Determination of Planetary Meteorology from Aerobot Flight Sensors

Phil Summers, Dave Barnes, Andy Shaw

Department of Computer Science
University of Wales, Aberystwyth
Aberystwyth
Ceredigion
Wales
SY23 3DB
UNITED KINGDOM
prs94@aber.ac.uk
dpb@aber.ac.uk
ajs@aber.ac.uk

ABSTRACT

Airborne robots, or aerobots, are fast becoming potential experiment delivery and planetary analysis vehicles. With the current trend towards a *faster, better, cheaper* methodology, aerobots seem to offer significant advantages over rovers and other methods of planetary exploration. This paper describes work performed in the Department of Computer Science at the University of Wales, Aberystwyth, as part of a PhD project, to examine potential methods that could be used to make an aerobot mission a realistic prospect.

An essential part of any aerobot mission will be environmental analysis using a number of specialised sensors. These sensors are potentially made up of accelerometers and gyroscopes together with a wind sensor such as an anemometer and possibly temperature sensors. Such sensors will be used to allow the aerobot to quantify the weather patterns of the target planet and to determine the best course of action to achieve its goals. For a planet such as Mars, we have a relatively good knowledge of the global climate, but for an aerobot mission to succeed we must be able to analyse and determine the local climatic conditions around the aerobots' current position.

In keeping with the *faster, better, cheaper* methodology it is important to minimise the number of sensors carried by an aerobot. It therefore seems reasonable to provide sensors that can play a dual role in order to maximise the use of the gathered data. That is, if environmental data could be extracted from sensors primarily used for flight control, we would be further minimising cost whilst not impacting the level of scientific analysis performed.

The main focus of the work at Aberystwyth has been to investigate a potential method for retrieving environmental data from sensors that are primarily used to control flight. Using the FlightGear open source flight simulator together with terrain produced from MOLA data and realistic Martian weather patterns extracted from a Navier Stokes simulation, we have been able to produce a model of Mars that affords the ability to simulate an aerobot on a planetary exploration mission. In order for the aerobot to survive on Mars it must be able to react quickly to situations without operator input due to the long communications delay. Therefore any mission information sent to the aerobot from Earth must be at an abstract level. To simulate this we have developed an autopilot that is functionally similar to that in a conventional aircraft and allows the aerobot to carry out these abstract instructions. The autopilot was built upon a fuzzy rule base that was trained with high-level flight operations. After numerous simulated aerobot flights, rule activation surfaces were produced. By examining these activation surfaces coupled with aerobot state data we are able to infer the external meteorological perturbations experienced by the aerobot. An analogy can be made with a conventional passenger jet flying through turbulence. The pilot of such a jet is providing input to the control surfaces of the aircraft to counteract turbulence. If the control input is examined together with positional information, we can infer the external perturbations the aircraft is experiencing and hence determine the external meteorology.

INTRODUCTION

The success of any mission to another planet can be measured by the amount of useful scientific data gathered by the probe or lander. This measure can be applied whether the mission consists of an orbiter, lander or some other form of exploration vehicle. Airborne robots (aerobots) are no different in this respect, however they do open up a whole new set of problems. An aerobot is at far more risk from the prevailing weather conditions due to the fact that it is an

airborne vehicle. With limited power reserves and potentially very high-velocity winds an aerobot must intelligently adapt its operations in order to carry out its mission objectives. These objectives would generally consist of the delivery of experimental packages to selected sites on the target planet and possibly the return to said sites at a later time to collect the results of the experiments [1].

Due to the very nature of an aerobot, and its sensitivity to external perturbations, it must know something about the weather it is currently experiencing. This may be to judge how to get to the next waypoint in the mission or to take evasive action to avoid an aggressive weather pattern.

The control system of an aerobot would consist of a number of instruments to allow the measurement of external perturbations; inertial sensors play an important role in keeping the aerobot stable and allowing a response to be made to these outside influences. However inertial sensors alone would not give any real useful scientific information; the data must be processed using complex algorithms and over time useful information could then be extracted [2].

