Estimation and Control for an Open-Source Quadcopter

Inkyu Sa and Peter Corke*

Abstract

This paper describes modelling, estimation and control of the horizontal translational motion of an open-source and cost effective quadcopter — the MikroKopter. We determine the dynamics of its roll and pitch attitude controller, system latencies, and the units associated with the values exchanged with the vehicle over its serial port. Using this we create a horizontal-plane velocity estimator that uses data from the built-in inertial sensors and an onboard laser scanner, and implement translational control using a nested control loop architecture. We present experimental results for the model and estimator, as well as closed-loop positioning.

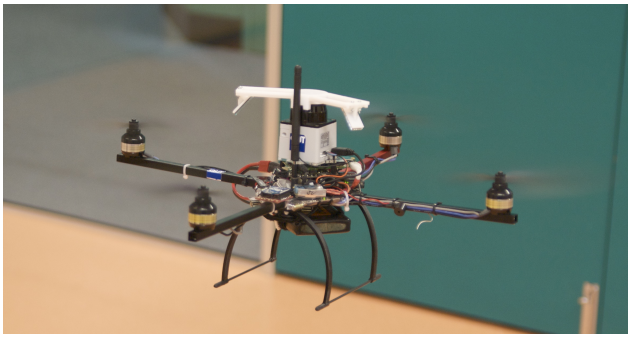


Figure 1: The MikroKopter in flight. The payload comprises an Hokuyo laser scanner, a ZigBee communications link and a Gumstix embedded computer with WiFi.

1 Introduction

Quadcopters are compact rotor craft air vehicles with vertical takeoff and landing (VTOL) capability. Like a conventional helicopter they can hover, but have significant other advantages such as ease of piloting and mechanical simplicity — they have no swash-plate mechanism. Although quadcopters have become very popular in recent times, the first quadcopters were built back in the 1920s and carried passengers! One of the earliest robotic quadcopters was by Pounds et al. [13, 14] at a time when it was necessary to build a vehicle from scratch. In recent times there has been an explosive growth of interest in quadcopters [3, 7, 12, 16] spurred by the availability of research-grade platforms and motion capture systems such as VICON systems for control [8, 10, 11].

The simplicity of the quadcopter makes it easy and cheap to build and there are a number of low-cost radio-controlled toy quadcopters. Some such as the Parrot can be used for interactive gaming. At the other end of the spectrum are remote-control photography platforms such as the DraganFly X8 and research-grade quadcopters such as the Falcon or Pelican from Ascending Technologies which have driven much recent research [1, 6, 9]. In the last couple of years a new intermediate class of vehicle has emerged — the serious amateur class — which is exemplified by the MikroKopter project (`http://mikrokopter.org`). This is an open-source project where the mechanical, electronic and software design is available and it can be purchased in component form, as a kit or a ready-to-fly set. The MikroKopter has a mass of 0.65 kg and an endurance of 18 minutes (with 2200 mAh battery). The nominal payload is 0.25 kg but we have lifted a payload of 0.85 kg although this reduces endurance to 8.3 minutes. For robotics research this platform offers a lower cost and higher payload than the research-grade vehicles but it has been disadvantaged by a lack of documentation and technical detail that would satisfy a researcher. A high payload is a significant advantage for research since prototyping becomes very difficult when having to worry about every added gram.

This paper presents several contributions. Firstly we show that a low-cost high-performance amateur-grade quadcopter, the MikroKopter, can be used for serious robotics research. Secondly we present details of the vehicle's sensors, control system and dynamic performance as determined through reverse engineering and system identification. Thirdly, we develop a velocity state estimator which compared to others

*The authors are with the CyPhy Laboratory, Queensland University of Technology, Australia. `i.sa@qut.edu.au`, `peter.corke@qut.edu.au`

such as [2] is computationally cheap, easy to tune and effective. Finally, we demonstrate stable hover using the built-in low-cost inertial sensors, a 10 Hz laser scanner and a simple nested control loop architecture.

The details are presented in the remainder of this paper. Section 2 details the characteristics of the MikroKopter platform, and Section 3 details our ROS-based control system which includes laser-scanner based localization. The dynamics of the quadcopter for translational motion are summarized in Section 4 and the estimation and control approach is described in Section 5. We present our experimental results in Section 6 and conclusions and future work in Section 7.

