

---

# Goal-directed Learning to Fly

---

Andrew Isaac  
Claude Sammut

ISAACA@CSE.UNSW.EDU.AU  
CLAUDE@CSE.UNSW.EDU.AU

School of Computer Science and Engineering, University of New South Wales, Sydney NSW 2052, Australia

## Abstract

Learning to fly an aircraft is a complex task that requires the development of control skills and goal achievement strategies. This paper presents a behavioural cloning system that learns to successfully fly manoeuvres, in turbulence, of a realistic aircraft simulation. A hierarchical decomposition of the problem is employed where goal settings and the control skill to achieve them are learnt. The benefit of this goal-directed approach is demonstrated in the reuse of the learnt manoeuvres to a flight path that includes novel manoeuvres. The system is based on an error minimisation technique that benefits from the use of model tree learners. The model trees provide a compact and comprehensible representation of the underlying control skill and goal structure. The performance of the system was improved by compensating for human reaction times and goal anticipation. We conclude that our system addresses current limitations of behavioural cloning by producing robust, reusable and readable clones.

## 1. Introduction

An expert predominately relies on highly developed tacit skills to competently perform dynamic real-time control tasks. Typically, the expert can only give an approximate and incomplete description of these tacit skill. For knowledge acquisition of the tacit skills “machine learning tools can, however, recover symbolic models of skills from behavioural traces. The resulting data-derived ‘clones’ show transparency and run-time dependability” (pp 1, Michie, 1993). In this paper, we present a novel technique for behavioural cloning that learns hierarchical rule structures in combination with classical control methods. The robustness, generality and transparency of the system are demonstrated by

application to a simulated piloting domain.

Behavioural cloning was originally formulated as learning ‘reactive’, situation-action decision trees from behavioural trace data. A limitation of the situation-action formulation is that it is often difficult to understand because it lacks goal structure. It is also not very robust to variation (see Bratko et al., 1998 for a review of early behavioural cloning work). To address these limitations, several researchers have adopted a hierarchical decomposition of the learning task. Bain and Sammut (1999) and Šuc (2001) present a two level ‘goal-directed’ framework that learnt the effects of control actions and a model of the goals directing the control.

These approaches have had some success but often by compromising either transparency or robustness. We take a different approach in learning control actions as an approximation of an operator’s reactions to any difference between the anticipated and actual state of the system and (as above) learn models of the goals directing the control. That is, the system models control skill as being separable into a reactive level and an anticipatory level. This approach allows us to combine traditional PID (Proportional Integral Derivative) controllers into a goal-directed hierarchy. The resulting controllers are robust and easy to understand. Thus, we have avoided many of the compromises found in previous methods.

To assess the robustness, generality and transparency of our system, we have chosen the task of controlling a simulated aircraft in turbulence. The flight domain is complex: the task context changes during the course of the flight; there are many cues and system variables to consider; the aircraft dynamics are complex and non-deterministic (with turbulence) and flying requires both predictive and reactive control, often based on the same cues.

A particular advantage of a goal-directed structure

over situation-action rules is that we can build controllers that are quite general. A situation-action formulation is difficult to decompose and therefore tends to work only for entire flight plans, with little variation. However, our goal-directed approach allows the system to learn a library of discrete manoeuvres that can be combined to create a controller for a wide variety of tasks. The generality of the clones can be tested by their performance over an arbitrary flight path. To avoid learning manoeuvres that may be specific to a particular task, we adopt a flight training approach similar in spirit to pilot flight training (Thom, 1998) that consists of two main steps. Namely, fly and learn one basic aircraft manoeuvre at a time and then test the (auto-)pilot by flying a takeoff and land circuit.

By performing a circuit consisting of variations on the learnt manoeuvres and by varying the degree of turbulence, we demonstrate how the clones of each given manoeuvre can be reused – without modification – in combination to successfully fly the circuit with a high degree of robustness. The induced rule sets are more compact and more transparent than previously found (Sammur et al., 1992; Bain & Sammur, 1999; Camacho, 2000).

