A helicopter named Dolly - Behavioural cloning for autonomous helicopter control

Gregg Buskey[a,b], Jonathan Roberts[a], Gordon Wyeth[b]

[a] CSIRO Manufacturing and Infrastructure Technology
P.O. Box 883
KENMORE 4069, Queensland, Australia
[b] School of Information Technology and Electrical Engineering, University of Queensland
ST LUCIA, Queensland, Australia
Email: Gregg.Buskey@csiro.au

Abstract

This paper considers the pros and cons of using Behavioural cloning for the development of low-level helicopter automation modules. Over the course of this project several Behavioural cloning approaches have been investigated. The results of the most effective Behavioural cloning approach are then compared to PID modules designed for the same aircraft. The comparison takes into consideration development time, reliability, and control performance. It has been found that Behavioural cloning techniques employing local approximators and a wide state-space coverage during training can produce stabilising control modules in less time than tuning PID controllers. However, performance and reliabity deficits have been found to exist with the Behavioural Cloning, attributable largely to the time variant nature of the dynamics due to the operating environment, and the pilot actions being poor for teaching. The final conclusion drawn here is that tuning PID modules remains superior to behavioural cloning for low-level helicopter automation.

1 Introduction

This paper considers the pros and cons of using Behavioural cloning for the development of low-level helicopter automation modules. The Behavioural cloning uses the aircraft state and teacher pilot actions to learn to carry out desired tasks. The approach is then compared to traditional PID modules designed on the same hardware, taking into account such factors as design time, method reliability, and control performance. Testing is carried out on our Xcell-60 experimental platform equipped with inertial, visual, and GPS sensing (Figure 1).

Helicopters are typically described in the literature as having unstable, nonlinear, time varying, coupled dynamics. Nevertheless, suites of PID control modules have been employed to produce autonomous helicopters capable of performing complicated missions [19, 1]. In recent years, more advanced control approaches such as gain scheduling

Figure 1: *Xcell 60 Helicopter Platform*

[21] and linearization feedback [13, 18] have been applied with considerable success. However PID control continues to dominate most small scale helicopter design systems mainly because they are simple, and do not require a dynamic model for design; the development of such dynamic models for small size helicopters has commanded considerable research focus over recent years [14, 11].

One of the problems with such simple PID approaches, is that the tuning of gains is noted to be tedious. Machine learning continues to receive mention as an alternative to hand-tuning PID gains for helicopter automation; self-learning is most commonly suggested [10, 2]. Discounted-future-reward type reinforcement learning has dominated, with successful aircraft stabilisation demonstrated on a Yamaha R-50 experimental platform. However, given the typically slow convergence of value function base reinforcement learning, control synthesis has been necessarily performed offline using an accurate dynamic model; as stated previously such models are far from trivial to construct. The other learning approach is behavioural cloning using a human teacher, which has been demonstrated on tasks such as backing a truck and trailer assembly [22], driving a car [20, 17], general task learning [23, 16], and

the famous pole-cart stabilisation problem [12, 8, 9]. In 1998 behavioural cloning was first applied to helicopter attitude control [15]. While some success was shown in simulation, the controllers failed to stabilise the experimental aircraft [15].

2 Behavioural cloning isn't all the same

Several behavioural cloning approaches were tested before full aircraft stabilisation was achieved. The first approach was to divide the helicopter automation control problem into tasks such as hover, forward, backward, turn etc. An arbitration layer would sit on top of the behavioural layer, switching between them according to some mission objectives (Figure 2). This is a typical

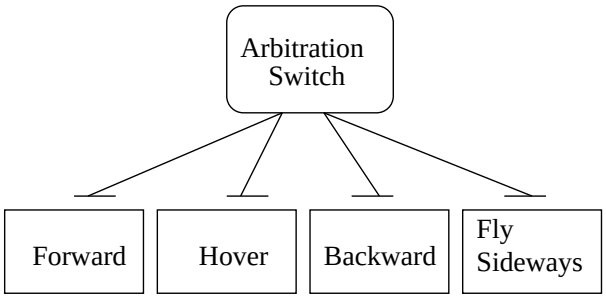

Figure 2: *Task decomposition learning architecture.*

behaviour-based approach. The first behaviour trialed for learning was hover. Hover was chosen because it is logistically the easiest to train (i.e small airspace required), and the most difficult envelope in which to achieve stabilisation. Consequently if hover could be learnt, then so to could the other behaviours. Multi-layer neural networks with and without feedback connections were used as the learning architecture [7].

