Formation Flight
CS 229 Project: Final Report

Zouhair Mahboubi
Tao Wang

December 11th, 2009
Stanford University

Abstract

This paper is submitted as the requirement for the final project report for the CS229 project. However, it is well over the suggested length because it is also intended as a comprehensive progress report of the Formation Flight project being carried out in the AI Lab. A great deal of effort has been put in getting two airplanes to fly autonomously, and our goal is to document our work so that similar attempts to use UAVs in the future might be able to leverage it and learn from our experience and mistakes.

Acknowledgments: We would like to acknowledge the invaluable help and advising of both Zico Kolter and Geoff Bower in this project. They are very helpful with problem solving, decision making and fun having during the flight testing. We would also like to thank our pilot Garett, whose skills have avoided some imminent crashes in the early stages of the flight-testing.

1 Introduction

The goal of our project is to demonstrate autonomous formation flight using unmanned air vehicles (UAVs). The project is inspired by migrating birds which fly in v-shaped formations in order to decrease the induced drag. By flying in the wake of a leading bird, they are able to either save energy or increase range.

Researchers have a relatively good understanding of the phenomenon, but apart from short flight demonstrations by F-18 carried out at NASA Dryden Flight Research Center, the concept is still in its infancy. However, flight-test results and models of the wake vortices show that it's possible to save as much as 20% in fuel consumption. If commercial airplanes took advantage of this, we could see a significant lowering of the fuel consumption by aviation, which would translate into a positive environmental impact by lowering CO_2 and NO_x emissions. This is also attractive for airlines since it represents a sizable cost saving (especially in the 'wake' of a potential carbon tax) without requiring them to change their current fleet.

2 Hardware and Software Architecture

In this section we describe the hardware and software that we are using to accomplish this task

2.1 Hardware

The airplanes we are flying are two identical remote-controlled 'ready-to-fly' trainers (Alpha60 model). The following table summarizes their properties:

Weight	3.8 kg
Span	1.78 m
Wing Area	0.6 m^2
Aspect Ratio	5.3
V_{cruise}	14 m/s

The airplanes were modified so that they can be flown autonomously. We decided to use the paparazzi [1] autopilot to accelerate the development process. Thus each airplane is equipped with a microprocessor, GPS antenna for positioning, infrared thermopiles and roll gyro for attitude, 900 Mhz radio for telemetry and a 2.4 Ghz receiver as a safety link. The trailing airplane also has a current sensor for power consumption and an airspeed sensor.

[1] PaparazzI The free Autopilot http://paparazzi.enac.fr/wiki/Main_Page

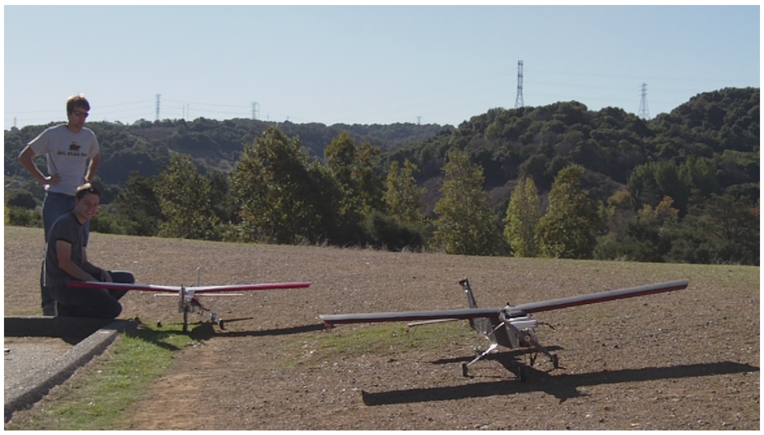


Figure 1: Airplanes in Cupertino Foothills

In order to overcome the inaccuracies of GPS positioning, we are relying on vision to track more precisely the position of the trailing plane relative to the lead plane. This is a reasonable approach and has been proposed for autonomous aerial refueling by the military. For this reason, the lead airplane (Batman) has 4 high-power LEDs that a camera on the trailing airplane (Joker) captures. A small computer runs a vision algorithm which is supposed to infer the orientation and relative position of the two airplanes. It uses the I2C protocol to communicate with the autopilot flight-code. A more detailed description of the hardware setup is provided in appendix A.1.

