerometer are grouped, y-gyro, pitch, and x-accelerometer are grouped, and z-gyro and z-accelerometer are common to both, causes no degradation in training performance compared with a fully connected architecture. It is also found that this problem is particularly ill-suited to bipolar activation units. Varying learning rate exhibits little effect on the overall performance so long as it is kept reasonably small (0.01->0.05). For the smaller data sets, increasing the stretch factor of the hidden layer units gives better (though never acceptable) performance on unseen data. Such a relationship is not evident when the larger training sets are used; convergence rate is swift for all stretch values tested. For the most part, successful training is found to be attributable to the training set.

This paper finds that from an optimization viewpoint, the dynamically complex nature of a helicopter hover is in fact a "well behaved" system. The next step is to trial the controller on the helicopter; testing is expected to begin late July. This will include carrying out investigations into the cause and implications of the neural net divergence on rudder and collective control. Following that is the incorporation of the vision system into the loop. The intended use for the four-point algorithm described in [2] is primarily translational displacement estimation from the helipad center, and possibly velocity estimation relative to the helipad.

References

1.R. Miller, O. Amidi, and M. Delouis, "Arctic Test Flights of the CMU Autonomous Helicopter," *Prepared for the Association for Unmanned Vehicle Systems Int. 1999, 26th Annual Symposium*, http://www.ri.cmu.edu/project/chopper (current January 2000).

2. O. Shakernia, et al., "Landing an Unmanned Air Vehicle: Vision Based Motion Estimation and Nonlinear Control," Asian Journal of Control, Vol. 1, No. 3, pp. 128-45, 1999.

3. J. Hertz, A. Krogh, R.G. Palmer, "Recurrent Networks," *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Company, CA., 1991, pp. 163-196.

4. G.F. Wyeth, *A Hunt and Gather Robot, doctoral dissertation,* Univ. of Queensland, St. Lucia, Dept. Electrical Engineering and Computer Science, 1997.

5. D.S. Moore, and G.P. McCabe, "Looking at Data: Distributions," *Introduction to the Practice of Statistics*, W.H. Freeman and Company, N.J., 1993, pp. 63-74.

6. S. Haykin, *Neural Networks, a Comprehensive Foundation*, Macmillan Publishing Company, N.J., 1994.

7. L.C. Jain, N.M. Martin, *Fusion of Neural Networks, Fuzzy Sets & Genetic Algorithms,* CRC Press LLC, Boca Raton, 1999.