2 Understanding the MikroKopter

2.1 The flight controller

The MikroKopter comprises four brushless-DC motors each with its own speed controller. These communicate via I^2C bus to the central flight controller board. The flight control board version 2.1 is based on an Atmega 1284 processor running at 20 MHz which implements the state estimator, control loops, decodes the pulse stream from the radio control receiver, and also receives commands over a serial data port and transmits status information. The flight control board holds a triaxial MEMS accelerometer and gyroscope, and a barometric pressure sensor. A magnetic compass can be fitted but we do not use one. Multi-channel input from a Futaba handset is read via a digital input pin. For safety the handset must be active to enable the quadrotor to fly. The flight controller has a serial port which can be used to receive commands or transmit status information. This is connected to a Zigbee module which allows commands and status to be communicated wirelessly as shown in Figure 3.

2.2 Flight control firmware

The great advantage of the MikroKopter is that the flight control software is open source — nearly 7000 lines of C. The non trivial disadvantages are that the code is written in German; it is poorly documented, particularly the serial data input and output protocol; and the state estimation algorithm is not any sort of familiar filter. These factors are not a limitation to somebody who simply wants to fly, but they are a problem to a roboticist who wants to control the vehicle via the serial port. For example, what exactly does "roll command" mean? Is it an angle or a rate, and what are the units?

We have attacked this problem in several different ways. Firstly the language issue. Reading code written in a different language is surprisingly difficult since the variable and function names are not mnemonic and the comments are opaque. Others have attempted to translate the code to English but this endeavour is always behind the current version of the software. We therefore wrote a simple Python script to "translate" the source code into English using a combination of keyword substitution and Google Translate which has a REST-based web service using JSON encoding. In a few seconds we can automatically create a good enough translation to allow the code to be understood.

Secondly, we reverse engineered the flight controller. Analysis of the flight control board schematic and sensor chip data sheets tells us the scale factors between sensed quantities (accelerations, rates, pressure) and voltages applied to the ADC pins of the microcontroller. Analyzing our translated code tells us the scale factors from pin voltages to internal state variables.

Inertial sensor bias is determined by averaging the sensor values on power up. The attitude estimator is based on a roll, pitch and yaw angle representation. It is a complex piece of code that is not obviously a Kalman filter or a complementary filter, but it does appear to take both gyro rates and accelerations into account. The sensors are measured at a high rate (> 1 kHz) and averaged into global variables which are read by the control thread which operates at 500 Hz. Roll and pitch control is achieved with a PD control loop.

2.3 Flight controller communications protocol

The flight controller communicates with a simple packet protocol over a serial port running at 57600 baud. The packets have a header and a 2-byte checksum (not CRC16). The `ExternControl` packet (11 bytes) provides the same inputs to the flight controller as it receives from the radio-control receiver and is a convenient way to allow control from a computer (or the MikroKopter Navi board which adds GPS waypointing capability). Roll and pitch values in this packet are 8-bit signed integers that represent desired roll and pitch values in degrees.

The `DebugOut` packet (66 bytes) provides important state information from the flight controller such as raw sensor values (inertial sensor values, radio-control receiver values), estimated state values (attitude, height) and current motor demand values. The values are all integers. From code analysis and simple static measurements of acceleration values as a function of vehicle inclination we have determined that attitude is given in units of $\deg/10$ and accelerations are in units of $g/611$. The accelerations `AccRoll` and `AccNick`[1] are respectively $-\ddot{y}$ and $\ddot{x}$ in the body frame.

2.4 Coordinate system

The MikroKopter adopts the convention that the red rod is the front and the pitch command control forwards (positive) and backwards (negative) movements. The roll command controls left (positive) and right (negative) movements. We adopt a standard aerospace coordinate convention with x-axis forward and z-axis downward and MikroKopter angles and accelerations are sign adjusted to fit this right-handed convention.

[1] Nick is German for pitch.

