Learning to Fly Simple and Robust

Dorian Šuc[1], Ivan Bratko[1], and Claude Sammut[2]

[1] Faculty of Computer and Information Science, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia
{dorian.suc,ivan.bratko}@fri.uni-lj.si

[2] School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
claude@cse.unsw.edu.au

Abstract. We report on new experiments with machine learning in the reconstruction of human sub-cognitive skill. The particular problem considered is to generate a clone of a human pilot performing a flying task on a simulated aircraft. The work presented here uses the human behaviour to create constraints for a search process that results in a controller – pilot's clone. Experiments in this paper indicate that this approach, called "indirect controllers", results in pilot clones that are, in comparison with those obtained with traditional "direct controllers", simpler, more robust and easier to understand. An important feature of indirect controllers in this paper is the use of qualitative constraints.

1 Introduction

Reconstructing sub-cognitive human skills from human experts' behavioural traces is one of the most fascinating applications of machine learning, also known as behavioural cloning. Learning to fly a fixed-wing aircraft in a flight simulator has become a benchmark for behavioural cloning [1, 2]. As well as providing complex control problems, flying also requires learning different kinds of manoeuvres for different flight stages and combining these manoeuvres to create a complete flight plan.

Early experiments in behavioural cloning adopted a "situation-action" approach in which control rules map the current state of the world directly into control actions [1, 2]. While these experiments were successful in constructing auto-pilots that could complete a flight, the situation-action approach had several problems. The control rules were not understandable nor were they very robust to noise and variations in the flight plan.

Subsequently, more "goal-directed" approaches have been employed [3, 4], giving greater readability and robustness. These methods first learn goal settings for different stages in a flight and then learn control rules to achieve those settings. The controllers are still directly derived from the human behaviour. In contrast, the work presented here uses the human behaviour to create constraints for a search process that results in the controller. We show in this paper that applying this approach, called "indirect controllers" [5], results in pilot clones that are comparatively much simpler, more robust and easier to understand than direct controllers. An important feature of indirect controllers in this paper is the use of qualitative constraints.

In the sequel we first outline the idea of indirect controllers that employ qualitative constraints. We then present the details of the flying task and experiments in extracting flying skills, and analyse the resulting clones with respect to their understandability and performance.

2 Indirect Controllers and Qualitative Strategies

2.1 Learning Direct and Indirect Controllers

The following is the usual procedure of applying Machine Learning to recover a control strategy from example execution traces. A continuous trace is sampled so that we have a sequence of pairs $(State_i, Action_i)$ ordered according to time. $State_i$ is the state of the system at time i, and $Action_i$ is the operator's action performed at time i. Then, usually, the sequence of these pairs is viewed as a *set* of examples, thereby ignoring the time order. The state is a vector of state variables: $State = (x_1, x_2, ...)$. A standard ML method is applied to induce the mapping from states to actions, whereby the state variables $x_1, x_2, ...$ correspond to attributes, and the actions are the class values. In continuous domains, both the state variables and the class are real-valued, therefore a numerical learning method, such as regression tree learning is appropriate for this. The result of learning, using this formulation of the learning problem, is a controller in the form of a function from system states to actions:

$$Action = f(State) = f((x_1, x_2, ...))$$

This controller maps the system's current state into an action directly, without any intermediate, auxiliary result. Therefore such controllers will be called *direct controllers*, to be distinguished from "indirect" controllers used in this paper.

We say that a controller is "indirect" if it does not compute the next action directly from the current system's state, but uses in addition to the state some additional information. Typical such additional information is a sub-goal to be attained before attaining the final goal. One idea to obtain such additional information, appropriate for handling dense sub-goals, is to generalise the operator's trajectory [5, 6]. Such a generalised trajectory can be viewed as defining a continuously changing sub-goal.

