Proceedings of Australasian Conference on Robotics and Automation, 3-5 Dec 2012, Victoria University of Wellington, New Zealand.

100Hz Onboard Vision for Quadrotor State Estimation

Inkyu Sa and Peter Corke
Queensland University of Technology, Australia
i.sa@qut.edu.au, peter.corke@qut.edu.au

Abstract

This paper introduces a high-speed, 100Hz, vision-based state estimator that is suitable for quadrotor control in close quarters manoeuvring applications. We describe the hardware and algorithms for estimating the state of the quadrotor. Experimental results for position, velocity and yaw angle estimators are presented and compared with motion capture data. Quantitative performance comparison with state-of-the-art achievements are also presented.

1 INTRODUCTION

The last few years have seen impressive results in quadcopter control using feedback from motion capture systems[Mellinger *et al.*, 2011], [Lupashin *et al.*, 2010]. Motion capture systems are essentially special-purpose high-speed multi-camera vision systems that triangulate the position of retro-reflective markers on the vehicle many hundreds of times per second. High spatial precision provides direct measurement of much of the vehicle's state vector which combined with the high sampling rate allows high-performance quadrotor control. For operation outside a motion capture system the controller generally has to deal with a suite of inferior sensors: altitude from barometric pressure which has low sensitivity and can drift; attitude from gyros and accelerometers which drift; and translational velocity from cameras and/or laser range finders. Much work has therefore focused on sensor fusion techniques[Achtelik *et al.*, 2011],[Bachrach *et al.*, 2010] to optimally estimate the vehicle state as well as sensor offsets and scale factors.

Low-cost high-frame rate cameras are now available and can be combined with small powerful multi-core computers to provide vision system capability that approaches that of a motion capture system. Instead of multiple fixed high-speed cameras observing the robot it is now possible for one or more high-speed cameras on the robot to observe the world. The accuracy of a motion capture system comes from the very wide baseline between cameras and the size of the quadrotor

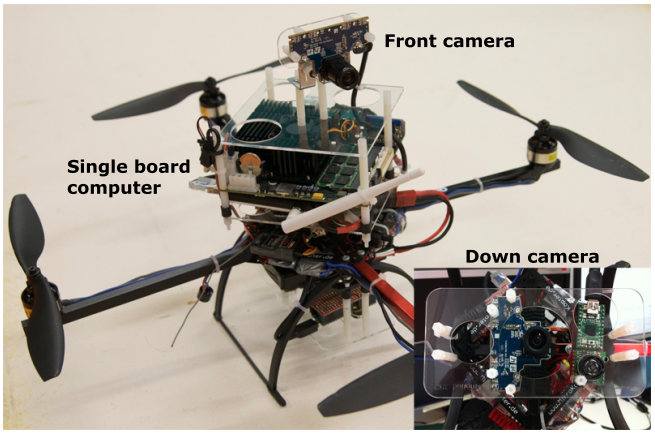
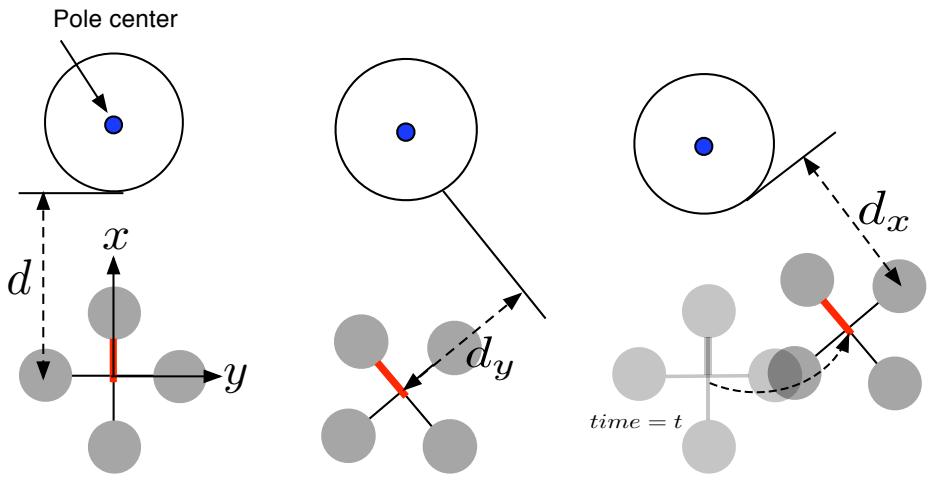


Figure 1: The MikroKopter with 1.9 GHz dual core Celeron and 16 GPUs cores, front and down cameras.

precludes this — multiple cameras, if used, are close to being co-located and this limits the ability to estimate 3D structure unless vehicle translational motion and bundle adjustment methods are used. A motion capture system also simplifies the vision problem, using pulsed infrared illumination and narrow pass band filters on the cameras to greatly simplify the image processing problem, essentially a black scene with several bright objects — a vision system on a quadrotor observes a complex world with clutter and variable lighting conditions.