## 2. Related Work

One of the novel features of our system is the learning of PID controllers as rule sets and combining them into a goal-directed hierarchical framework. Other control theoretic techniques have been applied to both goal and control learning, however they have either been hand crafted or based on complex numerical modelling. Stirling (1995) used an expert supplied control influence matrix and highly specified task structure to guide an exhaustive search of the goal space. This was combined with manually configured proportional controllers and the system was able to handle catastrophic failure of a control. Šuc (2001) demonstrated the robustness of control theoretic techniques in several dynamic control domains that have less well defined task structure than the flight domain. To improve the readability of the learnt numerical models they were converted to qualitative models describing monotonic regions of given state quantities. To produce working controllers, however, required the engineering of proportional controllers based on the qualitative constraints and domain knowledge. The whole process obscures the interpretation of any recovered skill knowledge. Camacho (2000) used a (turbulence-free) *F-16* fighter jet simulation<sup>1</sup> and learnt a level

<sup>1</sup>The ACM public domain flight simulator. <http://www.websimulations.com>

left turn that was robust to variation in initial conditions and (fixed) goal setting. The learnt control rules were proportional controllers based on first and second derivatives of an error term. The error was not goal-directed but based on a fixed goal setting for the manoeuvre. For our work, learning PID controllers as rule sets has the advantage of combining the robustness of the PID control framework with the relative simplicity and transparency of rule learners. PID controllers have been widely studied for Control Engineering (Franklin et al., 1995) and are well suited to non-deterministic dynamic control systems. Furthermore, by combining the controllers with a goal level we have the advantage of hierarchical and goal-directed control mentioned above. Such a system also has the potential to be combined with more deliberative procedural knowledge in a logic programming framework (Bain & Sammur, 1999).

To date, most of the flight simulators used in behavioural cloning research have not used turbulence and have been based on simple aircraft flight models. We use a simulator with a sophisticated flight model and added turbulence. The original work of Sammur et al. (1992) flew a takeoff and land loop of a runway with no turbulence using a handcrafted flight model of a *Cessna 150* light aircraft.<sup>2</sup> Using the same (turbulence-free) simulator, Bain and Sammur (1999) learnt individual manoeuvres and were able to reuse them to fly the majority of the takeoff and land loop, but the system was unable to perform the line-up and land stages. These systems used decision tree learners that required discrete state and action values. We have chosen rule based techniques that can handle continuous values, namely regression trees and model trees.

We develop our behavioural clones by flying a variety of goal settings and then learning a library of individual manoeuvres. Other behavioural cloning systems have been developed to recover a goal model from trace data over an entire flight. van Lent and Laird (1999) describe a system that uses a multi-level hierarchy and learns performance goals at each level of the hierarchy. To construct the system required the assistance of an expert to develop the hierarchies, categorise the state values and primitive actions and annotate the behavioural trace. Anderson et al. (2000) applied a modular neural network to learn a model of the pilot (as opposed to learning a model of how to fly considered here). The system automatically decomposed the task and required no extra input from the expert. The

<sup>2</sup>The simulator was a modified version of the Silicon Graphics Dog Fight simulator.

interpretation of what the system had learnt, however, is difficult. In this paper we avoid learning a specific flight plan by developing a library of clones to perform discrete manoeuvres. The only additional effort we require of an expert is the identification of goal-directing quantities. Using rule-based models, we argue, retains the transparency of the acquired control knowledge and produces compact rules.

### 3. Hierarchical Representation

Learnt skills are represented by a two level hierarchical decomposition with an anticipatory *goal* level and a reactive *control* level. The goal level models how the operator chooses goal settings for their control strategy and the control level models the operator’s reaction to any error between the goal setting and actual state of the system. For example, in flying, the pilot can achieve goal values for the desired heading, altitude and airspeed by choosing appropriate values of turn-rate, climb-rate and acceleration. The controls can be set to correct errors between the current state and the desired state of these goal-directing quantities. Goal models map system states to a goal setting. Control actions are based on the error between the output of each of the goal models and the current system state.