While it was found that the learned system's pilot mimicking capabilities were excellent, mimicking was occuring using all the wrong cues and for all the wrong reasons [4]. Within such a tight envelope, unobservable effects such as varying wind conditions dominated the dynamics, resulting in training data being information difficient with regard to showing any aircraft-attitude-to-pilot-action relationship. The result was destabilising rather than stabilising control [4].

The second approach looked to expand the training envelope such that unobservable effects would be less dominant. The automation problem was divided in a more traditional manner. Each aircraft axis was considered to be decoupled, with learning applied to each seperately. The resulting architecture, shown in Figure 3, was more representative of the PID modules later developed for comparison. The higher-level layer then simply sends demands such as desired roll, velocity, height etc. to these decoupled low-level modules.

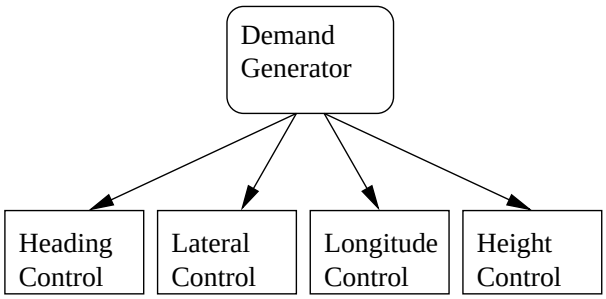

Figure 3: *More traditionally inspired decomposition learning architecture.*

Again multi-layer neural networks were employed as the learning architecture. However, no recurrent connections were included as it was found in [7] that all necessary state history (i.e. to be considered Markovian) was captured by the sensors available. Training of these modules was performed by what we termed destabilisation/restabilisation sequences, in which a single axis would be stabilised (eg lateral velocity induced) and then restabilised (velocity arrested); during the restabilisation phase the module would be mapping the relavant aircraft state to pilot actions.

While it was clear that this approach was providing a more information rich training environment, the feedforward global approximators were exhibiting training instability [6]. Tuning of learning parameters could improve learning stability, but at considerable learning time cost. Besides, such intense tuning of learning parameters defeats the purpose of using learning to avoid tedious tuning of PID gains.

The third approach used the same destabilisation/restabilisation sequences for training, but abandonned the global approximators in favour of the Fuzzy Associative Memory local approximator. FAMs were adopted for two reasons. The first is that there are no learning parameters to tune; the only design choice is fuzzy set placement. During learning, the FAM essentially performs a weighted average of the teacher actions in each cell, and consequently reduces the importance of learning stability issues. Secondly, local approximation means that once you've trained the system in one region of state space, it will not be forgotten if training is concentrated in another region. Thus one could be methodical about the training process; i.e. "we've been there so we don't need to go there again". This is important for online learning where the teacher needs to be able to make an informed decision about the type of action best next performed.

The FAMs trained using the destabilisation/restabilisation sequences successfully demonstrated stabilisation of all axes [5]. These include simple single-input-single-output 1-dimensional FAMs for heading, roll and pitch control, and multi-input-single-output 2-dimensional FAMs for lateral velocity control and

height control. Of course the question remains, "How advantageous was the learning approach when compared to traditional PID".

3 To PID or not to PID - that is the question?

Whether 'tis nobler to bear the slings and arrows of tuning PID controllers, or to take arms against a sea of gains, and by applying machine learning end them. This question of PID vs cloning must be considered in three parts; development time, reliability/repeatability, and control performance.