Our work is concerned with inspection of vertical infrastructure — tall man-made structures such as streetlights, electricity poles or the exterior surfaces of buildings — using a vertical take-off and landing (VTOL) robot platform[Voigt *et al.*, 2011]. We wish to automate the vehicle control and allow a ground-based teleoperator to provide "high level" commands that correspond directly to the important task-domain degrees of freedom such as shown in Figure 2. To achieve this we need to estimate the state of the quadrotor, position and velocity, with respect to these task-space DOF.

This paper presents three contributions. Firstly we show the feasibility of high-speed onboard vision-based state es-



(a) *time* = t (b) *time* = $t+1$ (c) *time* = $t+2$

Figure 3: Changing yaw angle, combined with a planar translation controller that keeps the pole in front of the quadrotor, causes the quadrotor to circle around the pole (red bar indicates the *front* rotor. Reference values for the translation controller are $x^* = d_x$ and $y^* = 0$. (a) initial pose, (b) after (instantaneous) yaw command, (c) after translational controller converges.

timation with a frame rate that approaches that of a motion capture system. Secondly we describe vision algorithms for estimating the states relevant to a pole inspection task: range and bearing of the pole relative to the vehicle and yaw angle in the inertial frame. Finally we evaluate the accuracy and time performance of these algorithms by comparison with motion capture data.

The problem definition is given in Section 1.1 and then the two visual state estimators are described: Section 2 addresses the pole tracker and Section 3 describes the visual compass. We present results in Section 6. Conclusions and future works are presented in Section 7.

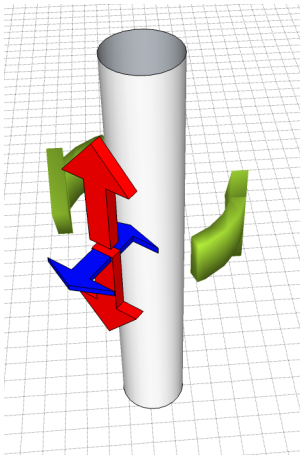


Figure 2: Reduced degrees of freedom (three) required for pole infrastructure inspection task.

1.1 Problem definition

The task-domain degrees of freedom are shown in Figure 2. Rotation around the pole can be simply achieved by yawing the vehicle while maintaining the pole at a fixed bearing angle — the result is motion in a fixed radius orbit around the pole as shown in Figure 3. This requires the ability to measure the bearing angle of the pole with respect to the vehicle and also the absolute yaw angle of the quadrotor. We propose to use vision for both tasks: the bearing angle is estimated by the pole tracker described in Section 2 and the yaw angle by the visual compass described in Section 3.

Motion along a radial line from the pole, toward or away from the pole, is based on feedback of pole distance which is also estimated by the pole tracker.

Currently vertical height is sensed using an ultrasonic rangefinder but for high-altitude outdoor work this would have insufficient range and we propose to use visual odometry from the pole instead — this is an area of future work.

2 Pole tracking

Our pole tracker is based on tracking the two edges of the pole over time. This is an appropriate feature since the pole will dominate the scene in our selected application. There are many reported line extraction algorithms such as Hough transform [Hough , 1959] and other linear feature extractors [Daming *et al.*, 2010] but these methods are unsuitable due to their computational complexity. Instead we use a simple and efficient line tracker inspired by [hager *et al.*, 1998]. The key advantage of this algorithm is its low computation. For 320×240 pixel images every iteration is finished in < 10 ms and only use 55% of the CPU (see Figure 14(b). There are three steps: boot strap, line searching and line model fitting.

We make use of Canny edge detector and Hough transform for boot strap. Once two candidate pole edges are detected (based on length and orientation) a fast tracker takes place of these time consuming algorithms while the vehicle is flying. The near vertical lines are represented in homogeneous form $\ell = (a,b,c)$ such that $\ell(u,v,1)^T = 0$.

The tracker operates on horizontal gradient (Sobel kernel) images. For a number of points (u_l, v_l) vertically distributed along the edge we search along horizontal lines (u, v_l) where $u \in u_l + [-d,d]$ and locate maxima. These maxima are input to a robust line fitting algorithm using RANSAC[Fischler *et al.*, 1981] which updates the model of the line for the next iteration. (see Algorithm 1 and Figure 4).

Algorithm 1: Line tracking algorithm

```
// bootstrap
while number of lines < 2 do
    if (l.length[i] >α) and (|l.angle[i]-90| < γ) then
        l[i] is a pole edge;
        initialise model ℓ_i = (a_i, b_i, c_i)
    else
        i=i+1;
    end
end
//tracker
while tracking lines continuously do
    I = next frame
    I_x = horizontal Sobel derivative
    for i = 1 to 2 do
        for v_l = 10 to 240 step 20 do
            u_l = (−b_i v_l − c_i)/a_i
            (ū_k, v̄_k) is maximum of pixels along line
            (u_l − d, v_l) to (u_l + d, v_l)
        end
        ℓ_i = (a_i, b_i, c_i) = RANSAC fit of points (ū_k, v̄_k)
    end
    estimate pole range and bearing from ℓ_1 and ℓ_2
end
```

Note that constant parameters $\alpha = 200$ and $\gamma = 5$.