The control level is modelled as a set of *proportional integral derivative* (PID) controllers, one for each control variable. A PID controller determines a control value as a linear function proportional to the error on a goal variable, the integral of the error and the derivative of the error. The basic form is

$$control = P \cdot error + I \cdot \int error dt + D \cdot \frac{derror}{dt} \quad (1)$$

where coefficients  $P$ ,  $I$  and  $D$  are set to values so as to optimise the control of the system. The integral term refers to accumulated error and the derivative term refers to instantaneous changes in goal setting.

As an example of the hierarchical representation a goal and control rule for a left turn is presented in Table 1. The goal rule determines the goal setting for *turn-rate* based on the relative azimuth and distance to the goal position. The *error*, *integral* and *derivative* are calculated with respect to the goal-directing value and the control rule determines the value to set the *aileron*.

#### 3.1. Implementation

The goal-directed control system interacts with the environment by determining the desired value for each goal variable based on the current state, comparing the desired values to the current values and determining each controllers’ action to reduce the errors. The output of the control models are delayed to prevent

instantaneous control actions and to approximate human reaction time delays.

Table 1. Sample Goal and Control rules.

---

GOAL RULE
IF ( $azimuth \leq -3.6$ ) THEN
IF ( $distance \leq 1715$ ) THEN
$t = 0.163 \cdot azimuth + 0.0013 \cdot distance - 1.90$
ELSE
$t = 0.073 \cdot azimuth + 0.0003 \cdot distance - 0.92$
ELSE
$t = 0.006 \cdot azimuth + 0.0022 \cdot distance - 1.93$
CONTROL RULE
IF ( $i \leq 11.71$ ) THEN
IF ( $d \leq -1.84$ ) THEN
$a = -0.059 \cdot e - 0.0091 \cdot i - 0.0207 \cdot d - 0.139$
ELSE
$a = -0.046 \cdot e + 0.0012 \cdot i + 0.0001 \cdot d - 0.005$
ELSE
$a = -0.043 \cdot e - 0.0236 \cdot i - 0.0122 \cdot d + 0.329$
WHERE,
$t =$ GOAL-DIRECTING TURN-RATE
$a =$ AILERON CONTROL VALUE
$e =$ CURRENT TURN-RATE – GOAL-DIRECTING TURN-RATE
$i =$ INTEGRAL OF $e$
$d =$ DERIVATIVE OF $e$

---

Controllers have been built for a variety of manoeuvres, such as turns, climbs and straight-and-level flight. This allows us to produce a library of controllers for use in complex tasks with multiple goal conditions. In our autopilot example, we hand crafted the flight plan to determine the appropriate controller and goals to use for each stage of the flight plan. However, by annotating controllers with pre- and post-conditions, it is possible to use the library in a planning system. This is future work.

#### 3.2. Learning Goal Settings

The learning process is simplified by learning the goal and control models separately. First, we describe how goal rules are learnt. The process begins by deciding which variables are to be used for the goal settings. For learning to fly an aircraft, standard flight instruction manuals (Thom, 1998) make this choice clear. For example, trainee pilots will learn to execute a “constant-rate turn”, that is, their goal is to maintain a given turn rate. In general, we make the reasonable assumption that an expert can identify goal variables.

Rather than collecting behavioural traces over an entire flight, pilots are only asked to execute specific ma-

noeuvres, each time ensuring that a particular goal setting is maintained. Data are collected for a variety of set values, such as a turn-rate of  $180^\circ$  per minute and zero climb-rate or zero turn-rate and  $500 \text{ ft}$  per minute climb-rate. Absolute positions and orientations are converted into quantities relative to the goal values set for each trial. All the behavioural traces for one manoeuvre were then combined. Although the goal value is given in advance, we found that some improvement could be gained by using the average value actually obtained by the pilot, rather than set value. This results in a controller that more faithfully reproduces the pilot’s behaviour.

A separate goal rule is constructed for each goal variable. In our first attempt at predicting goal values, we used a re-implementation of the CART regression tree learner (Breiman et al., 1984). This was later extended to produce model trees, similar to Quinlan (1993). The model tree learner was found to produce better performing and more compact models. The minimum node size is the only parameter supplied.