2.2 Software

One advantage of using the paparazzi system is that it is open source software, allowing us to easily modify the behavior of the UAV. Both Batman and Joker run essentially the same code, the only difference between the two is an XML setting file which determines the appropriate gain and flight plans of each.

In order to be able to properly implement our own control laws, an in-depth analysis of the software architecture was necessary to understand all the details. Figure 2 summarizes the software architecture. The processing and attitude sensing is done at 60Hz, while the GPS refresh rate is only 4Hz. The ground station (GCS) allows to uplink mission-level commands as well as change gains in-flight.

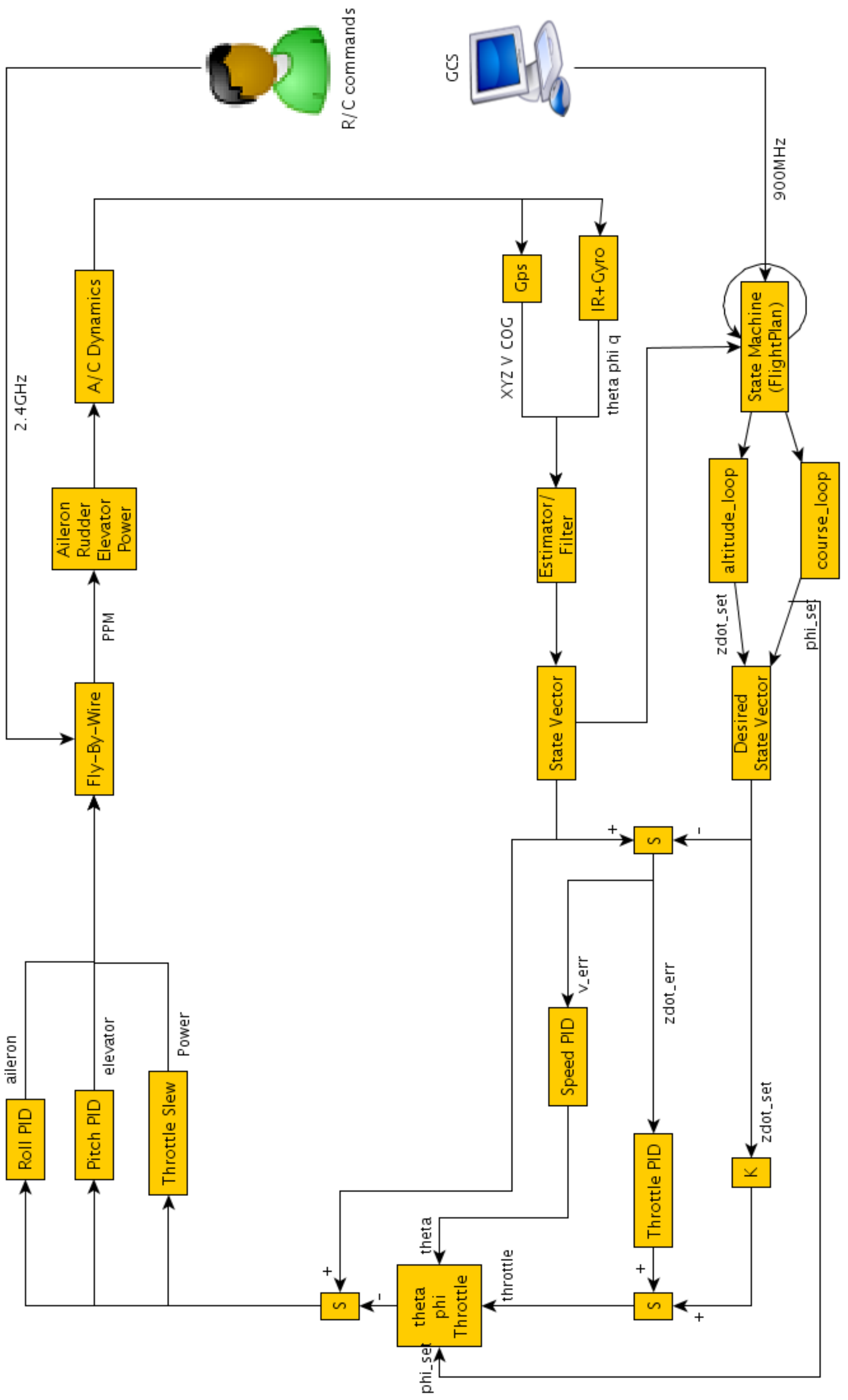


Figure 2: Software Architecture

3 Aerodynamic Simulation

While the paparazzi system comes with simulation capability which is useful for testing flight-plans, it is not realistic enough to be used for control-law testing or gain tuning (per example, it is possible to achieve positive climb rate with 0% throttle in the simulation).