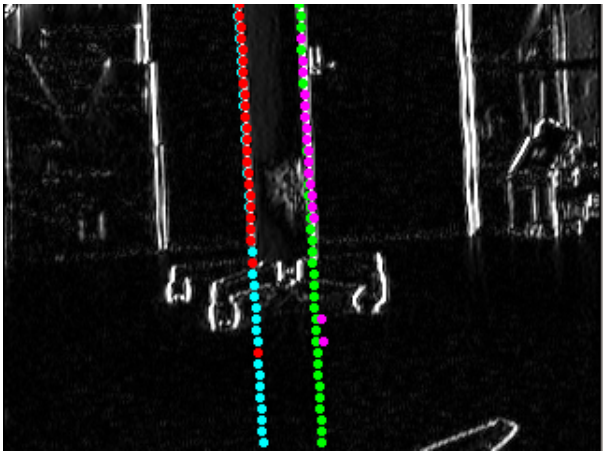
To convert detected lines to range and bearing we assume a simple pinhole camera model. Given D, f, I_w and I_h, we can estimate the distance between a image plane and a object as

$$Z = \frac{D \cdot f}{I_h} \tag{1}$$

where Z is the distance between an object and an image plane (metre), D = size of an object (metre) in this case the known diameter of the pole, f = focal length (metre), I_h = size of an object on a image plane (metre) and is based on the horizontal distance between the edges and the known pixel size. The focal length of Playstation EyeToy(normal mode) is 3.204 mm (determined by calibration) and the pixel dimension is 12 µm for 320×240 resolution. Note that when the camera is close

(a) An image (320×240) of the pole from the front high speed camera.

(b) Boot strapping phase: vertical yellow lines indicated the detected pole edges.



(c) A line tracker detects inliers (red and magenta dots) and creates the best line model for each vertical line then updates the next searching line model(cyan and green).

Figure 4: A tracker processing sequences. (a) Grab an image → (b) Boot strapping→ (c) Track lines

to the pole the diameter will be underestimated since the projected edge points correspond to tangents of the pole.

The distance between the lines is noisy due to pixel quantisation and somewhat insensitive to distance. Currently we apply a first-order Kalman filter to I_h to smooth the signal. The problem is complicated by the soft edges of the pole — the intensity falls off gradually due to the curvature of the pole. We calculate a bearing angle:

$$\alpha = \arctan(\frac{I_y}{f}) \tag{2}$$

$$I_y = \frac{f \cdot Y}{Z} \tag{3}$$

where I_y is the distance from the image center to the pole center on the image plane (metre), Y is the distance from the image center to the center of pole in world coordinate (metre). The overall time profiling and total elapsed average time is presented in Figure14(a).

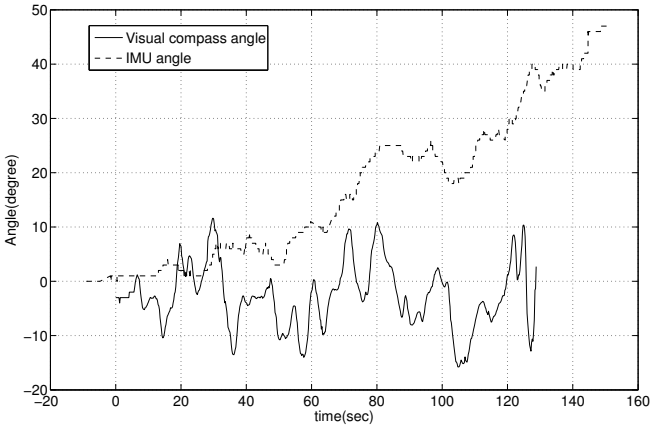


Figure 5: Yaw angle drift over time. The dotted line is yaw angle estimated from IMU and the solid line is the output of the visual compass. This result is obtained while hovering under manual pilot control; taking off and landing with as similar angle as possible.

3 Visual compass

The quadrotor's inbuilt yaw angle estimator integrates the yaw rate gyro which results in considerable errors due to drift, even over quite short time intervals as shown in Figure 5. A magnetic compass is drift free but very prone to disturbances, particularly from the magnetic fields of the quadrotor's high current motors. [Scaramuzza *et al.*, 2009],[Scaramuzza *et al.*, 2008] presented feature based approaches to estimate rotation omnidirectional images on a car like vehicle, but hey assume an omnidirectional camera with optical axis normal to the ground and assume a non-holonomic constrained vehicle. [Antone *et al.*, 2000], [Bazin *et al.*, 2012] proposed a vanishing point based rotation estimation approach, but such geometrical characteristics may not be present in our application environment. We used instead a down-looking light-weight camera to estimate yaw angle[Meier *et al.*, 2011]. There are four steps to estimate yaw angle:

1. Feature detection: Compute key-points that include image location and descriptors using GPU.
2. Matching: For each keypoint find the closest corresponding point in descriptor space, $\mathbf{M}_t$.
3. Estimate homography matrix, $\mathbf{H}_t$ using RANSAC to determine inliers and outliers from $\mathbf{M}_t$. We use the OpenCV findingHomography function.
4. Yaw angle calculation: Estimate incremental yaw angle from elements ($\mathbf{H}_{t_{(0,1)}}$ and $\mathbf{H}_{t_{(1,0)}}$) of the homography and accumulate to calculate total rotation.

