2014 International Conference on Unmanned Aircraft Systems (ICUAS)
May 27-30, 2014. Orlando, FL, USA

Close-quarters Quadrotor flying for a pole inspection with Position Based Visual Servoing and High-Speed Vision

Inkyu Sa and Peter Corke

***Abstract*— This paper presents a 100 Hz monocular position based visual servoing system to control a quadrotor flying in close proximity to vertical structures approximating a narrow, locally linear shape. Assuming the object boundaries are represented by parallel vertical lines in the image, detection and tracking is achieved using Plücker line representation and a line tracker. The visual information is fused with IMU data in an EKF framework to provide fast and accurate state estimation. A nested control design provides position and velocity control with respect to the object. Our approach is aimed at high performance on-board control for applications allowing only small error margins and without a motion capture system, as required for real world infrastructure inspection. Simulated and ground-truthed experimental results are presented.**

I. INTRODUCTION

Our work is motivated by the problem of inspecting vertical infrastructure such as street light or electrical poles. There are more than 175 million street lights in the world which need to be inspected periodically[1]. The options for inspecting locations above the ground are quite limited, and all are currently cumbersome. Ladders can be used up to a height of 5 $\sim$ 7 meters but are quite dangerous. Each year in the United States more than 160 people are killed in ladder accidents and 242,000 injured[2]. Cherry pickers can be used but vehicle access is required and the setup time is significant. Beyond that height, a person either climbs up the structure or rappels down from the top, both of which are slow and hazardous.

A variety of climbing robots have been proposed for man-made vertical infrastructure. Typically, these robots mimic reptiles [1], insects [2] or mammals [3], and they must physically contact the surface. This requires complex mechanical design and detailed dynamic analysis.

Inspection from manned rotorcraft is possible but is expensive and only suitable in non-urban environments. In recent years, we have seen significant technology advances in small unmanned VTOL platforms, in particular multi-rotors. These have been made possible by advances in power electronics, MEMS sensors, and micro controllers. These systems are cost efficient, have sufficient payload and ample endurance to perform useful tasks, including vertical infrastructure inspection. In addition they are low-weight, which reduces the hazard inherent in their operation.

However, controlling such vehicles requires considerable operation skill which reduces the general applicability of this technology. A skilled operator is continually estimating the pose of the vehicle with respect to the goal and adjusting the joystick accordingly. The precision of this falls off with distance and is impossible for applications that require measurement beyond line of sight.

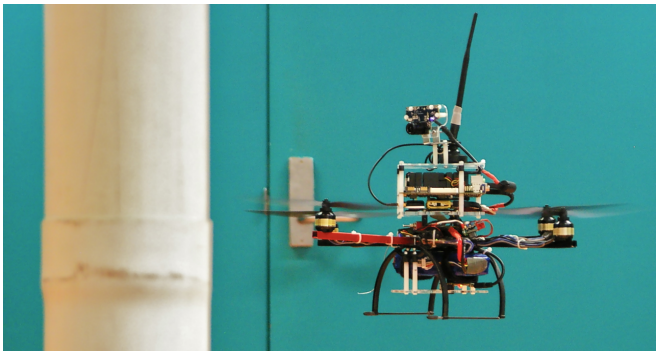


Fig. 1. The Cyphy Lab quadrotor research platform (MikroKopter) visual servoing with respect to a vertical structure. The vehicle carries a front facing 100 Hz camera and all processing is performed on board.

In this paper, we describe a high performance quadrotor visual servoing system that uses a 100 Hz front-facing monocular camera and 70 Hz inertial data to reliably stabilize a quadrotor close to a cylinder-shaped vertical structure. We assume the diameter of the cylindrical structure is given. Major differences to our previous work [4] and contributions of this paper are:

- Presenting 100 Hz position based visual servoing which makes use of Extended Kalman Filter (EKF) framework as an estimator.
- Introducing Plücker line representation in order to describe the geometric relationship between a structure and a camera.
- Design a nested control system and simulation.
- Demonstrating impact of 100 Hz processing by comparing its output with different sampling rates.
- Onboard processing, sharing source code and technical documents to community.

