Hexacopter
Progress Report

Brett Downing

Progress Report
Accurate judgement of progress to date and plan of further progress.

1 Preamble

Multicopters are increasingly popular... Simplicity... Market Saturation... rapidly changing legal situation... Recent Developments in power and control electronics.

Much of the technology related to multicopters is applicable to most other forms of UAV. Agriculture, mapping, targeted crop dusting, Cinematography, extreme-sport photography, Data collection, remote observation and inspection, hazard monitoring,

2 Notable Prior Work

Multicopters have received a great deal of attention by merit of their simplicity. A great deal of design work has come out of hobbyist communities such as DIY-Drones and

Their work often lacks scientific rigour, but frequently makes the first foray into experimental hardware, documenting for repeatability, and exposing their work to occasionally brutal peer-review.

2.1 Notable Technologies

Location stabilisation using Optical Flow Odometry Altitude estimation using optical flow methods (DIY Drones) Aggressive manoeuvres and precision flight with off-board processing Machine-Learning Feed-Forward controllers for accurate path following Computer Vision searching (Canberra UAV, outback challenge)

2.2 Notable commercial systems

Lily Camera Parrot AR-Drone Ardupilot Follow Me mode

2.3 Existing tools and software

2.3.1 Mission Planner

Mission Planner is a ground-station program written for UAVs that communicate over the MAVLINK protocol This will probably be our software of choice for monitoring the status of the drone during tests.

2.3.2 Tower

3 Where we started

3.1 Flight Hardware

The Hexacopter is a DJI F550 frame with fancy-pants motors and ESCs, 254mm props, 5Ah 11.1V Lipo battery. This platform gives us a generous lifting capacity, and enough flight time to run multiple tests.

3.2 Flight Electronics

The 2013 team decided to use a NAZA-V1-lite flight computer. This works well for free-flight, but does not make provisions for way-point navigation or telemetry.Interfacing is via Servo signals only.

In order to switch between manual and autonomous modes, the 2013 team installed a switching circuit. This circuit featured a 555 timer and eight relays. There was no schematic, and the only documentation was a copper layout. Only after meeting with the 2014 team did we become aware that there were dead channels on this board.

3.3 Autonomy Electronics

In Autonomous mode, the hexacopter is directed by a Raspberry Pi Model-B Single-Board-Computer. This platform is extremely widely used for all manner of projects simply due to its cost. It supports Linux and has drivers for almost any hardware we could hope to add to the system. It supports a camera over a dedicated interface. The server has a WiFi connection allowing us to remotely control the autonomous features, and even re-configure and recompile components during manual flight.

3.4 Autonomy Software

The on-board server is currently doing all of the autonomous navigation processing. This works reasonably well for slow manoeuvres but the latency in data collection, and the generation and re-interpretation of control signals leaves the Pi at a distinct performance disadvantage to autonomous routines running on the flight-computer.

The Pi runs a basic webserver hosting a page with basic autonomy controls. We can engage various

3.5 Sensors and auxiliary Hardware

The camera is stabilised for pitch and roll by two servos. The tilt servo was damaged and tilt stabilisation was almost non-existent. The servos receive correction information from the flight computer with a configurable gain parameter. The servos are able to hold the camera level in the steady state, but have a significantly delayed response.

The server collected position data from a second GPS module. This GPS module appears to have a filtering scheme with dead-reckoning configured for automotive applications; It exhibited lazy vertical response, and low accelerations.

Bearing data was to be collected from an X-sens branded IMU, though this was not installed at the time of hand-over, possibly due to an unreasonably heavy wiring harness. Instead, the navigation loops in the old code-base estimated the bearing by flying forward for a fixed interval before beginning navigation, relying on the flight computer to hold the bearing for the duration of the mission.

4 What we've achieved

Much of our efforts to date have been trying to correct the problems with the platform and its code-base.

Mathematics that estimates the location of a point of interest based on certain assumptions

With the heroic efforts of Jeremy Tan, we've implemented and tested code that we were able to salvage from the previous year-groups. Oddly, our tests of the previous year-groups' code have produced better results than are published in the respective papers, however the claims of the previous groups still appear grossly overstated. Various improvements to the hardware including landing gear, Wiring harness, enclosures

Reverse-engineering of circuits used by the previous groups. Having talked to the previous teams, we appear to have generated better documentation about the hardware than the teams had worked with initially.

Break-in to the NAZA GPS system, and subsequent removal of the Q-starz GPS. The NAZA GPS system uses a basic TTL-level serial protocol, but the message payloads are XORed with an obscure bit combination from the payload itself. It looks as though the protocol was deliberately obfuscated by DJI.

We found a forum that had a reverse-engineering effort for this protocol, and ported it to the Pi and its hardware serial port. This connection gives us access to the same GPS data that the flight computer is working with, reducing controller conflicts. It also gives us access to the NAZA's raw magnetometer data which could be used for bearing, but the inherent biases and changes due to unknown pitch and roll make this data difficult to use.

We've made a proposal and ordered parts to remove the NAZA-V1-Lite flight computer and replace it with a 3D-Robotics Pixhawk running the Ardupilot flight control software. This change allows us to utilise the vast array of supporting software that the Ardpilot community has written including ground-stations, telemetry loggers, Smart-phone apps, Kalman navigation filters and automatic flight control tuning. It also exposes all of the live flight parameters, raw and processed, available over a documented interface. This decision was partly motivated by our observations of an older APM-2.6 flight computer running similar firmware.