For this reason one of our goals was to integrate the system with a more accurate simulation. Initially we planned to re-write the control law in Simulink and use the simulator in Matlab's aerospace toolbox. However, in order to keep things rather 'lean', we decided to use JSBSim instead. JSBSim is a non-linear 6DOF dynamic simulator aimed for aerodynamic simulations written in C++. This means that it was possible to have a simulation environment which runs the exact same code as the one that the micro-controller runs, thus reducing the workload as well as the risks of making mistakes when porting Simulink control-laws back to C.

JSBSim relies on the concept of force and moment build up using stability derivatives. These are function of the aerodynamics coefficients which mainly depend on the geometry of the airplane. So in order to obtain these values, we measured the airplane and used AVL [1] a vortex-lattice code meant for aerodynamic analysis. However, it should be noted that this is a non-viscous and linear method, which is only approximate. It usually does a good job, but it does not give exact numbers and definitely does not have any of the non-linear aerodynamic effects.

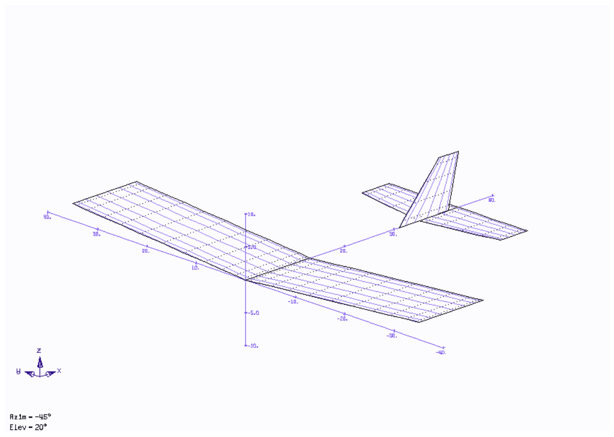


Figure 3: AVL Geometry for Stability Derivatives Extraction

4 Flight testing and results

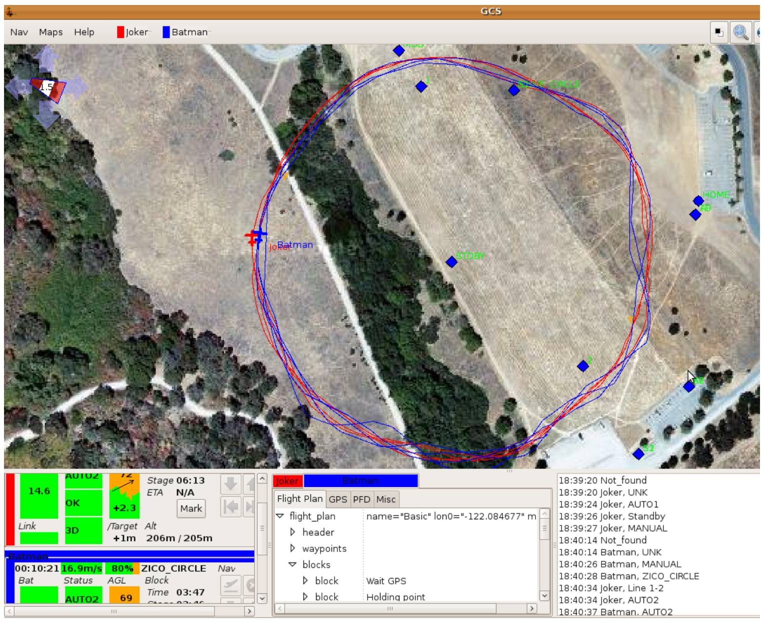
Flight-testing is done near the Rancho San Antonio Park in Cupertino, CA. For simplicity, a human-pilot takes care of taking-off and landing the planes. Once the lead plane is airborne and in autonomous mode, the second one takes-off and is put in autonomous mode as well. Depending on the plane, we get between 15 and 20 minutes of flight time, which is relatively short but with spare batteries we have enough time to carry out the test-flight plans.

So far we have been successful in getting both airplanes to fly autonomously at the same time. We repeated this on more than one occasion and have been able to resolve some of the major issues we had with the system (interference between engine and transmitter, board resets, unreliable vision system, etc.)