`https://code.google.com/p/cyphy-pole-quad`

This paper is structured as follows: Section II introduces related work and background. Section III describes image processing and camera system. Section IV and V address estimation and controller design. We present our experimental results in Section VI and conclusions in Section VII.

The authors are with the CyPhy Laboratory, School of Electrical Engineering and Computer Science, Queensland University of Technology, Australia. i.sa@qut.edu.au, peter.corke@qut.edu.au

[1]2007 Echelon, Monitored Outdoor Lighting

[2]2011 National Electronic Injury Surveillance System Data Highlights

II. Related Work/Background

Autonomous robot platforms require estimation of their state in order to navigate. GPS guided multi-rotor systems have been demonstrated for outdoor navigation and shown stable performance [5]. However, GPS is unavailable in indoor, underground, and in urban environments due to multipathing and signal attenuation. Significant research has thus been devoted to developing systems that can operate in GPS-denied environments. For example, Mellinger et al. [6] and Lupashin et al. [7] have demonstrated impressive control performance within a motion capture system. This provides accurate high-speed estimates of vehicle state, but operation is limited to the coverage area of the capture system.

If flying outside a motion capture system, a controller has to deal with a suite of less accurate sensors, many of which are subject to drift. Much research has been directed toward optimal estimation of vehicle state from multiple sensors; sensor fusion. Bachrach et al. [8] and Weiss et al. [9] presented systems that estimate state by utilizing a 40 Hz laser range finder, or a 10 Hz monocular camera in an EKF framework. Grzonka et al. [10] proposed an autonomous flying robot system based on Bayesian Simultaneous Localization and Mapping (SLAM) techniques. Laser data was transmitted to a ground station at 10 Hz and the vehicle was localized with respect to generated maps. We aim to fly without a motion capture system using fast sensing techniques (100 Hz) for measurement.

Fast sensing techniques in light-weight aerial robotics are gaining momentum with considerable growth in sensor and integrated circuit technology. This fast update rate allows quick response to disturbances and yields robust smooth maneuvers. Recent results have demonstrated the advantages of using these fast sample rate sensors. Curler et al. [11] presented preliminary results toward multiple flying vehicles that utilize structured infrared LEDs and a 100 Hz infrared camera for state estimation and control. This infrared camera is able to return the sizes and pixel centroid locations of point features. In our approach, we utilize an ordinary camera and extract natural salient line features using image processing techniques.

We use Plücker line representation to describe the target since it has useful linear projection characteristics. This line representation has been used previously for SLAM systems [12] and 3D line reconstruction application [13] for ground robots in static environments and for image-based visual servoing on a quadrotor system [14].

III. Image Processing/Camera System

In this section, we define our coordinate systems and introduce the Plücker line representation of the target object. Details of the line detection and tracking system are then presented.

A. Coordinate Systems Definition

We define three right-handed frames: world $\{\mathcal{W}\}$, body $\{\mathcal{B}\}$ and camera $\{\mathcal{C}\}$ which are shown in Fig 2. Note that both $\{\mathcal{W}\}$ and $\{\mathcal{B}\}$ have their z-axis downward while $\{\mathcal{C}\}$ has its z-axis (camera optical axis) in the horizontal plane of the propellers. We define the notation ${}^{a}\mathbf{R}_{b}$ which rotates a vector defined with respect to $\{b\}$ to a vector with respect to $\{a\}$.

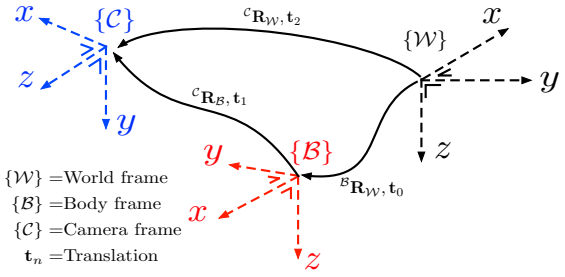


Fig. 2. Coordinate systems definition. Transformation between $\{\mathcal{B}\}$ and $\{\mathcal{C}\}$ is constant whereas $\{\mathcal{W}\}$ and $\{\mathcal{B}\}$ vary as the quadrotor moves. ${}^{\mathcal{C}}\mathbf{R}_{\mathcal{B}}$ rotates a vector defined with respect to $\{\mathcal{B}\}$ to a vector with respect to $\{\mathcal{C}\}$