There are however issues of false matching at take off which introduce large estimation error and we use an initial value from IMU once for a boot strap in the air. This integrative process, step 4 above, is also subject to drift but the rate is far lower than the IMU and could be reduced further by a key framing technique. The computationally expensive feature detection and matching uses the GPU and rest of steps run on the CPU. The overall time profiling can be found Figure 13.

4 Overall system architecture

The quadrotor is equipped with a single board computer with a dual core 1.9 GHz Celeron CPU and 16 GPU cores, runs Ubuntu Linux and ROS[1], weighs 314 g and consumes 20 W. Two Playstation EyeToy (100 Hz RGB at 320×240) are attached: one in the forward direction for pole detection and one facing downward for the visual compass. Each camera occupies one quarter of the bandwidth of a USB2.0 port. For height measurement, a cost effective sonar sensor and an 8-bit microprocessor is exploited with a moving median filter. A 5.0 Ah 4 cell Li-Po battery provides up to 16 minutes flight time.

The software is based on ROS and the architecture is shown in Figure 6 where blue boxes denote the ROS nodes which are individual processes. The SBC runs the ROS USB camera driver for two high speed cameras. The line tracker publishes the topic */pose2D* to the linear Kalman filter node for smoothing data. Different sampling rate and type of sensor data are fused in a complementary filter and publishes 100 Hz horizontal velocity. All these procedures are completed in less than 10 ms and other processes such as the visual compass, sonar height estimator, control and telemetry can also be accomodated. The ROS serial node communicates with the MikroKopter flight control board over USB. Every 16 ms it receives a `DebugOut` packet and the inertial data (converted to SI units and the our coordinate frame) is published as the */mikoImu* topic. These topics can be recorded in a log file (ROS bag format) and later replayed (using `rosbag`) to test the state estimator and controller offline with image and inertial data. All software ROS packages are available on online[2].

5 Velocity estimation

Quadrotors are force actuated vehicles and have very little natural damping, though blade flapping does add some damping[Dinuka *et al.*, 2011]. To ensure stability in flight a controller requires a high-quality velocity estimate: smooth, high update rate with low-latency. Differentiation of the position from the line tracker pose estimator results in velocity at 100 Hz with a latency of 10 ms. Raw high update rate position data is noisy due to limited resolution of camera. Small changes in a low resolution image plane introduce significant changes in distance and bearing measurement. Therefore we use the MikroKopter acceleration measurements which we read at 60 Hz with low latency and fuse them to create a velocity estimate. We estimate accelaration in the frame $\{Q\}$

[1] http://ros.org

[2] http://www.ros.org/wiki/MikroKopter/

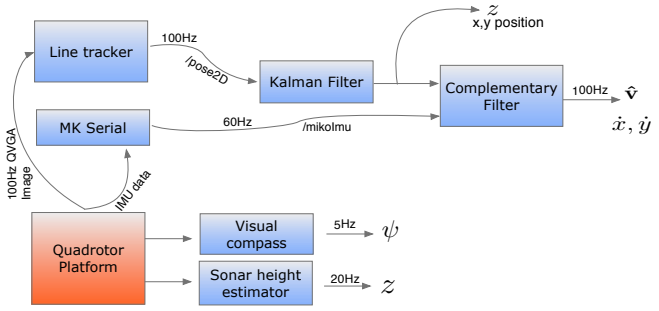


Figure 6: Software implementation using ROS platform where blue boxes represent ROS nodes and the orange box is a single board computer on a quadrotor. The prefix '/' denotes a ROS topic. Note that different sampling rate sensors are fused in the complementary filter.

which is centred on the vehicle but parallel to the inertial frame

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

where a_x, a_y and the measured acceleration from the flight control board and θ, ϕ denote the pitch and roll angles respectively. Accelerometer bias is estimated and corrected in the MikroKopter flight controller. We fuse asynchronous acceleration and differentiated position (from the pole tracker) using a discrete-time complementary filter[Jonathan *et al.*, 2002]:

$$\hat{v_x}(t+1) = \hat{v_x}(t) + {}^{Q}\ddot{x}(t) + K(\tilde{v}_x(t) - \hat{v}_x(t))\Delta_t \qquad (4)$$

where $\tilde{v}_x$ is obtained from differentiation of the line tracker pose estimate. Compared to other sensor fusing filter approaches the computation is simple and there is only one tuning parameter, K. 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[Corke , 2004].