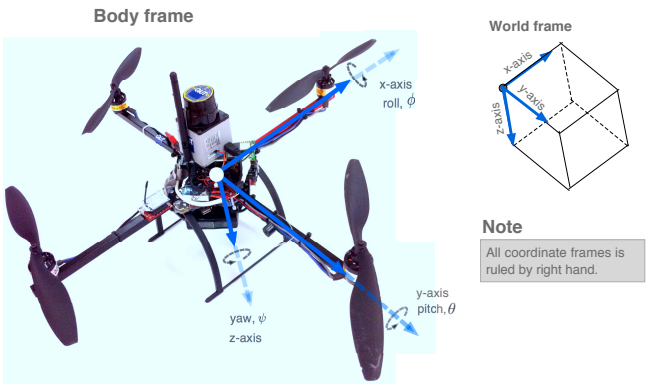


Figure 2: Coordinate system. We use standard aerospace conventions and convert all MikroKopter angles and inertial measurements to this convention.

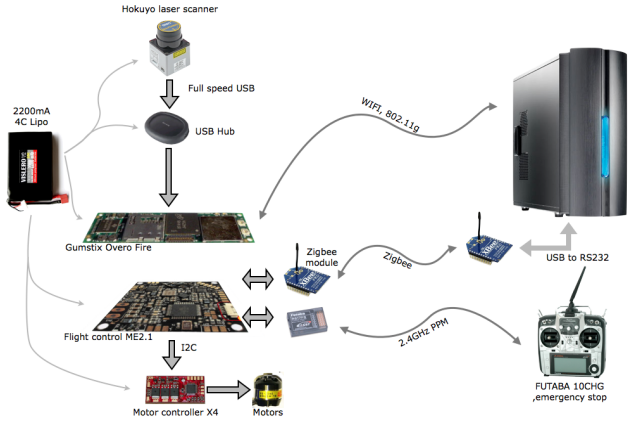


Figure 3: The laser scanner is attached to a USB Hub because the Overo Gumstix USB host only supports High Speed USB. The Zigbee module is used to transmit IMU data to the ground station and receive commands. The WiFi connection connects the ROS nodes on the Gumstix to the ground station. For safety a manual pilot transmitter is linked to the quadrotor system.

3 Overall system architecture

Figure 3 shows the hardware architecture proposed for this work. The quadcopter carries an Overo Gumstix which runs Ubuntu Linux and ROS (`http://ros.org`). An Hokuyo model URG-04LX laser scanner (10 Hz and 4 m range) scans in the horizontal plane and the "laser hat" from the City College of NewYork provides altitude as well. The total payload mass is 0.18 kg.

Figure 4 shows our implementation based on ROS, where dark grey boxes denote the ROS nodes which are individual processes. The Overo runs the standard ROS laser scanner

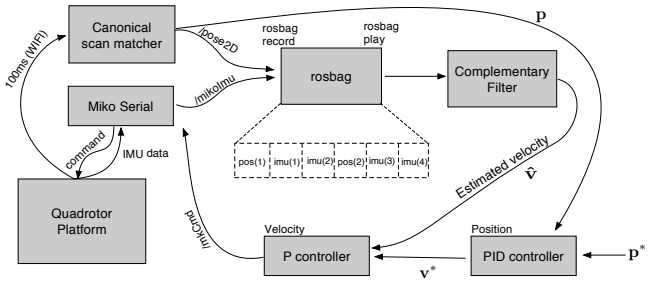


Figure 4: Software implementation using ROS platform where dark grey boxes represent ROS nodes and the prefix '/' denotes a ROS topic. p is position and * denotes demand.

node and publishes the topic */scan* over WiFi to the base station every laser scan interval (100 ms). The ROS canonical scan matcher subscribes to this topic, and estimates 2D pose (x,y,θ) using an ICP algorithm [5] which it publishes as topic */pose2D* . The ROS serial node runs on the base station and communicates with the MikroKopter flight control board over the ZigBee link. Every 50 ms it requests a `DebugOut` packet which it receives and the inertial data (converted to SI units and the our coordinate frame) is published as the */mikoImu* topic. This node also subscribes to the */mikoCmd* topic and transmits the command over the ZigBee uplink to the flight controller. These topics can be recorded in a log file (ROS bag format) and later replayed (using `rosbag`) to test the state estimator and controller.

3.1 Timing and latency

In order to determine the system latency between the Gumstix and the base station, we exploited the network time protocol (NTP) server. NTP allows us to synchronize the clocks of the two systems to better than 1 ms. After synchronisation we measured times using the ROS time stamp function on code running on the vehicle and the base station. This function provides micro-second accuracy and we estimate the WiFi latency as $\approx$ 7 ms. There is a 100 ms time delay from start of laser scan to receipt of data, 7 ms of WiFi transmission, and 5 ms for the ICP scan matcher. The inertial data is collected using a concurrent thread on the base station running every 50 ms. The latency in sensor values is around 12 ms and is dominated by serial transmission of the `DebugOut` packet. In summary the total delay is of the order of 110 ms.