3.1 Development time

Development time refers to the time taken to either tune the PID controllers, or to train the FAMs. This does not include the development time taken to discredit the earlier cloning approaches. If this method would be only applicable to helicopters, then that would be fair, however given that this approach could be applied to any multi-degree-of-freedom, unstable but human operatable system, only training vs tuning time provides an informative comparison of the two methods.

The PID design approach was highly methodical, stepping incrementally over a range of gains and applying step responses for each gain setting. Responses were analysed offline and the best gain chosen. The design of each control module demanded approximately 15 minutes of flight time, and approximately 1 hour of offline analysis.

The training of the FAMs was an order of magnitude faster. As mentioned in Section 2, the only design choice required for FAM control is input space set division. The pilot would perform a couple of mock training sequences, during which the range of aircraft motion was scrutinised, and set limits determined. Carrying out the mock training runs for set division took around 2 minutes. Training time usually took another 2 minutes, and so rounding for take off and landing we get approximately 5 minutes per control module development time.

3.2 Reliability/repeatability

The repeatability and reliability of correctly designed PID controllers is well documented. Thus the issue of reliability/repeatability applies only to the behavioural cloning. One training reliability issue is input space set division. The number of sets was determined according to the state range seen during the mock training trials, and the pilot's sensing resolution. That is, a pilot cannot distinguish attitudes to less than 1 degree error when the aircraft is 15 meters away travelling at $2ms^{-1}$, so if the maximum range were 5 degrees, dividing the input space into more than five sets would over granulise the input space. Too many sets can cause the lack of pilot resolution to turn into training ambiguity. Too few sets on the other hand results in a loss of our ability to capture the nonlinearity of the pilot's actions.

For the simple 1-dimensional roll, pitch, and heading control modules, this issue of set division had little effect on reliability, such that all training runs conducted produced stabilising controllers. For the 2-dimensional lateral velocity and height controllers on the hand, set placement proved more critical. The problem here was that the optimal set placement could vary greatly between training runs; i.e. the pilot was highly variable in his nonlinear control approach. All height control trials produced stabilising height controllers, however the quality of control varied. Many velocity controller training trials, on the other hand, failed to produce stabilising modules.

Unobservable wind effects were also thought to affect training success. Analysis of the pilot actions, particularly for height and lateral velocity control, shows the pilot taking action that seemingly had no effect on the aircraft state. In other cases the aircraft might stop without any pilot input (wind gust), meaning that the learning system will learn that upon arriving at the destination state, simply maintain the same control input as before, which would clearly cause the aircraft to overshoot in the absence of such environmental effects. Due to these changing conditions, the pilot's actions were often non-monotonic with respect to the input space, introducing considerable ambiguity into the learning process (see Figure 4).

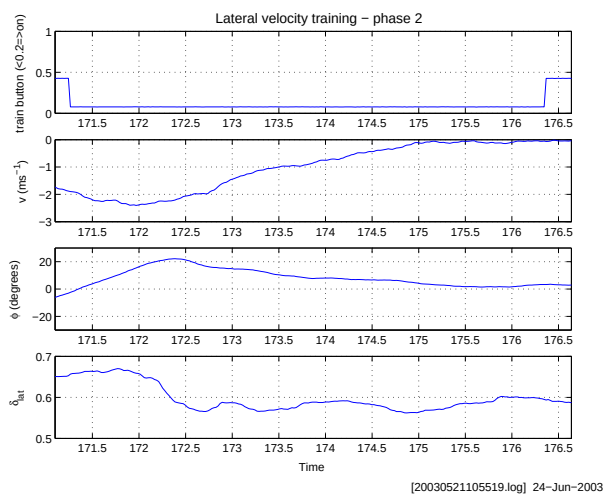

Figure 4: *Pilot stick movements in response to unobservable effects introduce ambiguity in traning data.*

3.3 Control performance

This section presents brief comparisons between FAM and PID control for all axes. A more comprehensive discussion of the FAM control performance and the PID performance can be sourced in [5] and [3] respectively. The reader must also recognise that these experiments were carried out in a highly uncontrolled environment; i.e. real world flight environment. Thus precise quantitative comparisons are for the most part meaningless. The discussion here is restricted to a more qualitative analysis.