The system under development at Aberystwyth uses rule activation data from a number of fuzzy autopilots to produce an increasingly accurate estimation of the current weather experienced by the aerobot. Each autopilot has a characteristic rule-firing pattern associated with a specific situation. Deviations from this known firing pattern allow the inference of external effects that may be causing the deviation. Furthermore, by performing certain flight operations, such as changing the direction of flight, it is possible to cross-validate inferences produced from the rule firing of one autopilot with that of another.

Whilst the majority of this work has been carried out in simulation, due to the problems associated with repeatability when using real aerobots in aggressive weather and environmental conditions, we believe our work will port well to real aerobots.

THE SIMULATION ENVIRONMENT

The simulation environment used to perform this work was based around the FlightGear open source flight simulator [3,4]. The FlightGear simulator has been in development for a number of years and has recently become fairly mature. The advantage of using FlightGear over more popular flight simulators, such as Microsoft Flight Simulator, stems mainly from its open-source licence. This licence ensures that the source-code for the simulator can be obtained which allows extensive modifications to be performed. For example, it is possible to add a new flight model, generate custom terrain, add realistic weather patterns and create new autopilot modules. FlightGear also supports the use of multiple computers, connected over a UDP socket, in order to synchronise displays. In this way a full wrap-around view could be created if the user so desired. We recently extended FlightGear to support both a stereo display using a Stereographics CrystalEyes encoder coupled with 3D glasses and a head-tracking system using a Polhemus Isotrack II device for adjusting the user's viewpoint based upon their head movement.

Fig 1 - FlightGear simulator with Martian terrain and multi-screen capability

Custom Terrain

In order to fly through a realistic environment a set of terrain was produced for the simulator that matched the surface of Mars as closely as possible. Publicly available data, gathered by the MOLA instrument [5], was used to create a terrain altitude map for the whole surface of Mars. This altitude map was then interpolated to 30 arc seconds and a digital elevation model (DEM) for each one-degree by one-degree section of Mars produced. The diameter of Mars is almost half that of the Earth and due to the fact that FlightGear is concerned mainly with simulation of Earth-bound flight its planetary parameters are set accordingly. These parameters were modified to reflect Mars, along with a number of other parameters such as the orbital eccentricity of the planet and gravitational constants. The DEM's were then fed into the scenery creation pipeline of FlightGear, a separate piece of software entitled TerraGear, and after a considerable amount of processing, (approximately four days run-time on a twenty processor Sun Enterprise machine), FlightGear terrain for the whole surface of Mars was produced. Fig 1 shows a screenshot of the Martian terrain within FlightGear (left), along with a photograph depicting the multi-computer capability projected onto a curved screen (right).

Weather Patterns

Realistic Martian weather patterns were integrated into the simulator by the use of a 3D Navier Stokes computational fluid dynamics (CFD) system [6]. The terrain produced from the MOLA data, described above, was imported into the CFD system. The system was then initialised with data from the Mars Climate Database [7] for the area under investigation and a 4 dimensional wind model was produced. This wind model was imported into the flight simulator in the form of a look-up table to provide the external perturbations necessary to carry out this investigation.

THE FUZZY AUTOPILOTS

To date, two distinct types of autopilot have been created and trained. These are:

- Altitude-hold
- Heading-hold

The autopilots were generated from expert flight data using fuzzy system identification software. A toolbox was obtained for Matlab [8] that could perform this system identification and generate a fuzzy model to describe the response of the system to stimuli. The toolbox was written by Robert Babuska and is freely available on the Internet.