4 Quadcopter dynamics

The quadcopter is an underactuated force-controlled system. In order to translate it must first rotate so that a component of its thrust vector can exert a force to accelerate it in the desired direction. In order to rotate it must adjust the rotors to maintain a constant vertical thrust component while creating a moment to rotationally accelerate the vehicle to the desired attitude. Our position controller generates desired roll and pitch angles for the high-bandwidth attitude and vertical

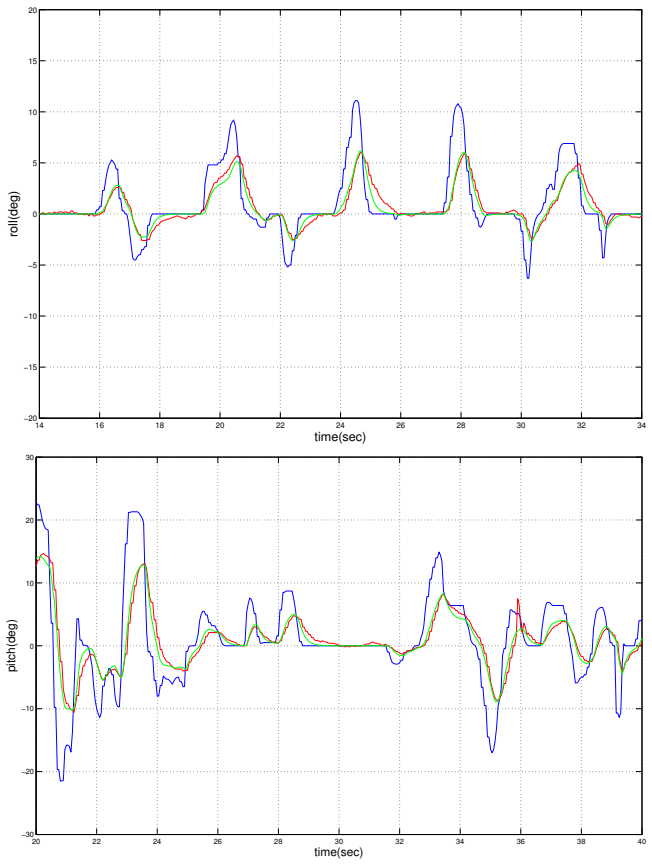


Figure 5: Measured and predicted roll (top) and pitch (bottom) angle for manual flight. Blue denotes the commanded stick input, red line the measured angle from the MikroKopter and green represents the model response.

thrust control loops provided by the MikroKopter flight controller.

4.1 Attitude dynamics

The MikroKopter flight controller runs PD controllers at 500 Hz for pitch and roll angle. We logged input commands and MikroKopter attitude estimates, see Figure 5, for manual flight. Using recursive least squares we fit an ARMAX model to this time series data giving a linear discrete-time (at 50 ms) model

$$F(z)_{pitch} = \frac{0.148}{z - 0.7639}$$
$$F(z)_{roll} = \frac{0.145}{z - 0.7704}$$

which corresponds to a rise time of ≈ 0.4 s. The response of these models to the pilot input is compared to the measured response in Figure 5. We have experimented with adjusting the PD parameters in the flight controller firmware and can increase the control bandwidth but this causes human piloting to become more challenging.

4.2 Equations of motion

We define three coordinate frames: the world frame $\{o\}$, the body-fixed frame $\{b\}$, and the frame $\{Q\}$ which has the same origin as $\{b\}$ and the x_Q and y_Q are the projections of x_b and y_b onto the world horizontal plane.

In the x_Q-direction the equation of motion is

$$m\,{}^Q\ddot{x} = T \sin\theta \tag{1}$$

where T is the total rotor thrust and θ is the pitch angle. In vertical equilibrium $T = mg$ and the small angle approximation $\sin\theta \approx \theta$ allows us to write this as a linear equation

$${}^Q\ddot{x} = g\theta \tag{2}$$

which is a classical double-integrator plant. This ignores aerodynamic drag which would add a small damping term proportional to ${}^Q\dot{x}$, and out of balance forces due to poor trim which would add a constant acceleration.