The PID and behavioural cloning heading control tracking, illustrated in Figure 5, shows no notable differ-

ence between the two, except for the FAM control being marginally slower and slightly more oscillatory. The roll and pitch control tracking shown in Figures 6 and 7, also show the PID and FAM controllers to have similar performance. In both cases, we note that the PID response is faster than that of the FAM. Furthermore, the effects of not having integral action present in controllers generated using simple cloning can be seen in the roll control tracking plot of Figure 6, where the FAM exhibits some clear steady state error. Not shown here, is that this error varied (see [5]) during a flight depending on the conditions, while the PID integral compensated for the varying dynamics.

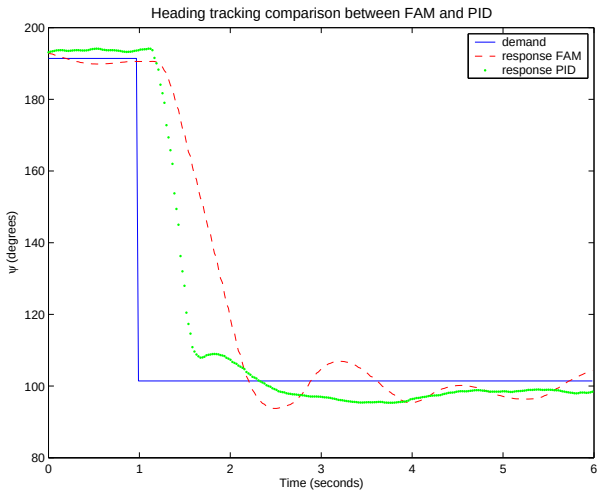

Figure 5: *Comparison between PID and FAM tracking performance for heading control.*

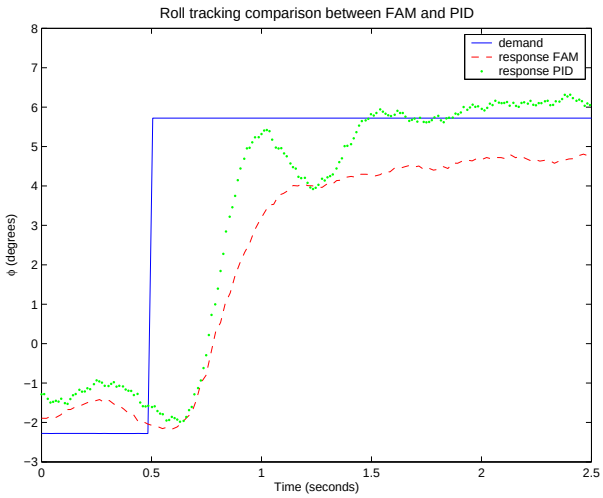

Figure 6: *Comparison between PID and FAM tracking performance for roll control.*

The PID and behavioural cloning height tracking, illustrated in Figure 8, shows the PID response is somewhat slower than the FAM response. However the results presented for PID height control are those achieved after very brief gain tuning. It is expected that the response of the PID control module can be greatly improved. The PID and behavioural cloning lateral velocity tracking, illustrated in Figure 9, shows that the PID indisputably outperforms the FAM module. It should be noted that the results presented here are the worst for both the FAM and PID, however the PID's considerable superiority was observed for best, worst, and average case scenarios (see [5, 3]). Given the problems encountered with learning reliability discussed in Section 3.2, coupled with the comparitively poor control performance (to PID) shown in Figure 9, longitudinal velocity control cloning experiments were not conducted.