Subgoals or generalised trajectories are not sufficient to define a controller. A model of the system's dynamics is also required. Therefore, in addition to inducing subgoals or a generalised trajectory, this approach also requires the learning of approximate system's dynamics, that is a model of the controlled system. The next action is then computed "indirectly", targeting the sub-goals as follows: (1) compute the desired next state on the generalised trajectory (i.e. next sub-goal), and (2) determine an action that brings the system closer to the desired next state. This amounts to trying to follow a generalised trajectory as follows. The next action is determined indirectly as: using the model of the system's dynamics, and the generalised trajectory, find the action that will minimise the difference between the generalised trajectory and the state resulting from this action. So an indirect controller computes actions as:

$$Action = \arg\min_A(diff(model(State, A), Trajectory))$$

The point of indirect controllers is that the problem of behavioural cloning is decomposed into two learning problems: (1) learning trajectory, and (2) learning dynamics. It has been shown experimentally that this decomposition may result in much better performance than the induction of direct controllers [5, 6]. Related ideas of using subgoals in behavioural cloning in aeroplane flying were also discussed by Bain and Sammut [3].

2.2 Learning Qualitative Strategies

In earlier experiments [7, 6], we found that *qualitative* descriptions of generalised trajectories are particularly useful. We will refer to them as *qualitative generalised trajectories*. They are described in terms of monotonic qualitative constraints regularly used in the field of qualitative reasoning. For example, the constraint $Y = M^+(X)$ says that Y monotonically increases with X: if X increases then Y also increases. Analogously, $Y = M^-(X)$ says that Y monotonically decreases with X. These constraints can have multiple arguments. For example, $Z = M^{+,-}(X, Y)$ means that Z is monotonically increasing in X and decreasing in Y. If both X and Y increase then Z may increase, decrease, or stay unchanged.

Monotonicity constraints can be combined into if-then rules to express piecewise monotonic functional relationships. For example: $if\ X < 0\ then\ Y = M^-(X)\ else\ Y = M^+(X)$ Nested if-then expressions can be represented as trees, called *qualitative trees* [7, 8]. Qualitative trees are similar to regression trees [9]. Both regression and qualitative trees describe how a numerical variable (class variable) depends on other (possibly numerical) variables (attributes). The difference between the two types of trees only occurs in the leaves. A leaf of a regression tree tells how the class variable numerically depends on the attributes within the scope of the leaf. On the other hand, a leaf in a qualitative tree only specifies the relation between the class and the attributes qualitatively, in terms of monotonic qualitative constraints.

In this paper we applied the learning of indirect controllers to the problem of reconstructing pilots' skills when flying an aircraft. Generalised trajectories were stated in terms of qualitative trees. To induce qualitative generalised trajectories from examples of pilots' flights, we used program QUIN [7, 8]. QUIN (Qualitative Induction) is a learning program that induces qualitative trees from numerical data. QUIN detects monotonic qualitative constraints that hold "sufficiently well" in the data. Roughly, QUIN employs a criterion of "qualitative fit" between a qualitative tree and the learning examples. In the spirit of the MDL principle, QUIN searches for "small" qualitative trees that fit the learning data well. These mechanisms also make QUIN relatively robust with respect to noisy data.

3 Learning to Fly

The flight simulator used in the following experiments is of a Pilatus two-seat turboprop PC-9 aerobatic training aircraft. The dynamic model was provided to us by the Aeronautical and Maritime Research Laboratory of the Australian Defence Science and Technology Organisation. The model was derived from wind-tunnel testing and data collection from an actual aircraft.

3.1 Flying an Aircraft

The main controls for a fixed-wing aircraft are: the elevator that controls pitching the nose up or down, the ailerons that control rolling of the aircraft, the rudder that controls the yawing of the nose left or right, the throttle controlling the thrust, and the flaps that increase lift when they are extended. Any change in aircraft attitude can be expressed in terms of the motion about three aircraft axes:

- *pitch* is the motion about the lateral axis;
- *roll* is the motion about the longitudinal axis (bank);
- *yaw* is the motion about the normal axis.