### 3.3. Learning Control Actions

From the goal rules, we can obtain the setting for a goal variable and therefore, we can find the difference (error) between the current state value and the goal setting. We augment the data set by calculating, for each record, the integral and derivative of the error. For example, if the set turn-rate is  $180^\circ$  per minute, then the error on the turn-rate is calculated as the actual turn-rate minus 180. The integral is then the running sum of the error multiplied by the time interval between time samples, starting from the first time sample of the behavioural trace, and the derivative is calculated as the difference between the error and previous error all divided by the time interval.

For each control available to the pilot (e.g. elevators, ailerons, throttle, flaps, etc), a model tree learner is used to predict the appropriate control setting. Linear regression is used in the leaf nodes of the model tree to produce linear equations whose coefficients are the  $P$ ,  $I$  and  $D$  of values of Equation 1. Thus the learner produces a collection of PID controllers that are selected according to the conditions in the internal nodes of the tree (see Table 1). In control theory, this is known as piece-wise linear control.

Originally, a regression tree learner was used, however, it produced very large trees that were difficult to interpret. Furthermore, since the leaf node contained only a single average value, the regression trees did not produce continuous control actions. The use of model trees to produce PID controllers means that we get

continuous control and while the PID equations break the logical interpretation of the tree, their simplicity and compactness makes them easy to understand.

### 3.4. Learner Parameter Selection

For the convenience of the pilots, it is desirable to keep to a minimum the number of trials they are asked to perform. Thus, the small number of behavioural traces we can obtain forms a quite sparse data set. This makes constructing the controller difficult. In particular, pruning against unseen data is a problem. Whereas pruning often begins by partitioning the original data and leaving aside some proportion of examples for testing, we have a simulation that can create new data “on the fly”. Thus, our process for building controllers is to iteratively construct a controller, test it by running it as an autopilot, repeating until a trial is completed successfully.

We proceed by first constructing a complete tree and successively pruning leaf nodes. Since we construct one goal tree for each goal variable and one control tree for each control action, there are many combinations of model trees to examine. To determine the best performing combination, we use a wrapper (Camacho, 2000) to run flight trials for each combination of trees.

Fifteen trials are run for each combination, with five different starting locations (see below). All trials are run with the highest turbulence setting ( $10 \text{ ms}^{-1}$ ). Each combination is performed with the same fifteen random seeds for the fifteen trials and the performance of each combination compared. The performance of the controller is tested by measuring the mean sum of squares error value (SSE) at the goal point. The SSE equation is

$$\text{horiz}^2 + \text{vert}^2 + (3.1 \cdot \text{heading})^2 + (1.55 \cdot \text{roll})^2 \quad (2)$$

where the heading and roll are scaled such that a horizontal or vertical error of  $31 \text{ m}$  ( $100 \text{ ft}$ ) has the same weight as a heading error of  $10^\circ$  or a roll of  $20^\circ$ . The combination of rule sets that produced the lowest mean SSE is chosen as the best.

## 4. The Flight Task

Having described our method for learning, we now describe the experiments that were undertaken with the flight simulator. As mentioned previously, flying is usually taught as a set of basic flight manoeuvres where a particular rate is maintained, for example, in a basic turn manoeuvre the pilot aims to maintain a constant turn-rate (e.g.  $180^\circ$  per minute). The basic manoeuvres flown were straight-and-level flight (*Level*), climb and descent (*Climb*), left and right level turns

(*Turn*), takeoff, and descent-to-land (*Landing*).

#### 4.1. The Aircraft and Simulator

The flight simulator is based on a high-fidelity flight model of a high-performance aircraft, thus providing more realistic aircraft control data. The aircraft modelled is the Pilatus PC-9 aerobatic aircraft. The PC-9 is an extremely fast and manoeuvrable propeller aircraft used by the Royal Australian Air Force (RAAF) as a ‘lead in fighter’ for training fighter pilots before they progress to jet fighters. The model was provided by the Australian Defence Science and Technology Organisation (DSTO) and is based on wind tunnel and in-flight performance data.