The fuzzy models generated were Takagi-Sugeno (TS) models and contained rules of the form:

$$\text{If } \textit{Altitude Error} \text{ is } \textit{Antecedent}_{1,1} \text{ and } \textit{Vertical Speed} \text{ is } \textit{Antecedent}_{1,2} \text{ then}$$
$$\textit{Pitch Trim} = CP_{AE}\,\textit{AltitudeError} + CP_{VS}\,\textit{VerticalSpeed} + CP_{O} \quad (1)$$

Where CP_{AE} CP_{VS} and CP_{O} denote the consequent parameters for the altitude error, vertical speed and an offset respectively.

The general process used for producing each autopilot was as follows:

1. Perform manual flights to gather training data.
2. Perform manual flights to gather validation data.
3. Generate a fuzzy model in Matlab from the training data.
4. Validate the fuzzy model using the validation data.
5. Convert the fuzzy rules into a form that could be used by the fuzzy autopilot module constructed for the FlightGear simulator.

Due to the number of tuneable parameters within the fuzzy system identification toolbox, many fuzzy models were produced for each autopilot. These models were then evaluated within the flight simulator and the best models chosen for further investigation. Parameters within the fuzzy identification toolbox that were modified were:

- Number of clusters (i.e. rules).
- Type of antecedent (product space or projected space).
- Fuzziness.

Initially approximately 1800 models were produced for each autopilot. From these 1800 the 100 best performing models were chosen. A generic fuzzy autopilot module was constructed for the FlightGear simulator that could be configured to use any set of fuzzy rules and could process any number of model inputs and outputs. This autopilot module was written in C++ and multiple instances of the autopilot can be instantiated within the simulator to allow many autopilots to run in conjunction with one another.

A diagram showing the architecture of the autopilots along with their integration into the FlightGear simulator is presented in Fig 2.

Altitude-Hold Autopilot

The altitude-hold autopilot was designed to allow an aerobot to climb or descend to a user-defined altitude and then hold position at that altitude regardless of external influences. The aerobot used for the experiments was a heavier-than-air, powered glider. As such, the aerobot had a number of control surfaces that could be used to influence the flight attitude. The autopilot could easily be modified to control a lighter-than-air aerobot such as a helium filled balloon.The inputs to the autopilot are:

- Altitude error
- Vertical speed

The output of the autopilot is the position of the pitch-trimmer required to reduce the altitude error and move toward the target height. It is important to note that the vertical speed of the aerobot is essential for producing an accurate model. If the vertical speed were not included as an input the autopilot would tend towards pitching the nose too high and thus entering a stall and losing all lift.

Heading-Hold Autopilot

The heading-hold autopilot is similar to the altitude hold autopilot. This autopilot allows the aircraft to turn to a user-defined heading and to hold its position at that heading. The inputs to this autopilot are:

- Heading error (-180 degrees to +180 degrees)
- Roll angle

The outputs are the positions of both the aileron and rudder control surfaces, the aircraft requires the rudder information to enable co-ordinated turns to be performed. The roll angle input serves a similar purpose to the vertical speed input in the altitude-hold autopilot.