The camera attitude can thus be expressed as a series of rotation matrix multiplications such that

$$ {}^{\mathcal{W}}\mathbf{R}_{\mathcal{C}} = {}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}{}^{\mathcal{B}}\mathbf{R}_{\mathcal{C}}; \quad {}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}} = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi) \tag{1} $$

and ϕ, θ and ψ denote roll, pitch and yaw angle. They are defined as rotations about the $\{\mathcal{W}\}$ z, y, and x-axes respectively. The camera is mounted on the vehicle with its optical axis in the horizontal plane, such that

$$ {}^{\mathcal{B}}\mathbf{R}_{\mathcal{C}} = \mathbf{R}_y(\pi/2)\mathbf{R}_z(\pi/2) \tag{2} $$

which we combine to write

$$ {}^{\mathcal{W}}\mathbf{R}_{\mathcal{C}} = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi)\mathbf{R}_y(\pi/2)\mathbf{R}_z(\pi/2) \tag{3} $$

The angular rates are related to angular velocity by

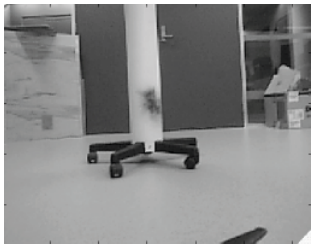
$$ \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} {}^{\mathcal{B}}\omega_x \\ {}^{\mathcal{B}}\omega_y \\ {}^{\mathcal{B}}\omega_z \end{bmatrix} \tag{4} $$

where ${}^{\mathcal{B}}\omega_x$, ${}^{\mathcal{B}}\omega_y$ and ${}^{\mathcal{B}}\omega_z$ are the angular velocity components about the x,y and z axes of $\{\mathcal{B}\}$.

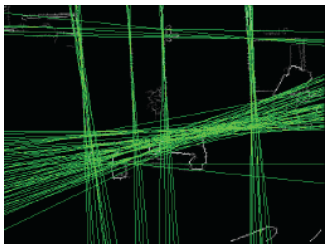
B. Line Representation

A quadrotor must tilt into the direction it wants to move, and this causes line features to move in the image. In order to describe this relationship, it is required to use a 3D line representation. A 3D line can be described using various parameterizations including two 3D points, intersection of two 3D planes, closest point with direction, or two projections. These representations vary in terms of their properties including completeness, reprojection characteristics with a perspective camera, and the number of internal constraints [15]. *Plücker coordinates* [16] have been widely used in the computer vision and the robotic community for 3D line reconstruction [13], line based visual servoing [14], and SLAM [12]. *Plücker coordinates* describe a line joining the two 3D points ${}^{\mathcal{W}}\mathbf{A}$ and ${}^{\mathcal{W}}\mathbf{B} \in \mathbb{R}^3$ in the world frame according to

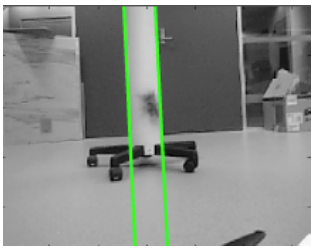
$$ {}^{\mathcal{W}}\mathbf{L} = {}^{\mathcal{W}}\tilde{\mathbf{A}}{}^{\mathcal{W}}\tilde{\mathbf{B}}^T - {}^{\mathcal{W}}\tilde{\mathbf{B}}{}^{\mathcal{W}}\tilde{\mathbf{A}}^T \tag{5} $$



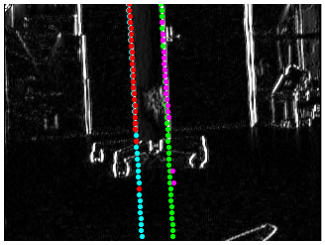
(a) Raw image (320×240) of the pole using front high speed camera.



(b) Hough transform yields 140 candidate lines. (green)



(c) Detected pole edges (green) chosen from line candidates based on slope, intercept, line length and edge strength.