The controls for the aircraft are the ailerons, elevators, throttle, flaps and gear levers. For our task, we did not use rudder control. The mouse was used to control the ailerons and elevators. Learning aileron and elevator control is the most difficult so we will concentrate on those. The throttle, flaps and gear were all controlled via the keyboard. The ailerons primarily control the roll of the aircraft, which in turn, controls the turn-rate. The elevators control the pitch and consequently affect the climb-rate. The throttle controls the thrust, which directly effects the airspeed, which in turn, effects the climb-rate. The flaps are mostly used during takeoffs and landings to provide extra lift at low speed. The gear is retracted during flight.

Turbulence was added to the flight model as a random offset to the velocity components of the aircraft. Turbulence was scaled by a parameter ranging from 0 to 10, where 10 equates to a maximum displacement of  $10\text{ ms}^{-1}$  ( $19\text{ kt}$ ) in the vertical direction and  $5\text{ ms}^{-1}$  ( $10\text{ kt}$ ) in the horizontal direction. A turbulence setting of  $10\text{ ms}^{-1}$  represents a maximum of a 5% deviation in the forward velocity, a maximum of a ten times deviation in the sideways velocity, and a maximum of a seven times deviation in the vertical velocity. The turbulence was highly exaggerated to give a visual cue to the occurrence of the turbulence as there is no physical feedback. In practice, the effect of turbulence is less near ground level. We model this by reducing the magnitude of the turbulence down to zero from  $100\text{ ft}$  to  $10\text{ ft}$  and having no turbulence at less than  $10\text{ ft}$ .

Figure 1 presents a sample screen image during a takeoff manoeuvre. The simulator world is presented as flat polygons on a grid world  $20\text{ km}$  by  $20\text{ km}$ , this is a low fidelity presentation and visual cues are limited. To provide visual cues to the flight path, we provide simple ‘target windows’. To denote a transition between stages of the flight task, there is a red dot bounded by a  $200\text{ ft}$  square track window, oriented

perpendicular to the target heading (i.e. the aim is to fly straight through the square as close to the centre as possible). In addition to 17 real valued and four integer valued state and control variables, the simulator output the aircraft relative distance, azimuth and elevation to the centre point of each track window.

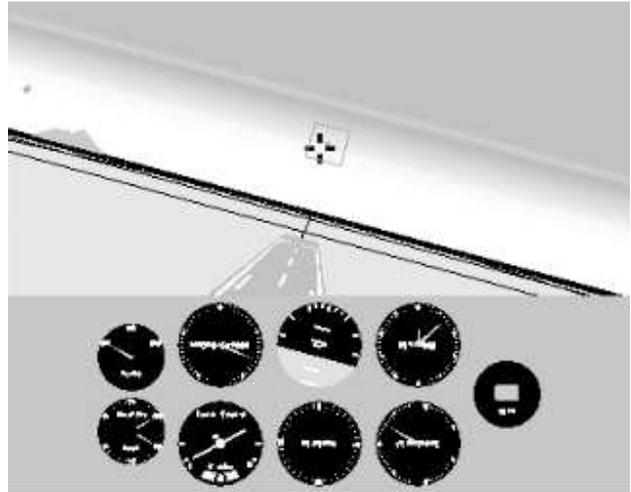


Figure 1. Sample simulator screen image, showing the view when heading toward the Takeoff goal window.

#### 4.2. Data collection and Processing

To produce a behavioural clone we collected behavioural trace data from a variety of flight conditions. This was done to promote data diversity. In all data collection trials, the turbulence was set to  $10\text{ ms}^{-1}$ . For conditions other than takeoff, five start positions were used, the centre of the start track, and the four corners  $100\text{ ft}$  from the centre. The corner starts were also angled away from the track by  $10^\circ$ . Four different altitudes and orientations were used for Straight-and-level, four different climb-rates (two positive and two negative, i.e. descents) for Climb, and four different turn-rates (two positive and two negative) for Turn manoeuvres. An extra set of Climb manoeuvres was also flown with flaps set to the takeoff position. To balance the data sets, there were equal numbers of each variation of a manoeuvre.