Using only position feedback the open-loop dynamics include two free integrators, the first-order attitude response and just over 1 sample time delay. In the z-plane these are poles at +1, +1, 0.959 and 0 which is not stable for any finite gain. Introducing a zero would help but a realizable lead compensator requires an additional pole at the origin which again leads to instability. A better alternative would be feedback of vehicle translational velocity.

5 Estimation and control

5.1 Complementary filter

To increase damping we require a high-quality velocity estimate: smooth, high update rate with low-latency. Differentiation of the position from the ICP pose estimator results in velocity at 10 Hz with a latency of 150 ms or 1.5 time intervals which significantly limits the gain that can be applied when used for closed-loop velocity control. Instead we use the MikroKopter acceleration measurements (`AccRoll` and `AccNick`) which we read at 20 Hz with low latency and integrate them to create a velocity estimate. We subtract the acceleration due to gravity using the MikroKopter's estimated roll and pitch angles

$${}^Q\ddot{x} = \frac{a_x + g\sin\theta}{\cos\theta} \tag{3}$$
$${}^Q\ddot{y} = \frac{a_y - g\sin\phi}{\cos\phi} \tag{4}$$

where a_x, a_y and the measured acceleration from the flight control board converted to our coordinate system, and θ, ϕ denote the pitch and roll angles respectively. Any estimator that relies on integration is subject to substantial errors due to drift, even over quite short time intervals. We therefore fuse these two estimates using a simple discrete-time complementary filter [15]

$$\hat{v}_x(t+1) = \hat{v}_x(t) + {}^Q\ddot{x}(t) + K(\tilde{v}_x(t) - \hat{v}_x(t))\Delta_t \tag{5}$$

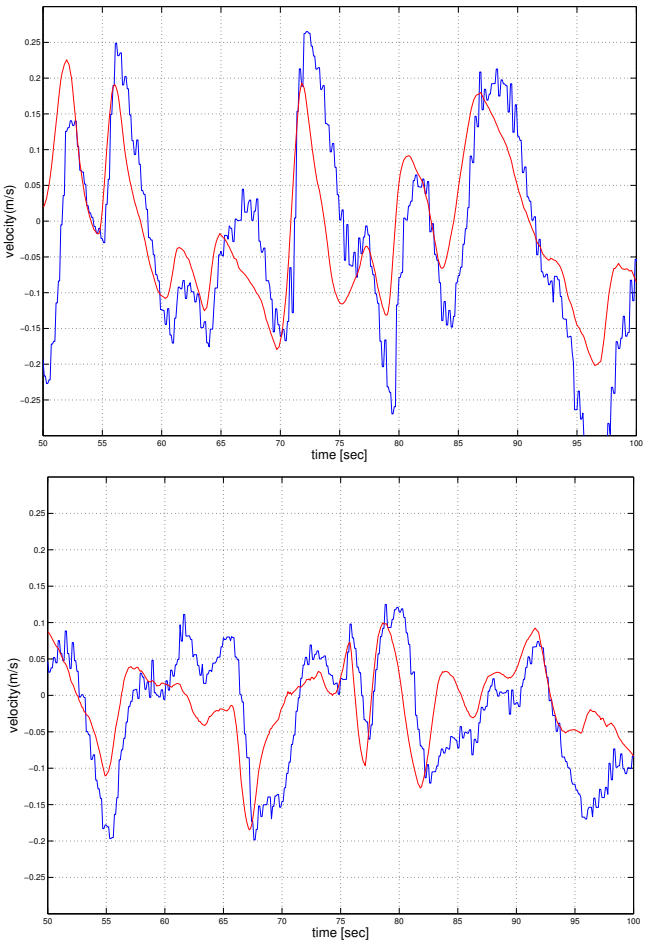
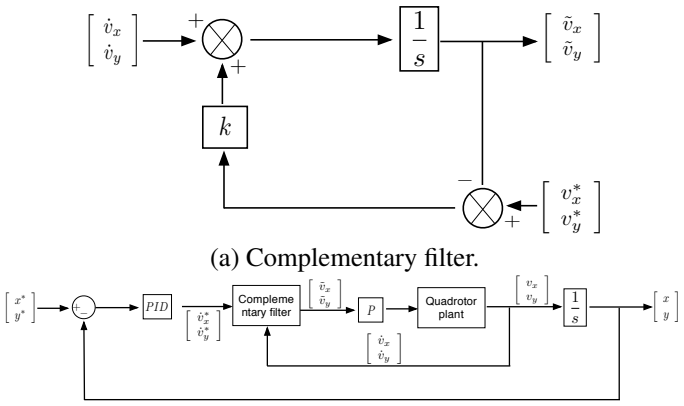