(d) The 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).

Fig. 3. Stages in the line tracking process [4]: (a) Raw image. (b) Hough transform. Computational time of this step varies depending on number of line features in the scene. (c) Select best candidates based on target geometry priors. Boot strapping is completed at this step. (d) Line tracking.

where ${}^{\mathcal{W}}\mathbf{L}$ is a *Plücker matrix* $\in \mathbb{R}^{4\times4}$. The tilde denotes the homogeneous form of the point ($\in \mathbb{P}^3$). A camera matrix (intrinsic and extrinsic) $\mathbf{C}(\xi_C) \in \mathbb{R}^{3\times4}$, is given by

$$\begin{aligned}\mathbf{C}(\xi_C) &= \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \xi_C^{-1} \\ &= \mathbf{K}\mathbf{P}_0\xi_C^{-1}\end{aligned} \tag{6}$$

where ξ_C is the camera pose $\in SE(3)$ with respect to the world coordinate, f_x and f_y are focal length, u_0 and v_0 are principal point. The projection of ${}^{\mathcal{W}}\mathbf{L}$ onto the camera image plane is given by

$$[\boldsymbol{\ell}]_\times = \mathbf{C}(\xi_C)\ {}^{\mathcal{W}}\mathbf{L}\ \mathbf{C}(\xi_C)^T \tag{7}$$

where $[\boldsymbol{\ell}]_\times$ is a skew-symmetric matrix and $\boldsymbol{\ell} = (\ell_1, \ell_2, \ell_3)$ is the homogeneous line equation on the image plane.

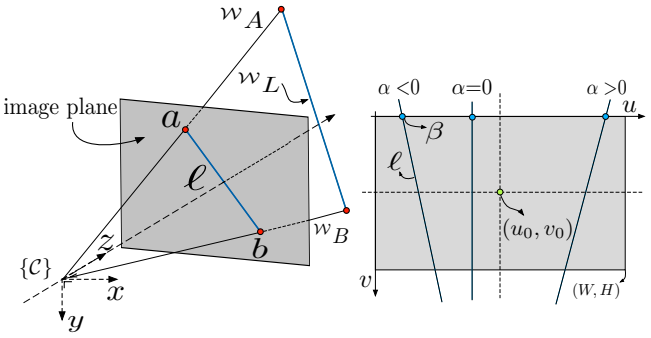
$$\ell_1 u + \ell_2 v + \ell_3 = 0 \tag{8}$$

u and v denote a horizontal and vertical axis of image plane respectively (see Fig 4).

C. Line Detection and Tracking

We make use of our previous fast line tracker [4] which we briefly summarize here. The computationally expensive Canny edge detection and Hough transform are utilized only for boot strapping. Once two pole edges are detected, we compute camera horizontal position using a pinhole camera model. This is used for EKF initialization.

Next, the tracker is invoked while the vehicle is flying. There are two steps: line searching and line model fitting.



(a) (b)

Fig. 4. (a) Perspective image of a line ${}^{\mathcal{W}}L$ in 3D space. a and b are projections of the world point and ℓ is a line on the image plane. (b) Image plane representation of slope (α) and intercept (β), measured in rad and pixels respectively.

Horizontal gradient (Sobel kernel) images are computed. We sample uniformly distributed points along previous detected lines. Afterward, we compute maxima along a fixed length of horizontal searching line. These maxima are input to a line fitting algorithm using RANSAC [17] to update the line model for the next iteration. The total number of samples, searching line and RANSAC parameters affect tracking performance. Since this tracking procedure must be completed in 10 ms, we empirically choose parameters within this constraint. 60 samples/line, 24 pixels for searching line, 0.3 pixel and 500 for RANSAC threshold and the maximum iteration respectively. Fig 3 illustrates these phases and more details on boot strapping and algorithm are presented [4].

Each line is represented in homogeneous form (8), but to simplify the state estimator, we use a simpler parameterization such that

$$[\alpha_k, \beta_k]^T, \text{ where } \alpha_k = \frac{\ell_1}{\ell_2},\ \beta_k = \frac{-\ell_3}{\ell_2} \tag{9}$$

and n denotes line number at sample k. Note that this parameterization is $\frac{\pi}{2}$ rotated form of a 2D line in order to avoid the singular case for a vertical line. This is non-singular for all cases except a horizontal line ($\ell_2 = 0$) which we do not expect in our application, see Fig 4(b).