The first step in processing the data is to convert the position and heading values to goal relative values for each record. The next step is to estimate the reaction time delay between a stimulus state occurring and the subsequent control reaction. Sammut et al. (1992) tested a range of time delays to determine experimentally the best time delay.

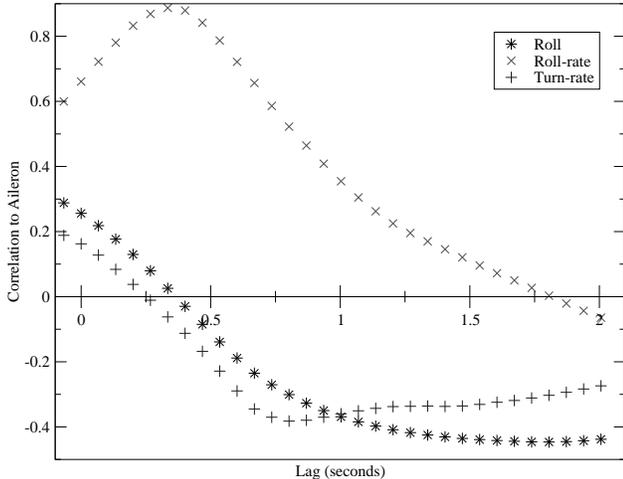


Figure 2. Variation in Aileron Correlation with Lag.

We explored the use of state delays by performing a lag analysis against the aileron control. Figure 2 presents the variation in correlation between the given variables and the aileron for different delays. In the case of zero delay, i.e. when the control reaction occurs instantaneously with the current state, roll, roll-rate and turn-rate are all positively correlated with the aileron. This would suggest a positive feedback control where the pilot reacts by increasing these values. Realistically, the pilot’s reaction will occur some time after the stimulus state. The roll and turn-rate correlation changes to a negative correlation after a 0.5 second delay, this means that there is a (desired) negative feedback where the pilot reacts to reduce the given stimulus states. The plateau in correlation for roll and turn-rate in the 0.7 to 2 second delay time range, also, suggests that once a delay is chosen in this region the exact value is less critical. We chose a 0.7 second delay that corresponds to the peak in correlation for turn-rate and aileron. To process the data, the state variables were placed into a circular buffer and output with the control actions that occurred approximately 0.7 seconds later for each training record. Goals were learnt by using the same time delay, this time to align the goal-directing variables’ states with the system state that occurs 0.7 seconds later. That is, the pilot’s anticipation of the future state is approximated. Without this anticipation offset, the clone did not direct the control toward the goal quickly enough. The use of an anticipatory delay needs further investigation. Sammut et al. (1992) also considered only records where a change in control occurred and for aileron and elevator only the record where the end of the control adjustment occurred. We use the same

data filtering.

The final step in the data processing is to calculate the error, integral and derivative terms for each goal setting as described above.

## 5. Results

Several experiments with this framework have been conducted. The first was to test how well our goal-directed PID control framework performed on individual manoeuvres. We first learnt the control rules only, and handcrafted simple goal rules for each manoeuvre. The behavioural clones were tested on fifteen trials in full turbulence ( $10 \text{ ms}^{-1}$ ) with the five starting positions described above. Straight-and-level flights, climbs and descents were all successfully performed for all fifteen trials (i.e. passed through the target window for all trials).

With the hand-crafted goal rule, the turn manoeuvre rarely succeeded. To improve our handcrafted goal rule, we learnt a goal rule for turn-rate and inspected the model tree. This led to insights into the goal structure that contradicted our intuitive understanding of the pilot’s behaviour. Bratko and Urbančič (1999) explored the effectiveness of Machine Learning in aiding skill reconstruction. They observed that in the case where a person has no common-sense model of the system, help from Machine Learning is essential. In our case Machine Learning provided the insight into where our common-sense model was failing.

Our next experiment was to build clones for each manoeuvre by learning the goal and control rules. In all cases using learnt goal rules produced better performing clones than using handcrafted goal rules. The improved success of the clones with the addition of learnt goal rules indicates that the clones’ robustness to turbulence is not just a result of using learnt PID controllers but also of the learnt goal rules.