Figure 6: This figure displays the body velocity in x and y axes (top and bottom respectively) calculated from the position data (blue) the complementary filter output (red) for $k = 0.5$. In this figure we could find the red signal leads the blue signal and smoothy filter out the noisy data.

where $\tilde{v}_x$ is obtained from differentiation of the laser-based pose estimate. Since this is computed at a slower rate than $\ddot{x}$ the filter takes the most recent value.

Compared to a Kalman filter the computation is simple and there is only one tuning parameter. Considering the estimator in the frequency domain, k controls the cross-over frequency: below this $\tilde{v}_x$ dominates and above it $\ddot{x}$ dominates. Complementary filters have been used previously for UAV velocity estimation, for example to fuse velocity from low-rate optical flow with high-rate inertial data.

We do not explicitly deal with accelerometer bias in this estimator since the bias is estimated and corrected in the MikroKopter flight controller. We could however deal with it quite simply by introducing an integral term to the complementary filter.



(a) Complementary filter.

(b) Nested control structure.

Figure 7: The gain for the P controller in the velocity loop is 27 and P=0.8, I=0.1 and D=0.7 for the position PID controller.

5.2 Control structure

The block diagram of our nested controller is shown in Figure 7. The inner loop is a velocity controller with proportional and integral control and feedback of estimated velocity from the complementary filter. The outer loop is a position controller with proportional control. This structure is equivalent to a PID controller and quite similar to the backstepping controller of [4], but the nested structure elegantly decouples the different sampling rates of the position sensor and the velocity sensor. The inner loop runs at 20 Hz and the outer loop at 10 Hz.

6 Experiments and results

A number of injuries have been reported around the world by students working in close proximity to quadcopters, so we have worked hard to ensure safe experimental practice. Our laboratory has two floor-to-ceiling glass walls which allows us to keep the quadcopter inside and the pilot/operator outside, see the accompanying video clip.

In our experiment the controller described above maintains position in the horizontal plane while the altitude is, currently crudely, controlled by laser altimetry. With the vehicle at a stable hover about an arbitrary world point denoted $(0,0)$ we use the ROS tool `rviz`, see Figure 8, to change the position setpoint. Figure 9 shows the position response of the vehicle, and Figure 10 shows the velocity response.

In a second experiment we log the estimated position of the vehicle while hovering and this is shown in Figure 11. The vehicle shows good hover performance with standard deviation of 0.13 m and 0.09 m in the x- and y-directions respectively. The random drift is caused by disturbance forces on the vehicle from air circulation and vortices, and room clutter which confounds the scan matcher since the room profile observed by the laser is a complex function of vehicle height and attitude.

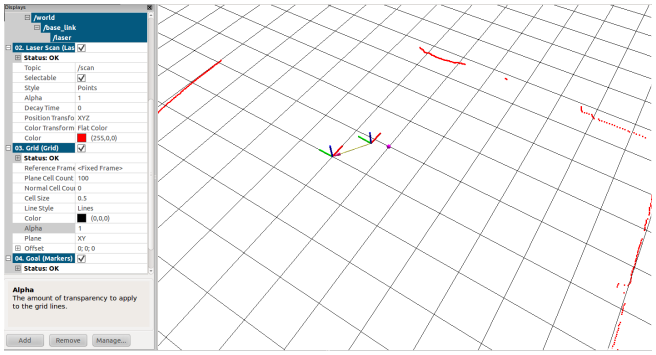


Figure 8: rviz GUI environment on the ROS platform which allows us to easily visualise 3D data such as laser and point clouds. We use rviz to specify the goal point of the quadrotor.

7 Conclusions and future work

In this paper we have described the characteristics of the MikroKopter relevant to horizontal motion, developed a velocity estimator and demonstrated closed-loop position control using a nested control structure. The control and estimation algorithms are computationally cheap and very easy to tune.