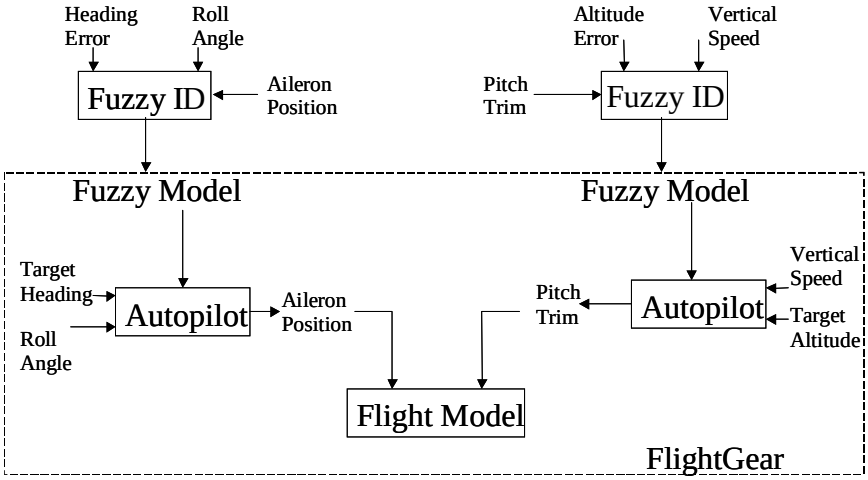

Fig 2 – The Autopilot/Simulator Architecture

EXPERIMENTS AND RESULTS

A number of experiments have been performed using both an individual autopilot and coupled altitude-hold and heading-hold autopilots, each with differing wind-speeds. Initially the winds from the CFD simulation were not used and a simple single direction wind of constant velocity was generated within the simulator. The first operation was to evaluate the altitude-hold autopilot.

Altitude-Hold Experiments

In order to test the altitude-hold autopilot models, an aerobot was placed in the air at an altitude of four hundred feet. The target altitude for the autopilot was one thousand feet. Each autopilot model was tested for exactly two minutes using the same initial conditions. Flight model data was gathered along with the autopilot rule-firing data. For each iteration of the autopilot (approximately 25Hz) every rule fires to some extent, be it 0% or 100%. This data was then used to plot graphs of each rule's contribution to the autopilot output at every time step - Fig 3.

Further experiments with the altitude-hold autopilot consisted of similar flights as above but with the introduction of a headwind at a constant wind-speed. The wind-speed was varied between 5m/s and 15m/s for each model iteration. As can be seen in Fig 4, the greater the wind-speed the shorter the time taken to reach the target altitude. This can be explained by the headwind causing an increase in the lift available to the aerobot and thus allowing a faster transition to the target altitude. Furthermore, comparison of the rule-activity graphs (Figs 3,5,6) for the three different wind-speeds shows similar rule firing patterns for each, but with a shift to the left of the graph, i.e. the rules fire for a shorter time as the altitude increases at a faster rate.