We have implemented two new algorithms for both longitudinal and lateral control. The first one is an implementation of total-energy control [2] which allows us to control speed and altitude simultaneously using a Multiple-Input-Multiple-Output (MIMO) control strategy. We used the 6DOF simulation to tune the parameters, but fine-tuning during flight tests was still necessary. For lateral control, we have closed the loop on the bank-angle in an attempt to achieve repeatable circles. By controlling the set groundspeed, we got each aircraft to catch-up to a virtual 'carrot' flying the same circle at a fixed velocity. Syncing the virtual carrot for each aircraft allows us to close the gap between the two aircrafts.

Unfortunately, our altitude hold and circle following has been less than satisfactory. Although we are able to hold groundspeed to within 1m/s, altitude error can be as much as 7m with +/-5m being common. As for the circle following, each airplane seems to do well over half of the circle (see figure 4), but has difficulties on the opposite side. So far we have noticed that this is very strongly correlated to the position of the airplanes relative to some of the hills near our flight site. Our hypothesis is that because we rely on IR sensors (i.e. horizon sensing) to determine attitude, the hills are a form of disturbance. Indeed, the lateral errors occur when the airplane is flying parallel to the hills (i.e. disturbance in lateral sensing) while the altitude hold is at its worse when the airplane is flying towards the hills (i.e. disturbance in the longitudinal sensing).

It's worth mentioning that we have been using GPS altitude and climb rate for altitude hold: this is known to cause problems given the noise in GPS measurements especially in the vertical direction. Apparently most 'professional' autopilots rely on barometric pressure for altitude hold. This might not be critical once vision-based positioning is achieved, since it's the relative position between the two aircrafts that matters.



```
14.6   AUTO2   72     Stage 06:13
       OK      +2.3   ETA   N/A
       3D     /Target Alt
               +1m    206m / 205m

Batman
00:10:21 16.9m/s 80% ZICO_CIRCLE
         AUTO2   69  Time 03:47
```

```
flight_plan    name="Basic" lon0="-122.084677" m
  header
  waypoints
  blocks
    block      Wait GPS
    block      Holding point
```

```
18:39:20 Not_found
18:39:20 Joker, UNK
18:39:24 Joker, AUTO1
18:39:26 Joker, Standby
18:39:27 Joker, MANUAL
18:40:14 Not_found
18:40:14 Batman, UNK
18:40:26 Batman, MANUAL
18:40:28 Batman, ZICO_CIRCLE
18:40:34 Joker, Line 1-2
18:40:34 Joker, AUTO2
18:40:37 Batman, AUTO2
```

Figure 4: Ground Track During Autonomous Following

At this point, we have demonstrated that it's possible to use GPS alone to get the airplanes close enough for the trail aircraft to acquire a visual of the LEDs on the lead one. However, it's questionable whether the sensors and control algorithms currently in use are good enough to achieve precise navigation. This leads us to think that the following steps in this project ought to focus on two things: investigating whether sensor noise is an issue, and trying to implement more advanced/aggressive controllers that would allow us to achieve a better performance. With this in mind, we propose the following as future work:

- Carry-out test-flights to determine sensor accuracy [2]
- Log available states and control inputs and learn the aircraft model offline
- Utilize the aircraft model to realize some more advanced controllers (ex. LQR)

So far we have neglected the vision part of the project seeing how we have assumed it's an easy problem. But while it's true that with 4 LEDs a simple SVD decomposition is enough to determine position and orientation, it's interesting to investigate whether redundant LEDs offer an advantage despite the added complexity of the vision algorithm: we expect that the redundancy will lower the potential of loss-of-lock due to occlusion, but it complicates the problem of pose-estimation.

[2]Per example, attempt to fly with constant roll. Or simply fly somewhere without hills.