When the aircraft is in approximately level flight, these correspond respectively to lift the nose up or down, banking the aircraft and changing compass heading. *Roll* is mainly controlled by the ailerons. *Pitch* is mainly controlled by the elevator. Yaw is controlled by rudder and is affected by $roll$.

In our experiments, the control variables were: $flaps$ and landing $gear$, *throttle* controlling the *airspeed*, stick x and y position ($stick_x$ and $stick_y$) controlling ailerons and elevators. As in previous experiments in learning to fly, we did not use the rudder.

Controlling the landing gear is very simple: raise the gear after take-off and lower the gear before landing. This rule is easily learned from any successful execution trace and will be omitted in the rest of the paper.

3.2 The Learning Task

The learning task is to do a standard left-hand circuit (see Figure 1) as described in the flying training manual [10]. The circuit consists of take-off, climb to a specified height, four left turns, descent to the runway and landing. The requirement to do some of the turns while climbing or descending makes the task more difficult. The success of the landing is score according to the descent rate, distance from the centre line of the runway and angle to the centre line. Since the visual field in the simulator is quite sparse and therefore provides few useful landmarks, we provided "windows in the sky" to help guide the pilot. These are squares whose centre marks the desired way point. When executing the task, pilots were supposed to fly through these windows. The side of each window is 200 feet (61m) in length[1]. This is intended to indicate a 100 ft tolerance when

[1] All units in the simulator are imperial since that is standard in commercial flight

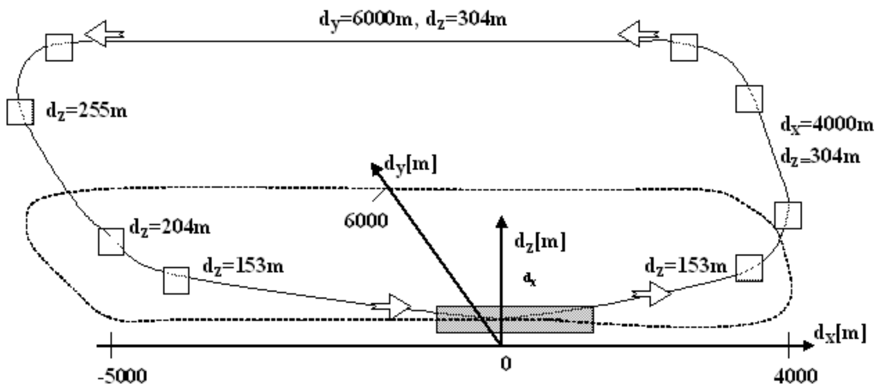

Fig. 1. The learning task: pilots were required to do the standard left-hand circuit, marked with 8 "windows". d_x and d_y denote the x and y distance from the runway and d_z denotes the altitude above the runway.

reaching a way point or goal. Each window (and the runway) defines the current goal for the pilot or the clone.

Pilots were considered to have flown successfully if they flew through all eight windows and land successfully. Successful landing requires low airspeed, near zero roll and pitch and landing gear down when the aircraft touches the runway.

4 Experiments

During a pilot's flight, the values of variables d_x, d_y, d_z, *roll*, *pitch*, *yaw*, *airspeed*, e_g, a_g, $stick_x$, $stick_y$, *throttle*, *flaps* were recorded. Here d_x, d_y and d_z stand for the distances in the three directions (in feet) from the starting position on the runway. The velocities of pitch, roll and yaw were not available. The attributes *goal elevation* e_g and *goal azimuth* a_g give the relative angles of the current goal window with respect to the current aircraft position. The goal elevation e_g is the difference between the current pitch of the goal window and the pitch to the aircraft. The goal azimuth a_g is the difference between the current direction of the goal window and the direction of the aircraft.

To make the clones independent of the runway/window positions we avoid using the variables d_x, d_y, d_z and *yaw*. However, controlling the airspeed of the aircraft requires some data about the aircraft position or the distance from the runway. For this reason, variables d_x, d_y, d_z were used while learning the control rule for airspeed, as will become clear later. Another representation that could be used is to convert absolute distances to distances relative to the target. This alternative is yet to be explored.