We have a large program of ongoing work. Firstly we are currently extending our approach to the z-axis using a complementary filter to combine vertical acceleration, laser altimetry and barometric pressure and a nested controller. Secondly we are exploring changes to the flight control firmware to improve the performance of the attitude control, and to increase the frequency with which inertial state is reported over the serial link. Thirdly we will migrate the pose estimator to the onboard Gumstix processor which eliminates the complexity, limited range and unreliability of the communications link. Finally, we are investigating other extroceptive sensors such as cameras with wide-angle fields of view and the Kinect range camera.

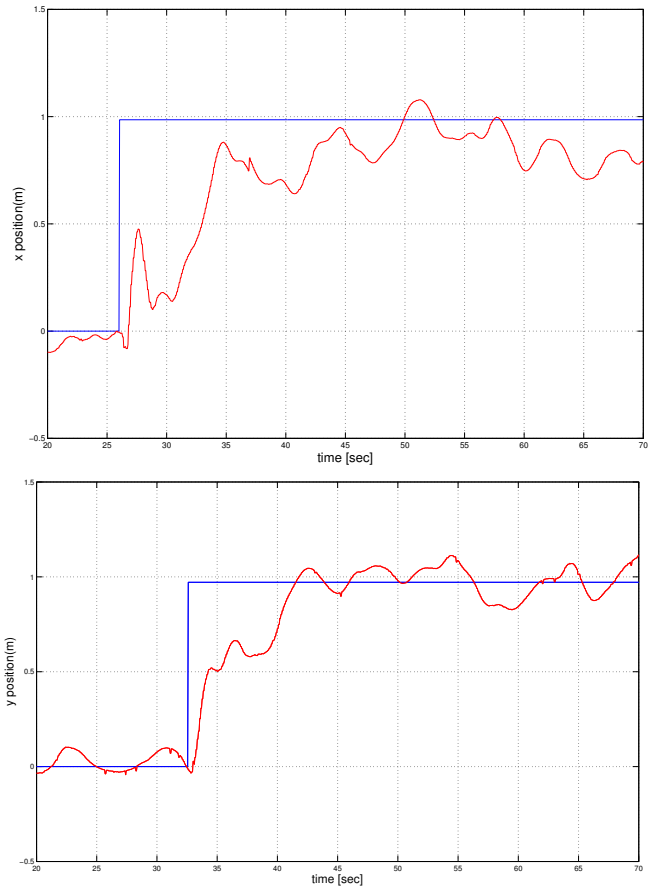


Figure 9: Measured x- and y-position response (top and bottom respectively) to step demand. Blue denotes the demanded position and the red is the measured position using the ICP laser scan matcher.

References

[1] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy. Stereo Vision and Laser Odometry for Autonomous Helicopters in GPS-denied Indoor Environments. In *Proceedings of the SPIE Unmanned Systems Technology XI*, volume 7332, Orlando, F, 2009.

[2] A. Bachrach, A. de Winter, Ruijie He, G. Hemann, S. Prentice, and N. Roy. RANGE - robust autonomous navigation in GPS-denied environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1096 –1097. MIT, may 2010.

[3] S. Bouabdallah, A. Noth, and R. Siegwart. PID vs LQ control techniques applied to an indoor micro quadrotor. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2451 – 2456 vol.3, 2004.

[4] S. Bouabdallah and R. Siegwart. Full control of a quadrotor. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 153–138, 9 2007-nov. 2 2007.

[5] A. Censi. An ICP variant using a point-to-line metric. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 19 –25, may 2008.

[6] Ruijie He, S. Prentice, and N. Roy. Planning in information space for a quadrotor helicopter in a GPS-denied environment. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1814 – 1820, 2008.

[7] Haomiao Huang, G.M. Hoffmann, S.L. Waslander, and C.J. Tomlin. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3277 –3282, 2009.

