THE UNIVERSITY OF WESTERN AUSTRALIA

FINAL YEAR PROJECT

Computer Vision Chase-Cam UAVs

Author:
Brett DOWNING

Supervisors:
Prof. Thomas BRAUNL
Chris CROFT

October 5, 2015

Abstract

Multirotors are here to stay, and may soon be expected to interact in a human environment. Commodity quadcopters are advertising capabilities to act as chase-cams and turn-key mapping solutions, but none of the current generation commodity uav chase-cams offer computer vision driven or even assisted flight modes to improve tracking, image framing or obstacle avoidance. Such vision assisted routines would also apply to autonomous or semi-autonomous inspection tasks for fixtures in remote or hazardous environments.

In this project, we build on the results of previous year groups and implement turn-key waypoint navigation and failsafe methods using the ardupilot software stack, and develop robust object tracking, data collection behaviours and exclusion zones on a computationally starved platform with an aim to integrate vision assisted behaviours in low-cost, lightweight UAVs.

Contents

1 Introduction

Background, Aims

1.1 Motivation

Recent Developments in power and control electronics has allowed small, consumer level UAVs to lift enough processing capacity to navigate at least partly by computer vision.

A number of Commercial Micro UAVs are beginning to advertise the ability to act as an autonomous chase-cam [?] [?]. Almost all of these systems rely on a tracking device on the user, and many navigate entirely by GPS. While the performance of GPS is improving, the performance of a chase-cam will be limited to the update rate and fix accuracy of the beacon and drone GPS modules. It also requires a lock to be achieved in both devices before tracking can commence.

The power distribution and plant monitoring sectors have recently been taking on small fleets of drones to perform routine inspection of remote, hazardous or otherwise difficult locations [?]. Many of these applications are reasonably repetitive and well defined enough to fully automate, but cannot be navigated by GPS alone due to the poor repeatability of the GPS system. During this project, we were approached by a mining safety startup interested in using UAVs for routine site inspections.

The UAVs we are targeting operate with small payloads, must react quickly to changes in the environment, and frequently feature camera systems. Despite falling costs of hobby and toy grade multirotor systems, Collision avoidance, Object tracking, mapping and inspection are not well catered for in the current market of UAVs. Part of the problem is that very few sensors are of the appropriate scale, or mass for low cost UAVs. Ground based vehicles have an advantage that a 1D sensor need only be swept in one axis to search for obstacles, but a UAV must collect data filling a volume ahead of it. About the only form of sensor capable of doing this is a 3D laser scanner which is a high-precision, high-cost, high-mass device. A simple camera collects data from an appropriately large volume at high enough speeds to navigate a multirotor, but the data is often difficult or computationally expensive to interpret.

With environmental data collected by an efficient computer-vision routine, the current generation of multi-rotor devices would be able to interact in a human environment.

Computer vision assisted control routines would permit common low cost UAVs to better frame extreme sport chase-cam film reels, automatically avoid obstacles, and stabilise GPS variances in remotely operated inspection tasks.

In this study we investigate the usability of computer vision alone to track and follow a moving target. It is hoped that this work can be used to improve the performance of camera tracking routines in autonomous chase-cam and cinematographic applications, and partially automate remote inspection

Much of the technology related to multi-copters is applicable to most other forms of UAV. The vast array of applications micro UAVs have found suggests this work may find use in Agriculture, mapping, targeted crop dusting, Cinematography, extreme-sport photography, Data collection, remote observation and inspection and hazard and environmental monitoring.

2 History / prior works / relevance

3 Project History

3.0.1 2013

The 2013 team, O'Connor [?] and Venables [?] established the project with the purchase of a DJI flame-wheel F550 Hexacopter with a NAZA-lite flight controller. This copter was fitted and tested, and finally converted into an autonomous platform with the addition of a Raspberry Pi single-board computer (RPi) and a circuit to switch the control channels from the radio receiver to the RPi GPIO outputs. The NAZA-lite at this time did not feature telemetry outputs or way-point inputs, but it was able to loiter in a stiff wind using a GPS fix. This team gathered location information for the RPi using a Qstarz GPS unit, and bearing information using an X-sens IMU. The sensor duplication did not exceed the maximum payload capacity of the platform, but it did suffer a short flight-time. Under direct control from the RPi, the drone was able to execute basic way-point navigation.