Flying an aircraft requires control rules for stick position ($stick_x$ and $stick_y$), *throttle* and *flaps*. These rules are learned from successful execution traces. In our experiments, we used traces from three pilots. We learned the clones as indirect controllers [5, 7]. That is, we separately learned the generalized trajectory and the system dynamics model. We used the qualitative induction system QUIN to learn qualitative constraints on the generalized trajectory.

When estimating the performance of the clones we consider the clone to have completed the task *completely* if it flies through all eight windows and lands successfully. We consider the clone to perform the task *partially*, if it flies through at least five windows, misses the other windows by less than 100 ft. and lands successfully. Considering the fact that the length of the circuit is 9,000 ft., missing a window by less than 100 ft. is still considered acceptable.

To induce an indirect controller, we have to learn from control traces two things: (a) a generalised trajectory, and (b) a model of the dynamics of the controlled system (the aircraft). First, we describe in the next section how the approximate system dynamics model is learned. Then we describe experiments in learning qualitative constraints on the generalized trajectory, i.e. learning the qualitative strategy. Finally, we transform the learned qualitative strategies into quantitative control rules that can be applied to controlling the aircraft.

5 Learning Dynamics

First we used locally weighted regression (LWR) [11] to learn the approximate system dynamics model. The prediction errors of the induced locally linear models were sufficiently small, so that these models of system's dynamics in combination with the generalized trajectory were easily sufficient to control the aircraft.

We also experimented with much simpler, global linear models of the system's dynamics. An example of such a global linear model of the system's dynamics, induced from three example flights by linear regression with $\Delta t = 1s$, is:

$$\begin{aligned}
r(t+\Delta) &= 0.994\, r(t) - 0.003\, v(t) + 2.773\, s_x(t) + 0.155 \\
p(t+\Delta) &= 0.958\, p(t) - 0.029\, v(t) + 0.775\, s_y(t) - 0.040 f(t) + 0.585 \qquad (1) \\
v(t+\Delta) &= 0.071\, p(t) + 0.964\, v(t) + 1.803\, t(t) - 0.022 f(t) + 2.109
\end{aligned}$$

Here r, p and v denote roll, pitch and airspeed, s_x and s_y the stick x and y position, f flaps and t throttle. The mean absolute errors of this linear model on a different flight trace are respectively (roll, pitch, airspeed) 0.429, 0.376, 0.378. Note that airspeed typically varies between 0 and 110 and roll and pitch are measured in degrees.

Although such a simple linear model is not perfect, we found that it is easily sufficient, in combination with a generalised trajectory, to control the system. Due to the simplicity and efficiency of the global linear model in comparison with locally linear models near the current state, we decided to use this linear model instead of LWR in the rest of our experiments. As found in our earlier work [5, 6], indirect controllers are in general not significantly affected by errors in the system's dynamics model as long as the model is qualitatively correct.

6 Learning Indirect Aircraft Controllers

To control the system with an indirect controller using a generalized trajectory, the generalized trajectory is combined with the learned dynamics model to constrain each of the control variables. An important criterion, when inducing the generalized trajectory, is comprehensibility. For example, if the generalized trajectory is induced through tree learning, the class values should be state variables that are intuitive to human thinking about controlling the particular dynamic system.

We decided to learn constraints on the aircraft's *roll*, *pitch*, *airspeed* and *flaps* with respect to the other variables. Constraints on these four variables will form the generalized trajectory. Such a trajectory, expressed in symbolic form, is usually easy to understand, as will be illustrated later. The constraints in the generalised trajectory determine the desired *roll*, *pitch*, *airspeed* and *flaps* for each aircraft position. Once the desired values of these four variables are known, it is possible to determine the control actions by the principle of indirect controllers using the model of the aircraft's dynamics.