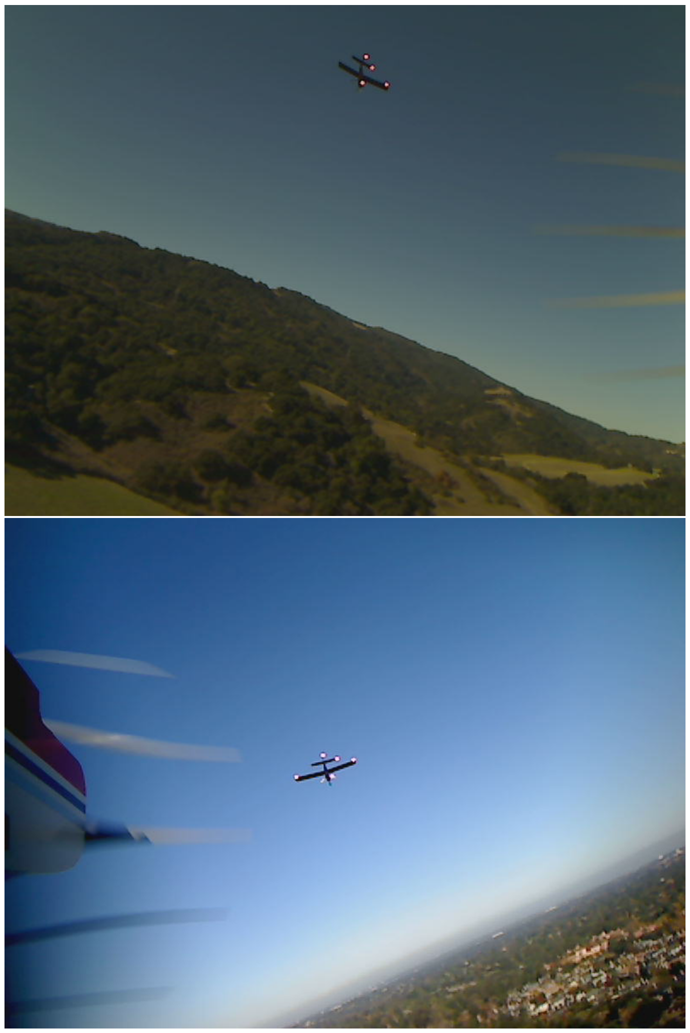


Figure 5: Sample Pictures of Lead plane from Trailing Plane (both autonomous)

References

[1] Mark Drela, Extended Vortex-Lattice Model. http://web.mit.edu/drela/Public/web/avl/

[2] Kevin Bruce, NASA B737 Flight Test Results of the Total Energy Control System. NASA CR-178285. January 1987.

A Hardware Description

A.1 The Paparazzi System

We use the Paparazzi Autopilot System as our autopilot unit to realize autonomous flight. Paparazzi is a free and open-source hardware and software project designed at ENAC University (France) for unmanned aerial vehicle (UAV) development. The hardware of the Paparazzi autopilot system consists of the Tiny control board, a GPS receiver, two IR sensor boards for attitude measurement, and a gyroscope to measure angular rates.

The Tiny control board uses the Phillips LPC2148 micro-controller as the main processing unit, and integrates versatile interfaces with various peripherals. For example, as shown in Figure 1, the control board interfaces with the GPS receiver and the radio modem via UART ports, while the IR sensors are connected to the ADC channels. Apart from controlling the servos and motor speed controller directly by outputting its own pulse-width modulation (PWM) signals in autonomous mode, Tiny Version 2 is also able to receive the pulse-position modulation (PPM) frame from the AR7000 Spectrum R/C receiver, and decode it into separate (PWM) signals in manual mode, so that a human pilot can takeover and control the plane via the R/C transmitter during takeoff, landing and emergency situations.

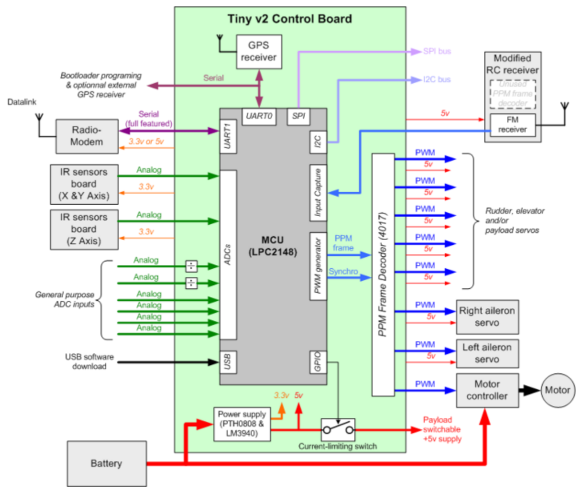


Figure 6: Tiny Version 2 control board interfacing with peripherals (http://paparazzi.enac.fr)

Unlike most autopilot systems which use inertia measurement units (IMUs) for attitude estimation, the paparazzi system uses infrared thermopiles for primary attitude sensing. The 3 orthogonal pairs of IR sensors measure the difference in IR radiation from the cold sky and the warm earth along the X, Y and Z axes, and thus estimate the orientation of the aircraft with respect to the horizon. While IR sensors are less susceptible to mechanical vibration than IMUs, they are slower in reaction time. Moreover, the IR sensors require manual calibration before each flight to compensate the difference in terrain and weather conditions.