3.0.2 2014

The 2014 team, Baxter [?], Mazur [?] and Targhagh [?] continued the project adding an internet accessible web UI to control the various autonomous features of the platform, displaying the flight-path of the copter and a live feed of the camera. The navigation routines were improved and extended to permit operation without reliable heading information, and the computer vision routines were extented.

3.0.3 2015

The project is being continued by Allen [?], Mohanty [?], Tan [?] and Downing [?]. Because of the rapid pace that the market is adopting new features and technologies, we saw fit to undertake a critical review of all aspects of the platform, and begin improvements that facilitate the current typical feature set.

3.1 Platform

3.1.1 Description

This project centred around a hexacopter based on the DJI F550 frame. This frame allows for a generous lift capacity, and plenty of space to fasten flight assist hardware, and represents a low cost platform so that our work can be reproduced by a sufficiently motivated hobbyist.

In this work, the flight tasks are based around two physically separate processors; the Flight Controller, and the Server. This allows a stable, known safe build to be maintained on the flight controller, while experimental code runs on the server. If the server fails for any reason, the flight controller will maintain flight, and it allows the operator to engage various fail-safe behaviours without having to rely on the experimental code. At hand-over, the server was a Raspberry Pi model B+, and the fight controller was a NAZA lite. This combination did not go together smoothly. The NAZA lite was designed as an entry level free-flight controller and did not expose any interfaces for telemetry or data acquisition. As such, the previous teams had to duplicate the GPS and IMU sensors to have that data available to the server's algorithms.

The NAZA also did not feature waypoint navigation features, or secondary command channels so the previous teams used a relay board to physically switch the control inputs from the RC receiver to the server, and implemented their own flight control algorithms on the server. We were told that server lock-ups frequently caused the craft to power-dive.

The server has been upgraded to a Raspberry Pi V2 single board computer, which we consider to be a computationally starved platform. At the time of writing, the Raspberry Pi V2 has less computational power than a smartphone in the $200 price bracket.

Flight controller processor, gimbal

3.1.2 Platform Weaknesses

(fitness for purpose etc) We made some attempts to break into the data channel between the NAZA flight controller and its GPS module, but the raw GPS and compass data was not particularly useful without the accelerometer and gyro data. This channel had also been deliberately obfuscated by DJI, making it quite clear that this was beyond the design intent of the flight controller. A community RC group had reverse engineered this link, and we ported their code to the raspberry pi and removed the Qstar-z GPS unit.

The physical switch of control from the user to the server, and the ongoing development of a control loop meant that RC switch was never configured to surrender the throttle channel, and the server was left without altitude control. This meant that the server had continuous perturbations from the user who remained responsible for maintaining a fixed altitude by sight alone.

The NAZA is a robust, if basic free-flight controller, and did not feature telemetry. The server did not have access to an estimate of remaining flight time, and the user could not retrieve logs in the case of a crash.

Waypoints were handled on the server despite the NAZA featuring GPS assisted loiter behaviour, and a reliable return-to-launch failsafe feature. In general, control loops should be as tight as possible and routing them through an entirely separate processor is clearly an unnecessarily long loop.

3.2 Team Achievements

After spending some time with the hardware configuration of the previous teams, we prepared a report outlining the weaknesses and proposing a change. We swapped the NAZA flight controller out for a 3DR Pixhawk running the Ardupilot 3.3 firmware. This gave us Waypoints, altitude control, failsafes, telemetry, and a GPS-only driven follow-me mode out of the box. The open and configurable flight controller with built-in autonomous and hybrid modes, meant that we could fully surrender control to the autonomous platform, including altitude control. The built-in mode change behaviours meant that we could relinquish and resume manual control at the remote, or activate fail-safes just as reliably as with the NAZA. The telemetry link supports the MAVlink protocol, which defines various commands from interrogating the status to setting waypoints, or even setting the instantaneous velocity vector. This let us keep the high-speed components of the control loops firmly inside the flight controller, and direct it with the lower speed vision tasks. This also gave us linear control inputs in SI units. The new configuration allowed us to fully eliminate duplicated hardware, an opportunity to fully re-build the wiring harness, and significantly reduce the weight of the craft, extending the flight time to almost twenty minutes.