In the remainder of this section, we describe experiments in learning generalized trajectories from example flights, and using these strategies in indirect controllers of the aircraft.

6.1 Learning Generalised Trajectories

In our case, a generalised trajectory consists of constraints on flaps, airspeed, roll and pitch. Here we describe the induction of these constraints from example execution traces. Constraints on each of the four "target" variables are expressed in terms of other state variables of the aircraft. We look at each target variable in turn, beginning with flaps.

Learning constraints on flaps. The flaps setting is a discrete variable with only three possible values, so it is appropriate to induce a decision tree that determines the flaps value from other state variables. An example of such a decision tree induced from one of the pilots' traces is:

$$\begin{array}{l} airspeed \leq 84.75 \\ \mid \quad airspeed \leq 62.35 : flaps = f_{land} \quad \{\text{flying slow, nose down}\} \\ \mid \quad airspeed > 62.35 : flaps = f_{normal} \quad \{\text{nose up}\} \\ airspeed > 84.75 : flaps = f_{off} \quad \{\text{flying fast, nose up}\} \end{array} \tag{2}$$

This rule can be interpreted as follows. Extending the flaps increases the surface of the wing and hence provides greater lift. The aircraft used in our experiments has the following three settings for the flaps: f_{land}=40, f_{normal}=20 and f_{off}=0. When flaps are extended (f_{land}), less speed is required to achieve a positive climb rate. In our control traces, experienced pilots usually used f_{normal} or f_{land} during take off, f_{off} when airspeed was high and f_{land} during the landing. Correspondingly the decision trees for flaps usually had only three leaves. Note that the induced decision tree above can be used for control directly, without a model of the system's dynamics.

Learning constraints on airspeed. In the traces we experimented with, controlling the airspeed is qualitatively quite simple: increase it to take-off, hold it constant during the flight and decrease it when landing. A good pilot uses the throttle to control the aircraft's climb and descent rates during a flight. Our control strategy is relatively naive in its lack of fine control of the throttle. However, it does closely emulate the actions of our pilots who tended to leave the throttle setting more or less constant. In the sequel we present performance results with rules obtained from regression tree learning, applying rather severe pruning of trees that resulted in trees with 6 – 10 leaves. These trees have small training error, are easy to understand and usually control the airspeed adequately. However, they have limited generality as they define functions of the form: $airspeed = f(d_x, d_y, d_z)$. The attributes d_x,d_y,d_z are relative to the runway position, so the induced trees may not be appropriate for other flight plans.

Learning qualitative constraints on roll and pitch. We found experimentally, that flaps and airspeed can be handled easily by decision or regression tree learning. But pitch and roll are not so easily susceptible to regression trees. Sometimes, successful regression trees were more complex (up to 20 nodes), harder to interpret and not as effective in control. So it is the pitch and roll where learning qualitative constraints really mattered.

A very elegant qualitative strategy was learned from some of the traces:

$$\begin{aligned} roll &= M^-(a_g) \\ pitch &= M^-(e_g) \end{aligned} \tag{3}$$

These two rules are quite intuitive and give a simple strategy to control pitch and roll by adjustment of the stick position. To explain this strategy, we first describe how roll and pitch affect the motion of the aircraft. Flying straight-and-level requires maintaining a constant heading and altitude. This can be achieved by keeping zero roll and near-zero pitch p_0. The exact pitch value p_0 that maintains the current aircraft height depends also on the airspeed and flaps. Roll and pitch angles are measured in the clockwise direction, whereas goal elevation and goal azimuth are measured in the standard anti-clockwise direction. To reach the goal with positive goal elevation and positive goal azimuth, the aircraft should climb and turn to the left. This is achieved by negative roll and negative ($< p_0$) pitch.