6 Experiments and results

In this section we compare the performance of the onboard 100 Hz state estimator with data recorded by a motion capture system. The diameter of the pole was assumed to be known and the camera intrinsic parameters had been estimated by a calibration process A summary of the RMS error for the state variables is given in Table 1. All results were obtained while in manual pilot mode. The video demonstration is available on online.[3]

6.1 Estimator accuracy

The performance of the pole tracker is shown in Figure 7 where it is compared with ground truth. Standard deviation

[3] http://www.youtube.com/watch?v=ZzfOHky9DYo

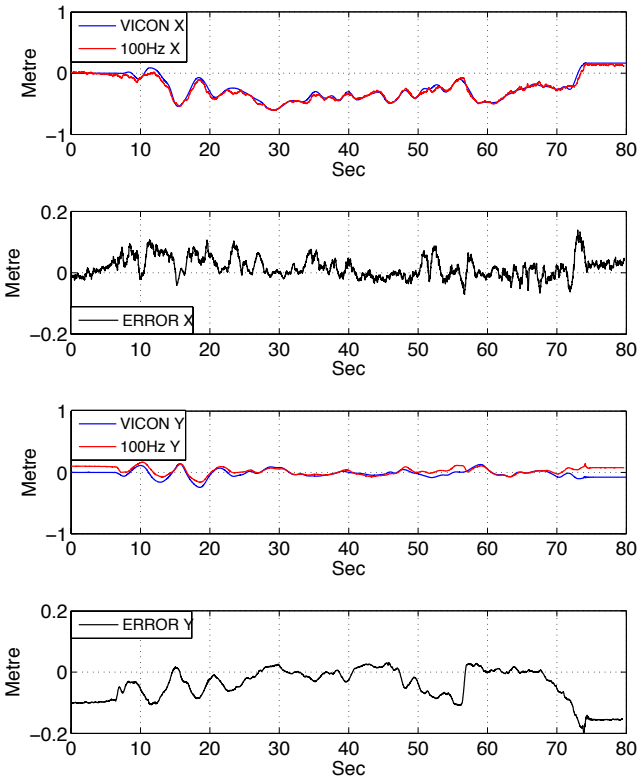


Figure 7: x, camera to a pole, and y, left and right, position estimation results from the forward looking high speed camera. The first and third rows represent x and y position estimation(red) and ground truth from a motion capture device(blue). The second and fourth rows are error between ground truth and estimate. Standard deviation for x and y are σ_x = 0.033 m and σ_y =0.0534 m.

for x and y, relative pole position, are σ_x = 0.033 m and σ_y = 0.0534 m. The performance of the proposed system is compared with other work reported results in Figure 8.

The performance of the velocity estimator is shown in Figure 9 where it is compared with ground truth. The complementary filter gain was $K = 0.9$. Standard deviation for v_x and v_y are σ_{v_x} = 0.0559 m/s and σ_{v_y} = 0.0413 m/s. The performance is compared with other state-of-the-art works in Figure 10.

Table 1: Summary of results

State variable	RMS	units
x	0.033	m
y	0.0534	m
z	0.012	m
ψ	1.425	deg
v_x	0.0559	m/s
v_y	0.0413	m/s

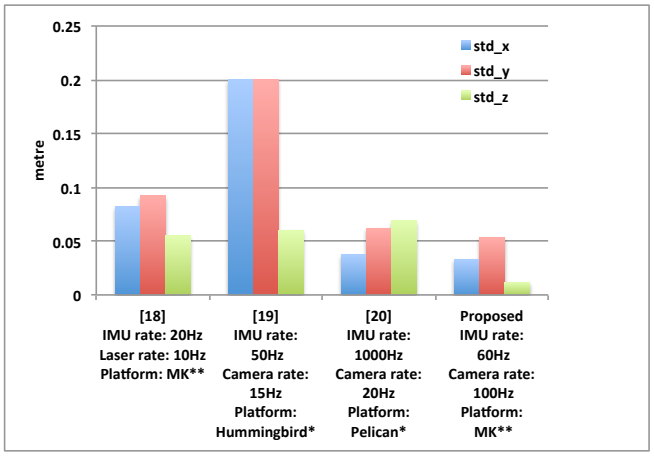


Figure 8: Position estimation performance comparing the proposed system with the state-of-the-art achievementsNote that the data is the standard deviations and is collected from the cited papers and a lower value denotes a better performance. * Ascending Technologies, ** MikroKopter.