We saw fit to refresh the Web UI and HUD, implementing a more appealing and maintainable template system, We implemented a Waypoint modifier routine using polygonal no-fly zones, Glyph detection Capture and tagging of images to suit proprietary offline 3D reconstruction algorithms. Computer vision driven chase-cam behaviour

By using a fully-featured open-source flight controller, we also had access to the community developed simulation environment. The Ardupilot simulator is designed to simulate the behaviour of the flight controller against various hardware configurations and flight styles. As such, it incorporates a comprehensive inertial model and noise injection to the various sensors. We were able to compile our tests for our development machines couple them to the simulator over a local TCP socket, and test them against the same version of the flight control firmware as we had uploaded to the physical copter.

4 Object tracker

4.1 Position estimation

After experimenting with the rudimentary PID loop, we moved on to basic triangulation methods to estimate the location of the object with the assumption that it was on the ground. We were estimating the height based on the GPS altitude data streaming from the NAZA GPS, but without access to the NAZA's internal barometer, the uncertainty in height caused the navigation loop stability to vary significantly minute by minute. After the upgrade to the Pixhawk, we had access to the data from the Extended Kalman Filter in the flight controller. We had also made significant changes to the way the flight controller was guided by the server, and this prompted a near total re-write of the object tracking code-base.

4.2 Desired relative pose

Once the object is located in 3D space, the copter will calculate a vantage point from which to view it. This vantage point could include information regarding the terrain, the size and trajectory of the object, the field of view of the camera etc. This is the point at which cinematographic style is included in the algorithm.

4.3 Observations and uncertainties

4.3.1 Structure from image

Resolving structure from image is a field unto itself. In general, the computational load of modelling the environment in three dimensions live is beyond supercomputing clusters that would fit in the mass restrictions of most multi-rotors. However, Simultaneous Localisation and Mapping (SLAM) systems have optimised components of this field to a vast array of algorithms of varying computational power requirements. One of the critical features of a modern SLAM systems is an acknowledgement that every measurement comes with some uncertainty. In many cases, the treatment of uncertainty can be very simple, carrying a one-dimensional confidence value, to the very complex, where every aspect of the measurement is recorded for post-processing where non-linear couplings can be accounted for.

Some systems are beginning to feature graph-traversal techniques where observations are stored as relative links in a graph, and networks are drawn through those links to optimise against the uncertainty of various aspects of the environment.

4.3.2 Uncertainties

A good uncertainty model encodes everything that is known and nothing of what is unknown. Good treatment of uncertainties combines knowledge without losing information, or adding assumptions. In the case of structure from image, a system that extracts maximal information from a single camera should be automatically capable of full stereoscopy using only the uncertainty analyses that applied to the monocular case. Monocular Camera uncertainty model Lidar Lite Uncertainty model GPS uncertainty model Time evolution uncertainty models Base Utilities Vector sum combination point sample Kalman style filtering Relative uncertainties Networks of relative measurements Graph traversal Nodes: Objects' time history (including self) Survey Markers Assumptions (GPS objects?) Links Relative Observations Motion Estimates IMU data Kalman style interpolation

4.4 Current implementation

4.4.1 Strengths

Encodes appropriate information in a covariance matrix form Cleanly integrates multiple observations into an object model Makes assumptions explicitly

4.4.2 Weaknesses

Camera model does not account for uncertainty from angle of pixels, gimbal pose, IMU sample time (angular velocity), GPS drift GPS drift is deliberately ignored in favour of applying a generous velocity uncertainty to the objects. Cannot describe complex distributions (unbounded polynomial order) arbitrary geometry Uncertainty Images from SoggyDog

4.5 Future Work

Velocity interpolation not implemented. Graph traversal (SLAM) not implemented. Describe hyperbolic distribution and flaws Describe Laser beam necking case

5 Test Results

5.1 Simulation tests

Gimbal pose, IMU data, GPS location etc We used

Camera uncertainty models combined with assumptions Pretty pictures Multiple observations with time-evolution

5.2 Live tests

Able to physically follow one object, while tracking multiple others. Acceleration limits, velocity limits, time-cutoffs. pitch and roll stability Basic colour matching limitations Not as robust as radio links, but

6 Future Work

Expand uncertainty analysis SLAM, end to end solution SIFT algorithm Optical Flow Recommendations: ROS, ROS, ROS.

7 Conclusions

We have greatly improved the capabilities of the UWA autonomous hexacopter platform, and have brought the code-base up to a level where it is feasible to implement a SLAM process.