Now we explain how the qualitative strategy (rules 3) achieves the current goal. Consider straight-and-level flight, where the aircraft approaches the goal with positive goal azimuth and positive goal elevation. Since the aircraft's distance to the goal is decreasing, goal azimuth and goal elevation angles are increasing. The induced rules decrease roll and pitch, producing a left-turn and climb. That is exactly what is needed to achieve the goal. If the turn (climb) was too strong, goal azimuth (elevation) becomes negative and is decreasing (since the aircraft is approaching the goal). Now the induced rules increase roll (pitch) producing a right-turn (descent) to the goal. In this way, the induced rules tend to achieve zero goal azimuth and goal elevation, causing the aircraft to fly straight to the goal. The induced rules also command a tighter turn or steeper climb using a larger absolute roll or pitch, when the absolute goal azimuth or elevation is larger.

The qualitative strategy learned from a more experienced pilot is correspondingly more elaborate. Unlike a beginner, he also used flaps. This is reflected in the rule induced for pitch:

$$\begin{aligned} roll &= M^{-}(a_g) \\ pitch &= M^{-,+}(e_g, flaps) \end{aligned} \tag{4}$$

When controlling the pitch to obtain the goal elevation, the flaps setting is also considered. The rule states that if the flaps setting is higher ($f_{land} = 40$, flaps extended for the landing), the pitch should be larger (nose further down). This actually describes the property of the flaps: extended flaps requires the stick to be further forward to achieve the same goal elevation. If flaps are off ($f_{off} = 0$), the wings provide less lift, requiring the stick to be pulled back further (assuming constant thrust).

6.2 Transforming Qualitative Strategy into Quantitative Strategy

A qualitative control strategy only imposes qualitative constraints on "target" variables. These are directly useful for explanation, but they cannot be immediately applied for control. First, qualitative constraints have to be transformed into quantitative functions.

The induced qualitative constraints (rules 4) were transformed into quantitative functions as follows. Let *roll* denote a function $[-a_g^{mx}, a_g^{mx}] \mapsto [-roll^{mx}, roll^{mx}]$ and *pitch* denote a function $[-e_g^{mx}, e_g^{mx}] \times [f_{off}, f_{land}] \mapsto [-pitch^{mx}, pitch^{mx}]$ satisfying the induced qualitative constraints given by rule 4. The superscripts mx indicate the extreme values of the corresponding variables. In this transformation, we used simple domain knowledge to define additional constraints on *roll* and *pitch*:

- roll is 0 when goal azimuth is 0 ($roll(0) = 0$).
- pitch is 0 when goal elevation is 0 and flaps are normal ($pitch(0, f_{norm}) = 0$).
- flaps affect pitch for a small value f_{dif} ($|pitch(e_g, f_{land}) - pitch(e_g, f_{off})| = f_{dif}$, $0 \leq f_{dif} \leq 6deg$).
- maxima of corresponding variables used for $roll^{mx}$, $pitch^{mx}$, a_g^{mx} and e_g^{mx}.

Note that this domain knowledge is not perfect and is just a naive commonsense knowledge about flying an aircraft. For example, to maintain zero goal elevation with normal flaps, a near zero pitch p_0 is required. Its exact value depends also on the airspeed. Since the qualitative strategy is goal-directed, it is able to reduce the errors resulting from imperfect domain knowledge or imperfect model of system dynamics.

We used the procedure described in [7] to randomly generate the functions for *roll* and *pitch* so that the resulting randomly obtained functions satisfied the corresponding qualitative constraints stated above. We used the grid $C = 8$ and random $0 \leq f_{dif} \leq 6$. Out of 50 randomly generated functions, 27 (54 %) succeeded to fly completely or partially (the criteria of success as defined earlier). Figure 2 shows one such clone. As observed in our experiments in other