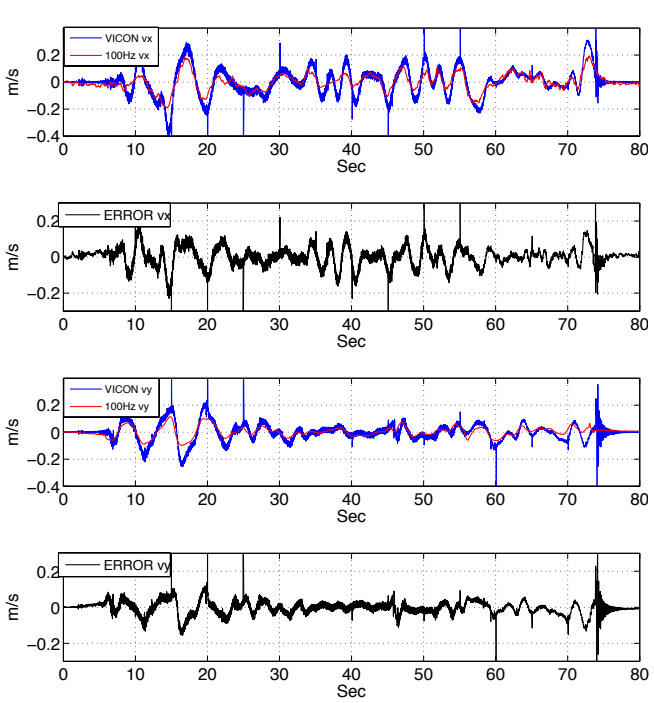


Figure 9: x and y velocity estimation results from the forward looking high speed camera. The first and third rows represent x and y velocity estimation(red) and ground truth from a motion capture device(blue). The second and fourth rows are error between ground truth and estimate. Standard deviation for v_x and v_y are $\sigma_{v_x} = 0.0559\,\mathrm{m/s}$ and $\sigma_{v_y} = 0.0413\,\mathrm{m/s}$. Note that peak noise of ground truth arises from the position derivative.

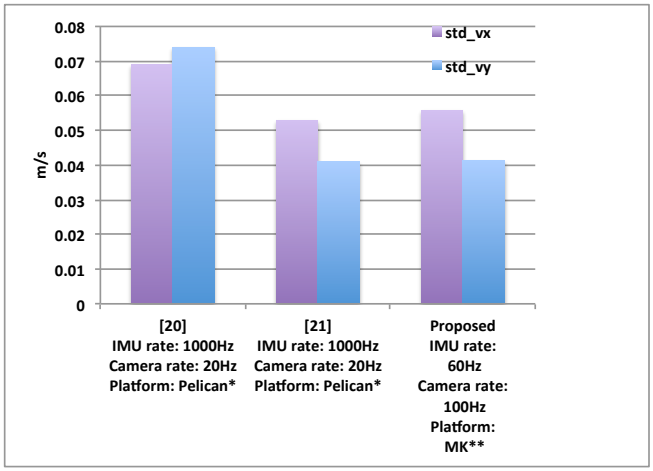


Figure 10: Velocity estimation performance comparing the proposed system with the state-of-the-art achievementsNote that the data is the standard deviations and is collected from the cited papers and a lower value denotes a better performance. * Ascending Technologies, ** MikroKopter.

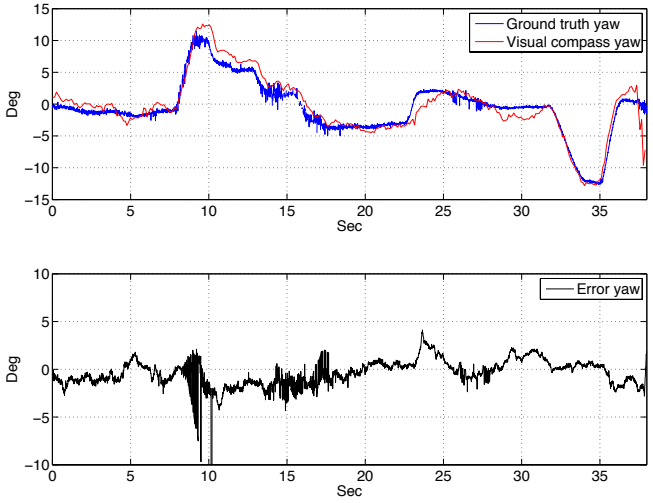


Figure 11: (Top) Yaw angle estimation of the proposed visual compass (red) and ground truth from a motion capture device (blue). (Bottom) error between ground truth and estimated angle. Note that there are changes in rotation at 10 s and 24s and these introduce error in yaw angle estimation. Standard deviation of error is $\sigma_\psi = 1.425^\circ$

The performance of the yaw estimator is shown in Figure 12 and has a standard deviation of $\sigma_\psi = 1.425^\circ$. We placed objects on the ground (our lab floor is quite textureless) and the average number of features is ≈ 250.