To investigate how the goal-directed PID control framework improves manoeuvre performance we conducted a baseline comparison to a situation-action clone. The situation-action clone has no goal level and models control action as a response to system state using goal relative values and was learnt with model trees. There were 15 trials all with  $10 \text{ ms}^{-1}$  turbulence and five different start positions. The situation-action clone successfully flew 11 Level, 9 Climb, 6 Left, 15 Takeoff and 9 Landing manoeuvres. The goal-directed PID controller was successful for all trials of all manoeuvres.

An experiment was performed to test the generality and goal-directedness of our behavioural cloning

Table 2. Per stage success rate (%) for given Clone style and level of turbulence.

CLONE STYLE TURBULANCE ( $ms^{-1}$ )	SITUATION-ACTION 1	GOAL-DIRECTED PID										
		1	2	3	4	5	6	7	8	9	10	
TAKEOFF	99	100	100	100	100	100	100	100	100	100	100	98
LEFT CLIMB	37	100	99	95	87	86	78	76	76	82	69	
OUTBOUND	14	100	100	100	100	100	99	98	97	98	95	
LEFT 1	7	100	100	98	98	80	85	83	79	76	78	
DOWNWIND	0	100	100	100	100	100	99	97	93	92	90	
LEFT 2	0	100	100	100	97	92	81	73	73	65	63	
INBOUND	1	100	100	100	100	96	94	82	87	88	77	
LINE UP	0	100	99	98	90	95	89	80	87	77	83	
LANDING	1	100	100	100	97	93	92	78	82	78	73	

framework. The clones’ performance over a standard takeoff and land circuit of a runway (Thom, 1998) was investigated. Our runway circuit consists of the following stages.

- Takeoff and climb to 500 *ft* with takeoff flap and retract landing gear once airborne.
- Left climbing turn through 90° to 1000 *ft*.<sup>†</sup>
- The outbound leg for 2 *km* at 70% throttle.
- Level left turn through 90°.
- Downwind leg for 5 *km*.
- Descending left turn through 90°.<sup>†</sup>
- Descending inbound leg for 2 *km* with 50% throttle.
- Descending left turn down to 500 *ft* to line up with the runway with flaps partially extended.<sup>†</sup>
- Descent-to-land with flaps fully extended, landing gear down and reducing airspeed.
- Landing near the start of the runway with airspeed, roll, heading, drift and descent-rate all within tolerance.

Note that the circuit contains manoeuvres (indicated by †) that were never learnt, such as a left climb or left descent.

Table 2 presents the success rates of the goal-directed PID control clone at each stage for various levels of turbulence. Also shown in the column labelled ‘Situation-Action’ is the success rate of the best performing situation-action clone for the circuit with a turbulence level of 1  $ms^{-1}$ . A stage is counted as successfully

flown when the aircraft flies through the target window of the stage. Landing is successful when the aircraft lands on the runway within certain tolerances. One hundred trials of the circuit were flown for each set level of turbulence.

To compare the robustness over the flight path the situation-action controller was optimised by ‘tuning’ the learning parameters for every rule for every stage. That is, the clone was learnt specific to the flight path. In this case it was able to fly the entire circuit successfully with no turbulence, however, for a turbulence setting of 1  $ms^{-1}$  the performance over the circuit deteriorated rapidly. This highlights the situation-action’s brittleness. In comparison the goal-directed framework is far more robust. The goal-directed framework is able to recover after missing track windows. For example, when the turbulence was set to 3  $ms^{-1}$  the left climb, first left turn and line-up stages have cases where the goal is missed. The following stages (outbound, downwind and landing) all have 100% success. This is reflected at all turbulence levels except for some of the landing stages. In general the success of the turn manoeuvres deteriorated quickly with an increase in turbulence level. This can be related to the fact that turbulence has the greatest effect on the sideways velocity of the aircraft.

The transparency of the system can be examined from Table 1. The rules are compact and demonstrate a high degree of robustness and generality. In comparison, the rules for the situation-action controller were over twice as long. The use of a relatively simple rule learner and PID control has improved the transparency of the behavioural clone.