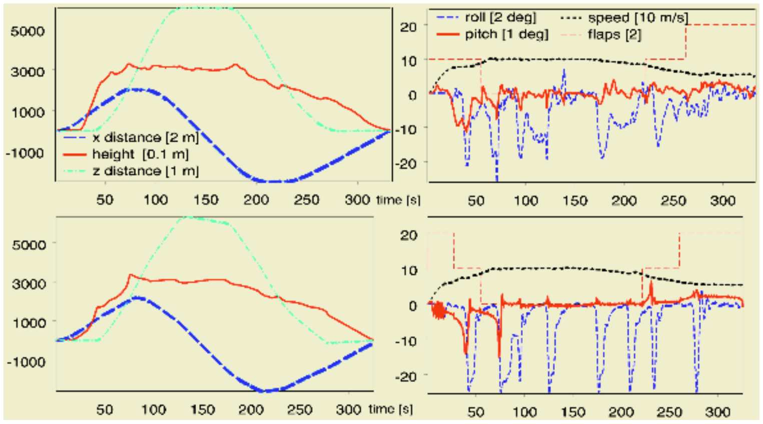

Fig. 2. Trace of the operator and trace of its clone. The upper two graphs show how the operator is flying. The lower graphs show a trace of a clone that controls roll and pitch according to the induced qualitative strategy (rule 4), and speed and flaps according to the induced trees. Eight negative local maxima of roll correspond to the eight windows where left-turn is required. Negative local maxima of pitch correspond to climbs (the first and second window) and positive local maxima to descent.

domains ([7, 6], for example) success of the clones could be improved by using more domain knowledge. In the flight domain, such additional domain knowledge could be the requirement that roll and pitch are small for small goal azimuth and goal elevation.

Note that we here used randomly generated functions satisfying induced qualitative constraints and simple domain knowledge. An alternative, described in [12], would be to use control traces to find a quantitative strategy that fits the control traces numerically and is consistent with the qualitative constraints.

6.3 Robustness of the Qualitative Strategy

To evaluate the robustness of the clones we performed two sets of experiments in which we tested:

- *the clone's robustness against the turbulence:* clones were tested in the simulator with turbulence added.
- *the clone's ability to complete different missions:* clones were tested with a modified flight plan that includes a right climbing turn, which is a maneuver not seen in the execution traces. The task also requires a sharper left-turn than any in the original flight plan.

We tested ten different clones that scored complete success on the original task. They control roll and pitch according to the induced qualitative strategy (rule 4) and speed and flaps according to the induced trees. Note that in this experiment, clones learned from traces of the original flight plan with no turbulence were required to deal with turbulence and a modified flight plan.

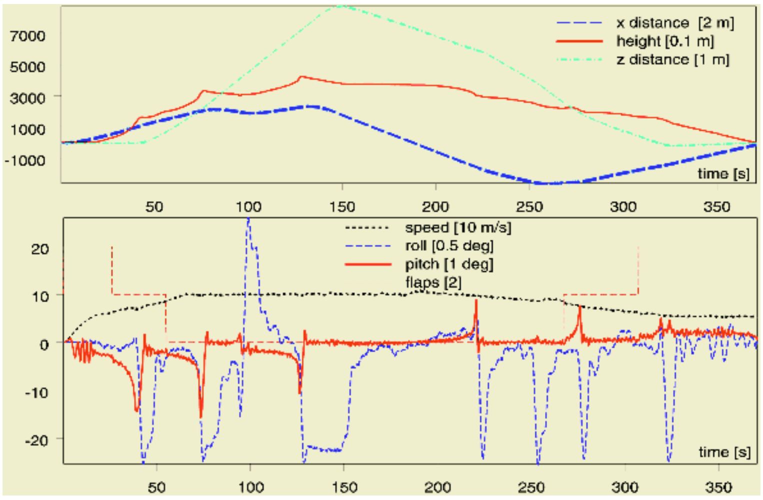

Fig. 3. Trace of a clone that does the modified flight task with strong turbulence. Positive maxima of roll corresponds to right-turn after third window. Oscillations in roll and pitch correspond to clone's adjustments to the turbulence. Note that this is the same clone as clone in Figure 2.

The results indicate good robustness of our clones in both respects. With turbulence, all the clones completed the task entirely or partially. Eight of ten clones also completed the modified plan entirely or partially. We obtained similar results, when the clones were tested with both modified flight plan and turbulence. Figure 3 shows a flight by a clone that carries out the modified task under turbulence.