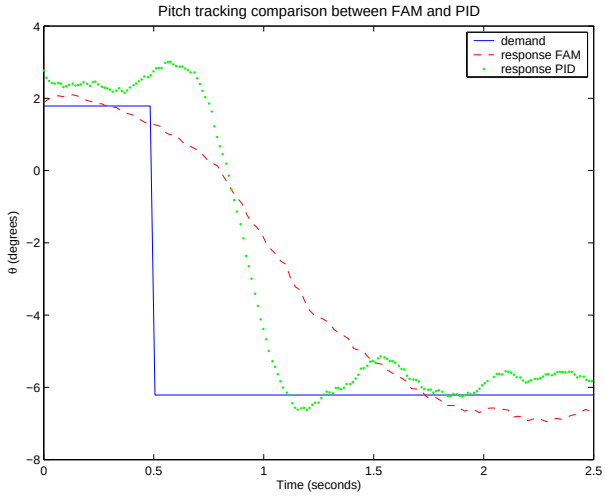

Figure 7: *Comparison between PID and FAM tracking performance for pitch control.*

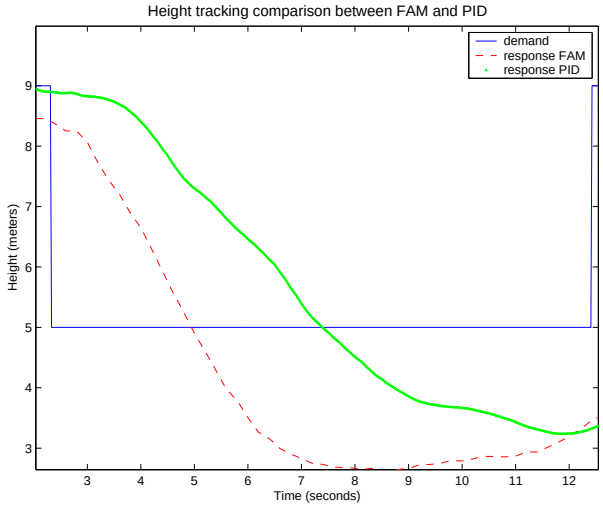

Figure 8: *Comparison between PID and FAM tracking performance for lateral velocity control.*

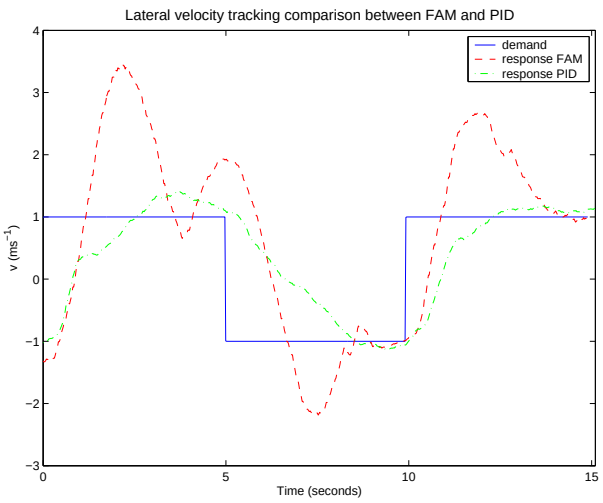

Figure 9: *Comparison between PID and FAM tracking performance for lateral velocity control.*

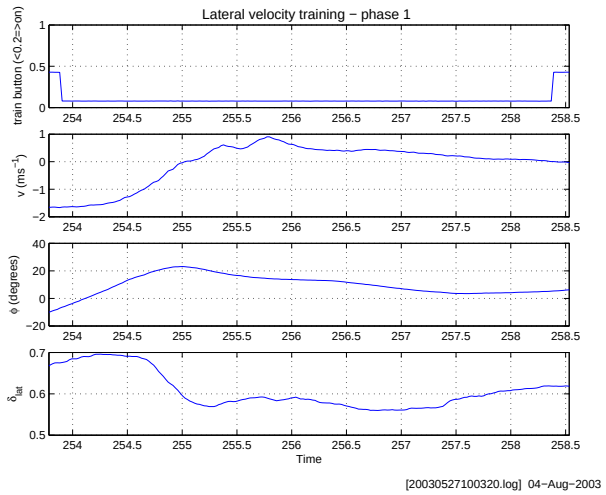

Figure 10: *Pilot overshoots the stabilised (zero velocity) mark by* $1ms^{-1}$.