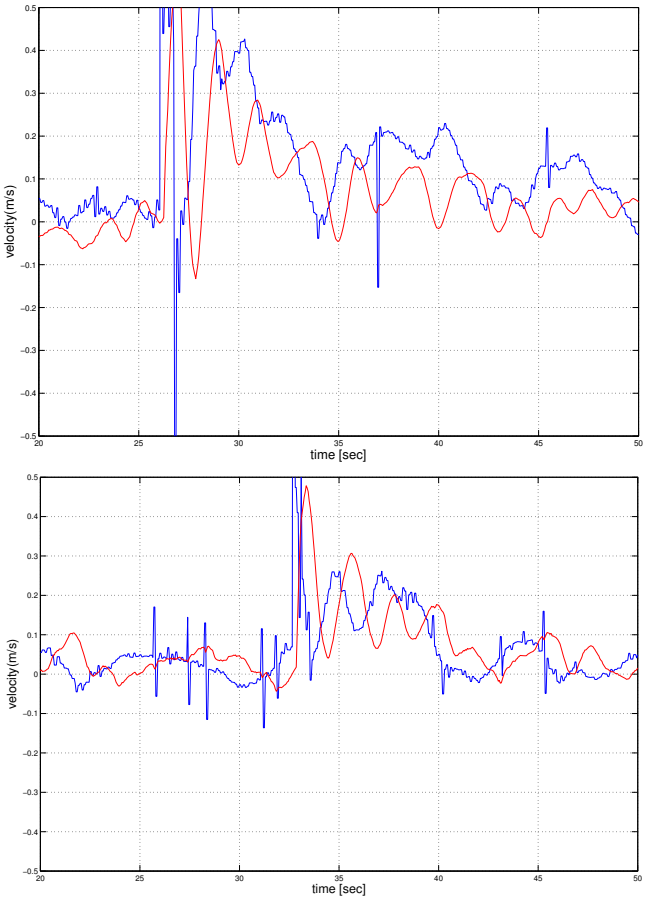


Figure 10: Measured x- and y-velocity response (top and bottom respectively) to step position demand. Blue denotes demanded velocity and the red is the estimated velocity.

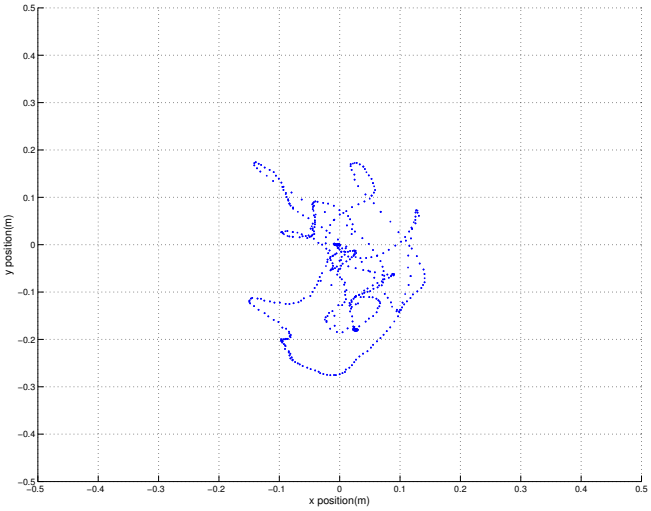


Figure 11: Trajectory of the vehicle centroid for hover at $(0,0)$ with standard deviation of 0.13 m and 0.09 m in the x- and y-directions.

[8] S. Lupashin, A. Schö andllig, M. Sherback, and R. D'Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1642 –1648, May 2010.

[9] Daniel Mellinger, Nathan Michael, and Vijay Kuma. Trajectory generation and control for precise aggressive maneuvers with quadrotors. In *Int. Symposium on Experimental Robotics*, Dec 2010.

[10] Daniel Mellinger, Michael Shomin, and Vijay Kumar. Control of quadrotors for robust perching and landing. In *Int. Powered Lift Conference, Philadelphia*, Oct 2010.

[11] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The GRASP Multiple Micro-UAV Testbed. *Robotics Automation Magazine, IEEE*, 17(3):56–65, Sept. 2010.

[12] Katie Miller. *Path Tracking Control for Quadrotor Helicopters*, July 2008.

[13] P. Pounds and R. Mahony. Design principles of large quadrotors for practical applications. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3265 –3270, 2009.

[14] P.Corke P.Pounds, R.Mahony. Modelling and control of a large quadrotor robot. In *CES*, 2009.

[15] Jonathan M. Roberts, Peter I. Corke, and Gregg Buskey. Low-cost flight control system for a small autonomous helicopter. In *Australasian Conference on Robotics and Automation*, 2002.

[16] Jizhong Xiao William Morris, Ivan Dryanovski. 3D indoor mapping for micro-UAVs using hybrid range finders and multi-volume occupancy grids. In *RSS 2010 Workshop on RGB-D Cameras 2010*, 2010.