## 6. Future Work and Conclusion

To simplify data processing we learnt goals and controls separately. This is possible for the flight domain

where control can be segmented into manoeuvres defined by homeostatic control. Other domains may not have such a natural decomposition and we are currently testing our system in less structured domains.

The system relies on very little expert input other than the expert performing their skill and giving a general description of goal-directing quantities for the task performed. To improve the application of the framework, the goal-directed clones could be combined in a more symbolic hierarchy such as in van Lent and Laird (1999) or with task theories as in Bain and Sammut (1999).

The temporal dynamics of the system are handled by preset lags of system states (for learning) or control actions (for execution). Reaction times and anticipation times will vary with the nature of the task. Bratko et al. (1998) discusses the use of reaction time delay and suggests that it may not be crucial to clone learning. We found that there was an improvement in performance by using reaction time and anticipation time offsets. We have not explored the issue further, but have found that lag analysis of the data may provide some basis for choosing the offsets and goal-directing quantities. Learning temporal dynamics is a difficult problem that requires further research.

## Acknowledgements

This work was done as part of the first author's Doctoral Dissertation. We thank Mike Bain, Malcolm Ryan and James Westendorp for their advice and assistance with the flight simulator. Thanks also to Eduardo Morales for his valuable reviewing of the various drafts of this paper. The Australian Defence Science and Technology Organisation, Melbourne, was a great assistance to us in the early stages of this research.

## References

- Anderson, C., Draper, B., & Peterson, D. (2000). Behavioural cloning of student pilots with modular neural networks. *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 25–32). Stanford: Morgan Kaufmann.
- Bain, M., & Sammut, C. (1999). A framework for behavioural cloning. In K. Furukawa, D. Michie and S. Muggleton (Eds.), *Machine intelligence 15*, 103–129. Oxford University Press.
- Bratko, I., & Urbančič, T. (1999). Control skill, machine learning and hand-crafting in controller design. In K. Furukawa, D. Michie and S. Muggleton (Eds.), *Machine intelligence 15*, 131–153. Oxford University Press.
- Bratko, I., Urbančič, T., & Sammut, C. (1998). Behavioural cloning: phenomena, results and problems. *Proceedings of the Fifth International Federation of Automatic Control Symposium on Automated Systems Based on Human Skill* (pp. 143–149). Berlin: ASLIB Press, South Australia.
- Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. New York: Chapman and Hall.
- Camacho, R. C. (2000). *Inducing models of human control skills using machine learning algorithms*. Doctoral dissertation, Engineering Faculty of Porto University, Portugal.
- Franklin, G., Powell, J., & Emami-Naeini, A. (1995). *Feedback control of dynamic systems*. Reading, MA: Addison Wesley.
- Michie, D. (1993). Knowledge, learning and machine intelligence. In L. Sterling (Ed.), *Intelligent systems*, 1–19. Plenum Press.
- Quinlan, J. (1993). Combining instance-based and model-based learning. *Proceedings of the Tenth International Conference on Machine Learning*. (pp. 236–243). Amherst, Massachusetts: Morgan Kaufmann.
- Sammut, C., Hurst, S., Kedzier, D., & Michie, D. (1992). Learning to fly. *Proceedings of the Ninth International Conference on Machine Learning* (pp. 385–393). Aberdeen: Morgan Kaufmann.
- Stirling, D. (1995). *Compressed heuristic universal reaction planners*. Doctoral dissertation, Basser Department of Computer Science, University of Sydney.
- Thom, T. (1998). *The flying training manual*. Williamstown Victoria Australia: Aviation Theory Centre.
- van Lent, M., & Laird, J. (1999). Learning hierarchical performance knowledge by observation. *Proceedings of the Sixteenth International Conference on Machine Learning* (pp. 229–238). Morgan Kaufmann.
- Šuc, D. (2001). *Machine reconstruction of human control strategies*. Doctoral dissertation, Faculty of Computer and Information Sciences, University of Ljubljana, Slovenia.