A.2 Vision System

A.2.1 LEDs

In order for the follower plane to track the position and orientation of the leader plane visually, 4 Phlatlight CBT-120 Red LEDs, connected in series and powered by a 4cell LiPo battery, are mounted to the leader plane. The four LEDs are located at the right wing tip, the right landing skid, the right horizontal stabilizer, and the vertical stabilizer, respectively. The power and brightness of the LEDs is regulated by LM3433 current controller, and can be manually adjusted by tuning a rheostat. An electrical brushed-motor speed controller acts as a "switch" of the LEDs. The speed controller is connected to the gear channel of AR7000 R/C receiver, so that it lets current through only when the gear switch on the R/C transmitter is turned on. In this setting, the LEDs can be turned on or off remotely when the plane is in the air.

A.2.2 Camera and Vision

A webcam and a video processing computer are mounted to the follower plane so that it tracks the leader plane visually. Initially we used the Gumstix Overo embedded computer as the video processing unit. However, our vision processing program does not run well on the Gumstix system, therefore, we switched to fit-PC 2 computer manufactured by CompuLab, which has slightly larger size than the Gumstix computer, but with much more reliable performance. Fit-PC 2 computer runs on a Intel Atom Z530 1.6 GHz CPU and has 1GB DDR2 memory. A webcam is connected to the fit-PC 2 computer via one of its USB port. The footage captured during flight is stored in a 2GB micro-SD card plugged into the fit-PC 2 computer. The positioning information extracted by the camera is supposed to be relayed to the autopilot as an input sensor using an I2C link. This is still in the process of being done.

A.2.3 Interface between autopilot and vision system

In order to control the trailing airplane based on the data from the camera, we need to establish an interface between the fit-PC and the Paparazzi autopilot. On the fit-PC side, it is most convenient to use its built-in USB port. On the Paparazzi side, the only available data port now is the 3.3V I2C (Inter-Integrated Circuit) bus, which uses only two bidirectional open-drain lines, Serial Data Line(SDA) and Serial Clock (SCL), internally pulled up with 10k ohm resistors. The USB and I2C data are readily inter-convertible by the Devantech USB-I2C adapter module, which acts as the master I2C device that sends the SCL signal. Paparazzi acts as the slave device which receives SCL signal. Although Devantech USB-I2C adapter operates at 5V logic level, the Paparazzi Tiny is 5V tolerant and can handle the signal despite the difference in logic levels. However, the Devantech USB-I2C adapter also has internal pull-up resistors. Therefore, to ensure that the interface works properly, the pull-up resistors on the Paparazzi Tiny board are removed. A simple schematic of the interface is shown in the following figure:

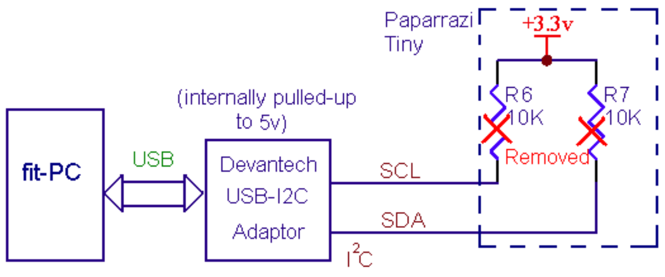


Figure 7: Schematic diagram of interface between vision and autopilot systems

A.3 Airspeed Sensor

Since the Tiny control board has a number of general purpose ADC channels, we can add more sensors to the system. Currently we have installed a MPXV4006DP differential air pressure sensor on to the plane to measure the airspeed during flight, so that we have better estimate on the relative speed of the plane. The differential air speed sensor has two holes, one connected to a brass tube extruding in front of the wing, the other hidden inside the wing. The sensor outputs a voltage proportional to the pressure difference measured by the two holes, thus allowing us to estimate the airspeed relative to the plane after a careful calibration on the sensor.