After breaking into the flight computer and agreeing to change to the Pixhawk, the X-sens IMU was never added to the copter, and the drivers were removed from the code-base.

5 What we intend to do

5.1 Go Fetch

The Object Tracking, chase-cam

5.2 Everything in Its Place

The object tracking code we ported over from the 2014 team took a relatively simplistic approach, feeding the pixel position from the image directly into two PID controllers. This did work, but altitude, camera angle, parallax, and other variables tended to make the controller go from a-bit-weak, to utterly unstable, within seconds.

We've built up a code-base to estimate the physical position of the target in coordinates relative to the copter, trying to sanitize the inputs to the control loop. The control loop now takes inputs of the target coordinates in metres. Our controller is still a collection of basic PID loops, but already, the behaviour is far more consistent in flight.

We expect to improve on the physicality of this, changing the controller outputs to units of metres per second, and estimating latitude and longitude of the target to assist in lost-target recovery (Section 5.2). Logging the estimates of the target's global coordinates is likely to lead to insights into the stability and validity of our work, and possible improvements.

5.3 Lost and Found

The copter is already beginning to exhibit strong tracking behaviour but having encountered a camera stability issue (Discussed in 6.2), we've not yet been able to apply aggressive parameters to the chase algorithm and the tracked object frequently leaves the field of view of the camera. We intend to write a search algorithm to attempt to locate an object shortly after it has left the frame.

6 Major Challenges

6.1 Limiting Controller Complexity

A method for the copter to chase an object on a screen is a relatively straightforward problem to grasp intuitively, but include camera geometry, position estimation, velocity and acceleration limits and such; just defining the problem turns into a wall of mathematics. We've made a lot of towards sensible estimation and chase routines already, but already the single-input-single-output PID controllers are becoming inadequate. The control scheme will need to be analysed in a more comprehensive manner, incorporating data sets from multiple sources to coordinate coherent actions in a multi-input-multi-output controller.

Limiting the scope and complexity of the controller may well become a matter of identifying diminishing returns. Rigorous flight performance analysis using sensible metrics, monitoring the time spent coding those incremental improvements, and assessing the wider applicability of the possible changes may help to identify when the controller should be declared "good enough".

6.2 Flying on a Steady Cam

As the platform stands, The gimbal does not sufficiently isolate platform motion to stabilise a simple control loop. For example, a forward motion instruction from the chase algorithm causes the copter's rear rotors to throttle up, then the platform pitches forward, then the copter accelerates forward. The pitch data is sent to the camera stabilisation servo which has a slow slew rate and a small delay time. This delay couples pitch motion into the image processing loop and adds a brief but intense upward swing to the location of the target in the image as the copter accelerates. This behaviour is controlled by the physical properties of the copter, the flight-computer's control loops, the servo's controller and the shutter lag on the camera. While all of this is technically possible to compensate for, calibrating against mechanical and electrical properties of a commodity servo and reverse-engineering a third-order control loop introduces a large number of variables and doesn't lend itself easily to empirical validation. Many of these parts were not selected with any great care either and we fully expect to swap out or modify parts of the copter during the course of the project.

It makes more sense to solve single whole problems; either cleanly stabilising the camera with a higher performance gimbal, or applying software image stabilisation using motion data from the flight controller to calculate through bulk motions followed by optical flow methods to remove the remaining jitter. The change to the Ardupilot flight computer will give us access to IMU data required for software stabilisation, but it may prove too computationally intensive for the PiV2 to process.

6.3 Tough Shit

When we were first shown the platform, my first thoughts were WTF is all that spaghetti doing on a flying machine. Between dead servos, incomplete configurations, undocumented custom circuits with known but undocumented faults, and a wiring loom that was unreliable, undocumented, and even simply incorrect; we've been set back by days at a time with intermittent faults.

The change to the Pixhawk is set to remove and simplify a large portion of the wiring harness, but maintaining a high quality of work over a long project is difficult; small, temporary hacks tend to become permanent fixtures. We believe we have the technical skill in the group to leave the wiring harness at least presentable, if not bullet-proof.

6.4 Extensible Code

Our team has selected projects that are able to begin very independently and achieve clearly defined goals, but then a series of hand-overs are to occur in which our work circulates, and each of our results are used to enhance the work of another. For this to happen, our code needs to reasonably well documented and conform to some kind of standard. As with the hardware modifications, maintaining a high degree of quality across the board is difficult, but we hope that this code-rotation measure will force some level of accountability to documentation and code cleanliness.

7 Timeline

Gantt chart here

7.1 Milestones

7.1.1 Following the Leader

Copter can follow a visual cue at low accelerations.

7.1.2 Never Give Up, Never Surrender

Copter implements a basic search for a lost target.

7.1.3 Out-Pace the Red-Shirts

Copter can keep a visual cue in sight under aggressive accelerations

7.1.4 What are you doing up there?

Copter's chase algorithm includes additional data to fully locate in three dimensions. (Inclusion of other sensor data for superior state estimation (PIKSI, Lidar, Sonar))

7.1.5 You Can't Run, You Can't Hide

Copter can make a time-dependent, weighted search space based on the target's previous motion, and search it.

8 References

Holographic transforms Target coordinate estimation with landmarks Optical flow uncertainty map Altitude estimation using optical flow methods