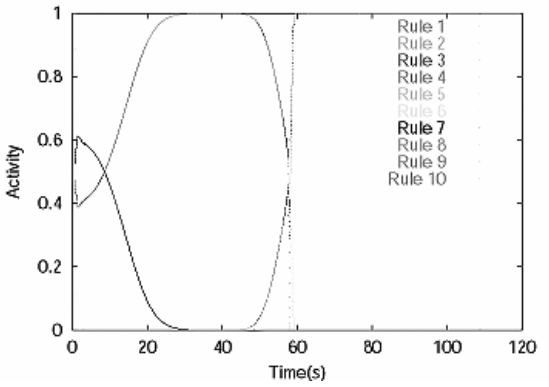

Fig 3 - Rule activity for 5m/s wind

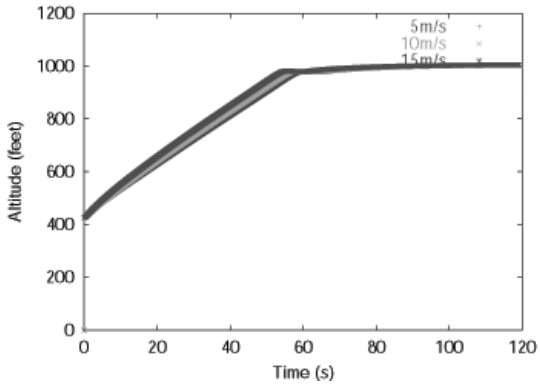

Fig 4 - Altitude versus time for increasing wind-speeds

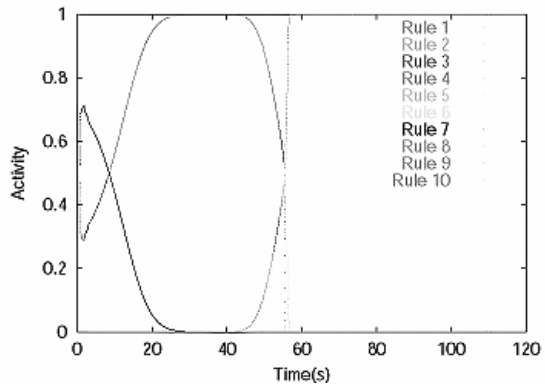

Fig 5 - Rule Activity for 10m/s wind

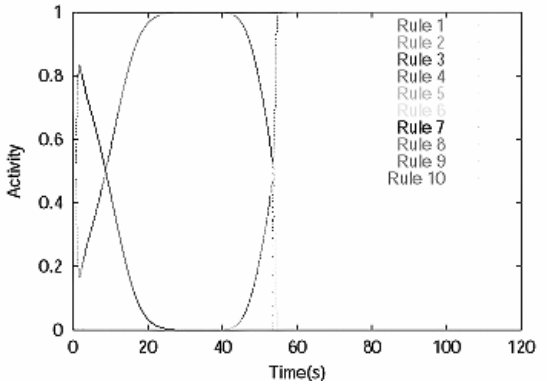

Fig 6 - Rule activity for 15m/s wind

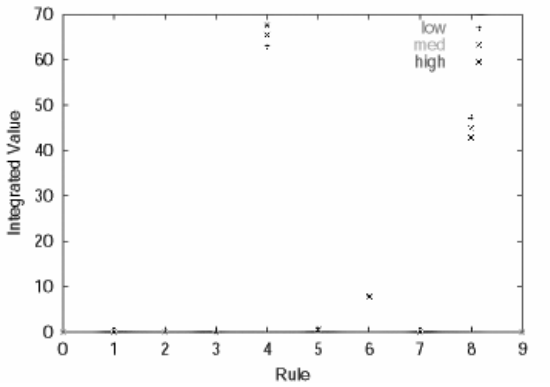

Fig 7 - Integrated rule activity for low/med/high winds

Euler integration was performed for each of the rules in the activity data to generate a metric that affords easy comparison of activity at different wind-speeds. Plotting these values shows a trend for each rule within the autopilot as shown in Fig 7, and by finding the equation of the trend for each rule we can estimate the head-wind component experienced by an aerobot when the external perturbations are unknown. Such an estimate was made for a wind-speed of 45m/s, the flight was performed, and the results were found to have an accuracy of within 0.5%.

An interesting point to note is that with a very high wind-speed of 120m/s the autopilot was still able to reach the desired altitude, albeit with the aircraft travelling backwards!

Heading-Hold Experiments

Initial results from the heading-hold experiments were extremely good. The autopilot had the ability to turn to the correct heading smoothly, using an acceptable bank angle and then to hold at the desired heading indefinitely, even with the introduction of wind. The graph in Fig 8 shows the roll angle and heading error of the aircraft as it banks to turn through ninety degrees. It can be seen from the graph that the roll angle gradually increases over the first 5 seconds or so, then stabilises until the roll out point is reached. The roll out point is approximately equal to the roll angle (which in this case is 20 degrees), that is, as the heading error reaches 20 degrees the aircraft begins to roll out of the turn. The autopilot performs this task well and the heading, on completion of the roll out, is correct to within a hundredth of a degree and very stable.

The graph in Fig 9 shows the rule activity for the flight depicted in Fig 8. Many more rules are active within the heading-hold autopilot as compared to the altitude-hold autopilot, although not all models have been examined to date. The rule activity does seem to follow the trend of having a characteristic firing pattern regardless of the amount of wind applied. The heading-hold autopilot models are currently being investigated in order to select the most accurate set for further investigation (*c.f.* altitude hold experiments).