A.4 Datalink

Each plane is equipped with a Maxstream 9XTend RF module so that they can both send and receive command from the ground control system. The RF module is connected to the paparazzi system via a UART port and is set to transmit at 1Watt to increase range, while those on the airplane transmit at 0.5Watt to reduce power consumption.

A.5 Power System and Integration

On the leader plane, a 4S 14.8V Lithium-Polymer (Li-Po) main battery powers the entire paparazzi autopilot system, the R/C receiver, the Maxstream RF module and the main motor and the servos. A separate secondary battery is used to drive the LEDs, since they could draw as much as 5A of current. Similarly, the paparazzi system, main motor and servos of the follower plane are powered by a 14.8V main battery. The fit-PC 2 computer runs on 12V DC, and is thus powered by a secondary battery.
The airborne systems of both planes are integrated into a relatively compact package, most of which fits into the fuselages, so that drag during flight is minimized and on-field setup is simplified.

A.6 Ground Control Station

We use the GCS that comes as part of the paparazzi system to monitor and control the UAVs in flight. But although it does display the attitudes and positions of the two planes using simple graphics, one can hardly visualize the actual state of the planes with only the dots and lines. The paparazzi ground station is useful in tuning the control gains and setting up desired routes for the planes, so we decided to implement a secondary ground station just to visualize the orientations and relative positions of the planes.
Sending video directly down to ground would be a simple solution, but it takes significant amount of cost and transmit power to setup a video link with reasonably good quality, not to mention possible interference between the video and the datalink. Moreover, mounting additional batteries and a powerful transmitter onto the follower plane would definitely increase its payload and power consumption, thus reducing our flight-time and decreasing the relative percentage of induced drag.

Therefore, we made the secondary ground station 'eavesdrop' on the existing datalink for the states of the airplanes, such as position, altitude, attitude, velocities, and used Flightgear simulation to visualize these states. Flightgear is a free open-source project which creates advanced flight simulator framework for use in research and academic environments. In the Flightgear simulation, we use Rascal 110 as our aircraft model, which resembles our actual airplanes in size. The Flightgear program is run in parallel with a program which constantly receives downlink data and updates the Flightgear on the states of the aircrafts.

Thus, Flightgear displays the real time simulated view of the airplanes with sophisticated graphics, which allows us to better visualize their performance.

In the next stage of the project, fit-PC 2 computer on the follower plane will process the video and calculate its own relative position and orientations with respect to the lead plane by tracking the LEDs. These data will also be transmitted to the secondary GCS via the existing datalink and the lead plane will appear in the simulated scene seen by the follower plane in Flightgear simulation. Thus, we have a graphic visualization which resembles a real time onboard video to assist our control on the airplanes, while not incurring an additional video link.

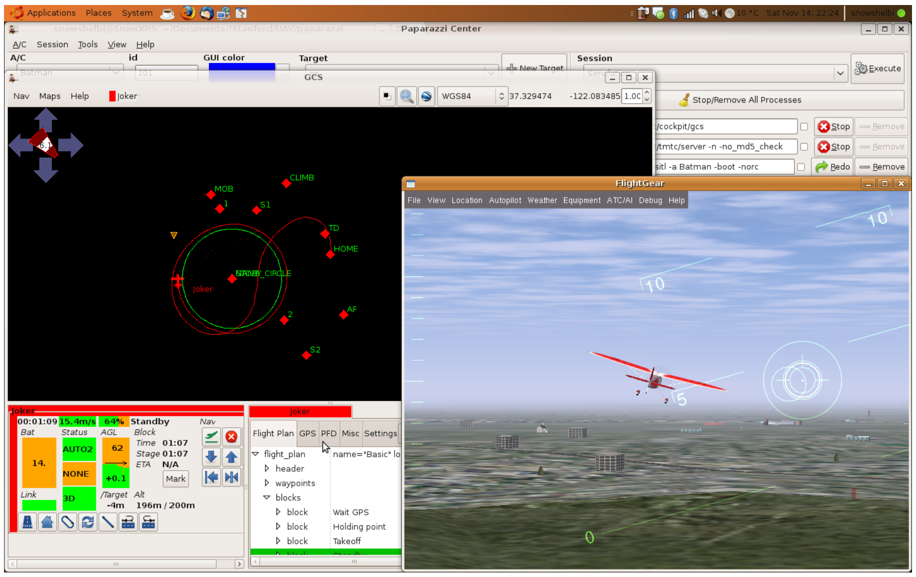


Figure 8: Sample of Simulation with FlightGear