6.4 Flying Simply

Here we give an example of a very simple controller that is consistent with the induced qualitative rules $roll = M^{-,+}(a_g, flaps)$ and $pitch = M^{-}(e_g)$. It uses linear quantitative rules:

$$\begin{aligned} roll &= -1.8\, a_g \\ pitch &= -1.0\, e_g + 0.02\,(flaps - f_{normal}) \end{aligned} \tag{5}$$

To compute the actions s_x and s_y aiming to achieve the desired roll and pitch linear dynamics (Eq. 1) is used. Flaps are controlled by the rule 2 and the airspeed by the induced regression tree with six leaves. Such a controller completes the original task, can perform the modified plan (and also other tasks) and is robust to turbulence. Note that the controller is also robust with respect to (reasonable) changes in coefficients in equations 5. For example, rules $roll = -3\, a_g$ and $pitch = -3.0\, e_g + 0.03\,(flaps - f_{normal})$ also do well.

7 Conclusions

In this paper we experimented in the well-known domain of cloning air pilots' skills. We applied the approach of inducing from pilots' performance data *indirect controllers* using qualitative representations of the generalised trajectories. In comparison with the more traditional learning of direct controllers in this domain on similar flying tasks, our results show that the indirect controller approach leads to simpler and more robust clones.

Two limitations of the work described in this paper that belong to future work are: (1) Human expert intervention was required in determining "target" variables featuring in constraints defining the generalised trajectory; it is a challenging problem to determine these variables automatically; (2) our induced controllers are rather task-dependent, in particular the rule for *airspeed* corresponds closely to the flight plan of the example flying task; in future experiments this should be generalised to an arbitrary specification of a flight plan.

Acknowledgements

The work reported in this paper was partially supported by the Slovenian Ministry of Education, Science and Sport. We thank the Aeronautical and Maritime Research Laboratory for providing the PC-9 model.

References

1. Sammut, C., Hurst, S., Kedzier, D., Michie, D.: Learning to fly. In: *Proc. 9th International Workshop on Machine Learning*, Morgan Kaufmann (1992) 385–393
2. Camacho, R.: Using machine learning to extract models of human control skill. In: *Proceedings of AIT'95. (1995)* Brno, Czech Republic
3. Bain, M., Sammut, C.: A framework for behavioural cloning. In: *Machine Intelligence 15*, Oxford University Press (1999) 103–129
4. Isaac, A., Sammut, C.: Goal-directed learning to fly. In: *Machine Learning, Proc. 20th International Conference (ICML 2003)*, AAAI Press (2003) 258–265
5. Šuc, D., Bratko, I.: Problem decomposition for behavioural cloning. In: *Proc. of the 11th European Conference on Machine Learning*, Springer (2000) 382–391
6. Šuc, D., Bratko, I.: Symbolic and qualitative reconstruction of control skill. *Electronic Transactions on Artificial Intelligence, Section B* **3** (1999) 1–22
7. Šuc, D.: *Machine Reconstruction of Human Control Strategies.* Frontiers in Artificial Intelligence and Applications, volume 99. IOS Press, Amsterdam, (2003)
8. Šuc, D., Bratko, I.: Induction of qualitative trees. In: *Proc. of the 12th European Conference on Machine Learning*, Springer (2001) 442–453
9. Breiman, L., Friedman, J., Olshen, R., Stone, C.: *Classification and Regression Trees.* Wadsworth, Belmont, California (1984)
10. Thom, T.: *The Air Pilot's Manual. 7 edn.* Air Pilot Publishing Ltd (2003)
11. Atkeson, C., Moore, A., Schaal, S.: Locally weighted learning. *Artificial Intelligence Review* **11** (1997) 11–73
12. Šuc, D., Vladušič, D., Bratko, I.: Qualitatively faithful quantitative prediction. *Artificial Intelligence* (2004), Accepted for publication.