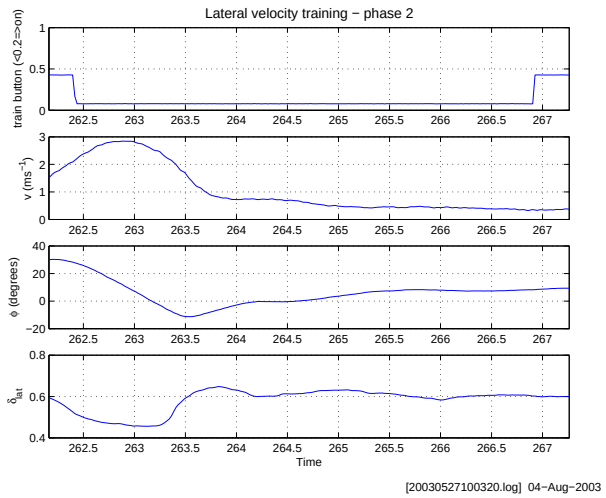

Figure 11: *Pilot settles with an SSE of* $0.5ms^{-1}$.

4 Pilots aren't great teachers

In Section 3.2, several issues regarding learning reliability were highlighted, including optimal set placement variation due to pilot policy variation, and environmental effects varying the dyanmics during the learning process. Another problem, hinted at in Section 3.2 was the pilot's sensing ability. In many of the training trials, the pilot's lack of ability to perform the desired task was noted. This inability would often manifest in the form of large overshoots (see Figure 10) and steady state errors (see Figure 11). Given that the pilot is recognised at both the national and international competitive RC helicopter levels, this cannot be attributed to using a poor pilot for training. The problem reduces to the pilot's inability to sense aircraft velocities and attitudes to within error levels acceptable from a computer controlled viewpoint. The pilot has other higher-level cognitive processes to compensate for their poor sensing abilities. They also have extra sensors not available to the computer, such as being able to detect for wind and pre-compensate for the dynamic effects. These processes are not available to the cloning system, and thus it is trying to learn from a teacher while being given only half the information upon which the teacher's actions are based.

5 Conclusions

This paper has compared the two seperate streams of control research conducted on the CSIRO autonomous helicopter project; traditional PID and FAMs generated using behavioural cloning. It has been found that the behavioural cloning approach is faster than tuning PID gains for generating heading, roll and pitch control modules. The cloning approach was found to reliably generate these modules, with comparable control performance to the PID controllers. However the absence of integral action in the cloned FAM modules resulted in them exhibiting some susceptibility to wind conditions. Generation of the more complicated height and lateral velocity controllers using cloning, on the other hand, was found to be less impressive. Control quality greatly varied between learning trials, and on no occassion was the velocity control quality of the cloned system's performance comparable to the tuned PID modules. It is expected that this same quality deficeit will exist for the height control once the PID module is properly tuned. This is attributable in part to the input set division, which cannot be well defined a priori due to the pilot's policy variation from trial to trial. However it is more attributable to varying dynamics during training due to changing wind conditions, and the pilot's inability to perform the desired tasks, or largely to sense the aircraft's

state, to within accuracy levels necessary for a reactive system to learn from state-action duples.

Acknowledgments

The authors would like to thank the whole automation team for their invaluable assistance and support. In particular, Peter Corke, Craig Worthington, Les Overs, Stuart Wolfe, Steven Brosnan, Pavan Sikka, Graeme Winstanley, Mathew Dunbabin, Elliot Duff, and our pilot Fred Proos.

References

[1] O. Amidi. *An Autonomous Vision-Guided Helicopter*. PhD thesis, Dept of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, 1996.

[2] J.A. Bagnell and J.G. Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In *International Conference on Robotics and Automation*, 2001.