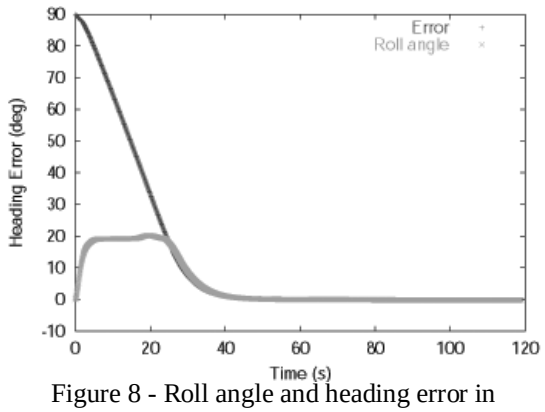

Figure 8 - Roll angle and heading error in heading-hold experiment

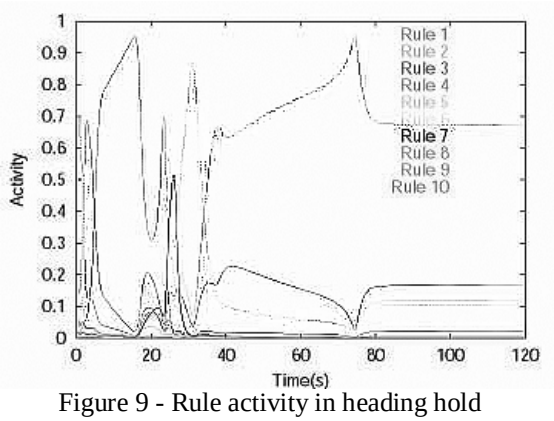

Figure 9 - Rule activity in heading hold experiment

CONCLUSIONS

From the results presented within this paper we can conclude that there is indeed a correlation between the activity of rules within a fuzzy autopilot and the external perturbations present within a flight environment. These perturbations influence the rule activity by a predictable amount. Given this, along with the fact that for a given model the rule activity follows a similar pattern regardless of the external influences, we have shown that it is possible to predict unknown external perturbations by extrapolating known trends in rule activity. The altitude-hold autopilot results confirm that this is correct.

Further work is required with the heading-hold autopilot, and as yet, the complete set of fuzzy models have not been examined or evaluated. The results presented in this paper for this autopilot were produced by selecting one of the models at random. Given this, the law of averages suggests that a better model resides somewhere within the data set and thus better results may be obtained.

The combination of two autopilots running at the same time will allow cross-checking of any predicted external perturbations. By running both the altitude-hold and the heading-hold autopilots and by, for example, turning the aerobot through ninety degrees, the predicted headwind can be measured as a crosswind by the heading-hold autopilot, and the predicted crosswind can be measured as a headwind by the altitude-hold autopilot. Using this method it will be possible to increase the accuracy of the prediction of external perturbations.

REFERENCES

[1] Barnes, D., Summers, P., Shaw A., "An investigation into aerobot technologies for planetary exploration," *ASTRA Workshop*, 2000, ESA WPP-179.

[2] Cancro, J., Tolson, H., Keating, G., "Operational data reduction procedure for determining density and vertical structure of the Martian upper atmosphere from Mars Global Surveyor accelerometer measurements," *Technical Report*, NASA Langley Research Centre, 1998.

[3] Olson, C., "The FlightGear Project," *Website: http://www.flightgear.org*

[4] Findlay, D., "The FlightGear flight simulator project," *Proceedings of Linux.conf.au*, 2002

[5] Zuber, M. T., *et al*, "Mars observer laser altimeter investigation", *Journal of Geophysical Research, 97, 1992*

[6] Koster, F., Griebel, M., "Development of CFD Code NaSt3D GP," *Website: http://wissrech.uni-bonn.de/research/projects/koster/koster3.html*

[7] Lewis, S. R., *et al*, "A climate database for Mars," *Journal of Geophysical Research – Planets, 97*, 1992

[8] Babuska, R., "Fuzzy modelling and identification toolbox for use with Matlab," *Website: http://lcewww.et.tudelft.nl/~babuska/klbook.htm*