Altitude estimation is important for a vertical take off and landing vehicle. Currently we are using ultrasonic ranging, not vision, for this degree of freedom and the performance against ground truth is shown in Figure 12. The height mea-

gpuMatching	1.71	2.74	11.68	12.62
FindingHomography	3.08	3.53	2.68	4
Total	26.74	28.28	38.22	41.59

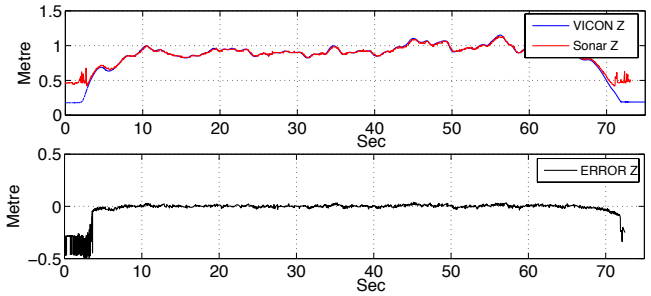


Figure 12: Altitude estimation results from the sonar sensor. Top shows ground truth (blue) and estimation (red). Bottom is error between two measurement. Note that before taking off the noise on the sonar sensor is due to a minimum reading range of the sonar sensor, 0.15 m. Standard deviation of error is $\sigma_z = 0.012$ m

surement is compensated with roll and pitch angle and has a standard deviation of $\sigma_z = 0.012$ m when the vehicle is above 0.5 m. However for outdoor applications at heights over 5 m this sensor will fail and we will replace it with a vertical visual odometer.

6.2 Time performance

The overall CPU load for our visual state estimator is shown in Figure 14. The line tracker takes 55% and the visual compass 20%. Although the line tracker is fast, around 3 ms per frame, it runs at a very high frame rate.

We also benchmarked the feature detectors used for the visual compass in terms of processing speed, CPU and GPU usage and number of features. [Furgale and Barfoot , 2010] report 40% faster execution time for a GPU compared to OpenCV[4] but we observe high initial costs of the GPU-based SURF feature detector, which we are investigating.

Figure 13 shows time profiling of the proposed visual compass. and overall processing time 40 ms for one iteration with a 320×240 image.

7 Conclusions and future work

We have shown the feasibility of high-speed onboard vision-based state estimation with a frame rate that breaks the 100 Hz barrier and starts to approach that of a motion capture system. We have exploited commodity cameras and computers and while the latter is currently larger and heavier than we would like the trend continues toward smaller, cheaper and more powerful computers.

The use of high-speed vision promises some important benefits. From a vision standpoint the scene change from frame to frame is small which means that simple vision features (with poor invariance properties) should work well. From a control standpoint a high frame sample rate, well

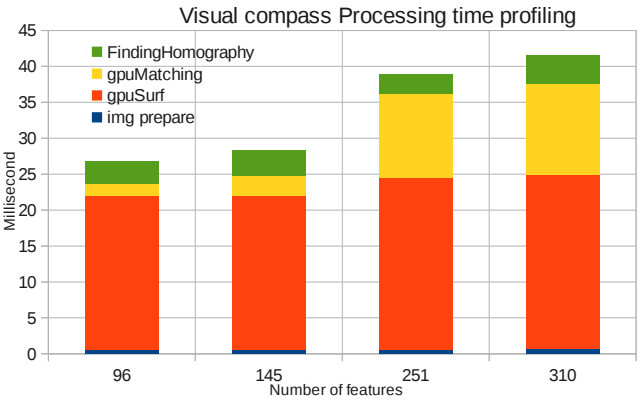


Figure 13: This graph shows the visual compass timing profiling with different number of features. We use around 200 features for matching with 1 pixel RANSAC threshold and 5 pixel perspective transformation threshold filtering. Note that GPU based SURF detector shows the similar elapsed time with different number of features.

above the dynamics of the vehicle, allows us to consider the plant as a continuous time system — though the latency of the sensor, USB communications and computing pipeline is yet to be determined.

The key elements of our system are a line tracker that uses model information to infer distance and bearing angle, and a feature-based visual compass based on down looking imagery. Both vision-based estimators demonstrate ground-truthed levels of position and velocity accuracy comparable with other systems reported in the literature.

There are many areas for future work. Our line tracker is very well suited to parallel execution and could be ported to the GPU. It could also benefit from the use of IMU data to better predict the location of the line in the scene and the use of subpixel techniques to improve the precision of the line model. Distance sensitivity from apparent width is fundamentally limited and could be improved by using short baseline stereo or structured light. The SURF feature detector, even on a GPU, is quite expensive and we will investigate FAST/BRIEF features for this purpose, and also investigate key framing in the visual compass to reduce drift even further. Vision-based altitude estimation, currently ultrasonic, will be implemented as either vertical visual odometry or feature contraction in the downward camera, or a fusion of both. All software ROS packages and comprehensive tutorial are available on online[5].

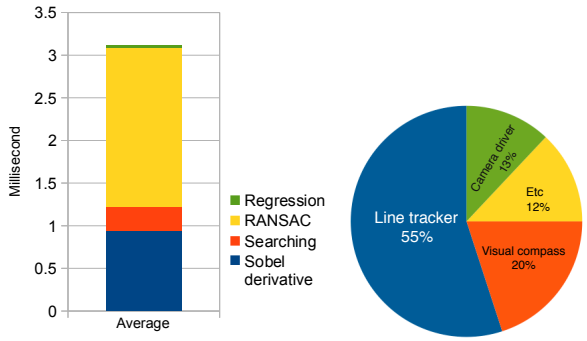
Acknowledgments

We would like to thank Timothy Gurnett for assistance while using the VICON system at the QUT Cube Lab.