[3] G. Buskey, J. Roberts, P. Corke, and G. Wyeth. Helicopter automation using pid and low cost avionics. In *Australasian Conference on Robotics and Automation*, Brisbane, Australia, December 2003.

[4] G. Buskey, J. Roberts, and G. Wyeth. Online learning of autonomous helicopter control. In *Australasian Conference on Robotics and Automation*, Auckland, New Zealand., December 2002.

[5] G. Buskey, J. Roberts, and G. Wyeth. Experiments in learning helicopter control from a pilot. In *International Conference on Field and Service Robotics*, Japan, July 2003.

[6] G. Buskey, J. Roberts, and G. Wyeth. Learning autonomous helicopter control from a pilot. In *Proceedings of the International Conference on Advanced Robotics*, 2003.

[7] G. Buskey, G. Wyeth, and J Roberts. Autonomous helicopter hover using an artificial neural network. In *IEEE International Conference on Robotics and Automation*, pages 1635–1640, Seoul Korea, May 2001.

[8] R.A. Chambers and D. Michie. Man-machine cooperation on a learning task. In R. Parslow, R. Prowse, and R. Elliott-Green, editors, *Computer Graphics: Techniques and Applications*. Plenum, London, 1969.

[9] P.E.K. Donaldson. Error decorrelation: A technique for matching a class of functions. In *Proceedings of the Third International Conference on Medical Electronics*, pages 173–178, 1960.

[10] R. Enns and J. Si. Helicopter tracking control using direct neural dynamic programming. *Proceedings of the International Joint Conference on Neural Networks*, 2:1019–1024, July 2001.

[11] V. Gavrilets, B. Mettler, and E. Feron. Nonlinear model for a small-size acrobatic helicopter. In *AIAA Guidance, Navigation and Control Conference*, pages 1593–1600, Montreal, Canada, August 2001.

[12] E. Grant and B. Zhang. A neural-net approach to supervised learning of pole balancing. In *IEEE International Symposium on Intelligent Control*, pages 123–129. IEEE, September 1989.

[13] T.J. Koo and S. Sastry. Outputtracking control design of a helicopter model based on aproximate linearization. *IEEE Conference on Decision and Control*, 4:3635–3640, 1998.

[14] B. Mettler, T. Kanade, and M.B. Tischler. System identification modeling of a model-scale helicopter. Technical Report CMU-RI-TR-00-03, Robotics Institute - CMU, March 2000.

[15] J.F. Montgomery. *Learning Helicopter Control through "Teaching by Showing"*. PhD thesis, University of Southern California, May 1999.

[16] U. Nehmzow. Application of robot training: Clearing, cleaning, surveillance. In *Proceedings of International Workshop on Advanced Robotics and Intelligent Machines*, Salford, UK, April 1995.

[17] D.A. Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. PhD thesis, Carnegie Mellon University, 1992.

[18] J.V.R. Prasad, A.J. Calise, Y. Pei, and J.E. Corban. Adaptive nonlinear controller synthesis and flight test evaluation. In *Proceedings of the 1999 IEEE International Conference on Control Applications*, pages 137–142, Kohala Coast-Island of Hawaii, August 1999.

[19] C.P. Sanders. Hierarchical control of small autonomous helicopters. *IEEE Conference on Decision and Control*, 4:3629 – 3634, 1998.

[20] J.F. Shepanski and S.A. Macey. Teaching ann systems to drive: Manual training techniques for autonomous systems. In *Neual Information Processing*, pages 693–700. American Institute of Physics, New York, 1988.

[21] K. Sprague, V. Gavrilets, D. Dugail, B. Mettler, and E. Feron. Design and applications of an avionics system for a miniature acrobatic helicopter. In *Digital Avionics Systems Conference*, pages 3.C.5–1 – 3.C.5–10, Daytona Beach Florida, October 2001.

[22] Li-Xin Wang and J.M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1414–1427, November/December 1992.

[23] G.W. Wyeth. *A Hunt and Gather Robot*. PhD thesis, Eept of Electrical and Computer Engineering, University of Queensland, 1998.