[4] http://opencv.willowgarage.com/wiki/

[5] http://www.ros.org/wiki/MikroKopter/Tutorials

Line tracker Processing time profiling



(a) The line tracker timing profiling.

(b) The total CPU usage while running all algorithms on onboard.

Figure 14: (a) The total average time is 3.11 ms. We measure elapsed time by marking time stamps at a starting point and an end point of algorithms. (b)The line tracker uses 55% due to the high update rate. The sonar sensor altitude estimator, the complimentary velocity estimator and the serial interface driver with the MikroKopter flight control board use 12% of CPU. We measure this CPU usage with the top command.

References

[Mellinger *et al.*, 2011] Daniel Mellinger and Vijay Kumar. *Minimum Snap Trajectory Generation and Control for Quadrotors*. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)

[Lupashin *et al.*, 2010] Lupashin, S. and Schöandllig, A. and Sherback, M. and D'Andrea, R. *A simple learning strategy for high-speed quadrocopter multi-flips*. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)

[Achtelik *et al.*, 2011] Markus Achtelik and Stephan Weiss and Roland Siegwar *Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments*. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)

[Bachrach *et al.*, 2010] Bachrach, A. and de Winter, A. and Ruijie He and Hemann, G. and Prentice, S. and Roy, N. *RANGE - robust autonomous navigation in GPS-denied environments*. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)

[Voigt *et al.*, 2011] Voigt, Rainer and Nikolic, Janosch and Hurzeler, Christoph and Weiss, Stephan and Kneip, Laurent and Siegwart, Roland. *Robust embedded egomotion estimation*. Proceedings of IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)

[Baeten *et al.*, 2002] Baeten, J. and Bruyninckx, H. and De Schutter, J. Shared control in hybrid vision/force robotic servoing using the task frame. Proceedings of IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)

[Hough , 1959] Hough Machine Analysis of Bubble Chamber Pictures. Proceedings of the International Conference on High Energy Accelerators and Instrumentation

[Daming *et al.*, 2010] Daming Shi and Liying Zheng and Jigang Liu. Advanced Hough Transform Using A Multilayer Fractional Fourier Method. Image Processing, IEEE Transactions on

[hager *et al.*, 1998] Hager, G.D. and Toyama, K. X vision: A portable substrate for real-time vision applications. Computer Vision and Image Understanding

[Fischler *et al.*, 1981] MA Fischler and RC Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM

[Scaramuzza *et al.*, 2009] Scaramuzza, Davide and Fraundorfer, Friedrich and Siegwart, Roland Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)

[Scaramuzza *et al.*, 2008] Scaramuzza, D. and Siegwart, R. Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles. Robotics, IEEE Transactions on

[Antone *et al.*, 2000] Antone, M.E. and Teller, S. Automatic recovery of relative camera rotations for urban scenes . Proceedings on Computer Vision and Pattern Recognition(CVPR)

[Bazin *et al.*, 2012] Jean Charles Bazin and Cdric Demonceaux and Pascal Vasseur and Inso Kweon Rotation estimation and vanishing point extraction by omnidirectional vision in urban environment. The International Journal of Robotics Research

[Meier *et al.*, 2011] Lorenz Meier and Petri Tanskanen and Friedrich Fraundorfer and Marc Pollefeys PIXHAWK: A System for Autonomous Flight using Onboard Computer. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)

[Dinuka *et al.*, 2011] Dinuka Abeywardena and Sarath Kodagoda and Rohan Munasinghe and Gamini Dissanayake A Virtual Odometry for a Quadrotor Micro Aerial Vehicle. In Proceedings of the Australasian Conference on Robotics and Automation (ACRA)

[Jonathan *et al.*, 2002] Jonathan M. Roberts and Peter I. Corke and Gregg Buskey Low-Cost Flight Control System for a Small Autonomous Helicopter. In Proceedings of the Australasian Conference on Robotics and Automation (ACRA)

[Corke , 2004] Corke, Peter An Inertial and Visual Sensing System for a Small Autonomous Helicopter. Journal of Robotic Systems

[Bourgeois , 2009] Frederic Bourgeois Delay and Dropout Tolerant State Estimation for MAVs. Master thesis,(Supplimentary maetrial)

[Inkyu *et al.*, 2012] Inkyu Sa and Peter Corke System Identification, Estimation and Control for a Cost Effective Open-Source Quadcopter. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)

[Weiss *et al.*, 2011] Weiss, S. and Siegwart, R. Real-time metric state estimation for modular vision-inertial systems. Robotics and Automation (ICRA), 2011 IEEE International Conference on

[Weiss *et al.*, 2012] S Weiss and M Achtelik and S Lynen and M Chli and R Siegwart Real-time Onboard Visual-Inertial State Estimation and Self-Calibration of MAVs in Unknown Environments. Proc. of the IEEE International Conference on Robotics and Automation (ICRA)

[Furgale and Barfoot , 2010] P T Furgale and T D Barfoot Visual Path Following on a Manifold in Unstructured Three-Dimensional Terrain. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)