IV. Estimation

A. Horizontal plane EKF

The position and velocity of the vehicle in the horizontal plane is estimated using a 100 Hz vision and 70 Hz inertial data. These sensor modalities are complementary in that the IMU outputs are subject to drift over time, whereas the visually acquired pole edge measurements are drift free and absolute with respect to the world coordinate.

1) Process model: The process model for a flying body assumes constant acceleration as used by [18].

$${}^{\mathcal{W}}\hat{\mathbf{X}}\langle k+1|k\rangle = \mathbf{A}\,{}^{\mathcal{W}}\hat{\mathbf{X}}\langle k|k\rangle + \mathbf{B}\mathbf{b}_k + \mathbf{v} \tag{10}$$

where ${}^{\mathcal{W}}\mathbf{X}_k = \left[{}^{\mathcal{W}}x_k, {}^{\mathcal{W}}y_k, {}^{\mathcal{W}}\dot{x}_k, {}^{\mathcal{W}}\dot{y}_k, \phi_k, \theta_k\right]^T$.

$\hat{\mathbf{X}}\langle k+1|k\rangle$ is the estimate of $\mathbf{X}$ at time $k+1$ given observations up to time k. $\mathbf{b}_k = \left[{}^{\mathcal{W}}\ddot{x}_k, {}^{\mathcal{W}}\ddot{y}_k, \dot{\phi}_k, \dot{\theta}_k\right]^T$

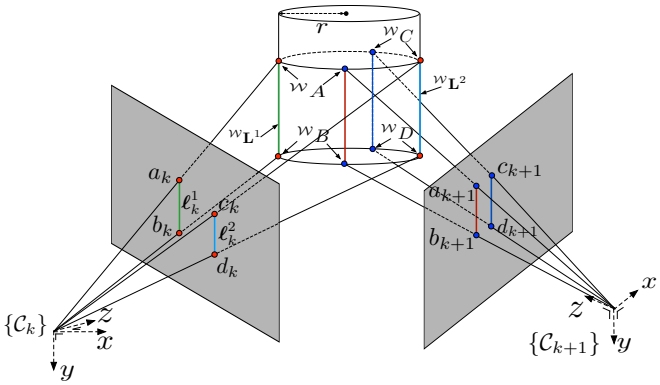


Fig. 5. Projection model for a cylindrical object. $^{\mathcal{W}}A, ^{\mathcal{W}}B, ^{\mathcal{W}}C, ^{\mathcal{W}}D \in \mathbb{R}^3$ denote points in the world frame with $a_k, b_k, c_k, d_k \in \mathbb{R}^2$ denoting their corresponding projection onto a planar imaging surface at a sample k. Although we actually measure different world points between frames, they are considered to be the same point due to the cylindrical nature of the object and the choice of line representation.

represents the sensor observed motion of the vehicle. $\mathbf{A}$ and $\mathbf{B}$ describe the evolution of a state vector and are given by

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 & 0 \\ \Delta t & 0 & 0 & 0 \\ 0 & \Delta t & 0 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix} \tag{11}$$

It is worth mentioning accelerometers measure the difference between the actual acceleration of a robot and the gravity vector in $\{\mathcal{B}\}$. Therefore, accelerations in $\{\mathcal{W}\}$, $\left[^{\mathcal{W}}\ddot{x}, ^{\mathcal{W}}\ddot{y}, ^{\mathcal{W}}\ddot{z}\right]^T$ are

$$^{\mathcal{W}}\mathbf{a} = {}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}\, {}^{\mathcal{B}}\mathbf{a}_m - \mathbf{g} \tag{12}$$

where $\mathbf{g}$ is a downward gravity vector, $[0, 0, g]^T$ and $^{\mathcal{B}}\mathbf{a}_m$ is the accelerometer measurement. Process noise $\mathbf{v}$ is given as

$$\mathbf{v} \sim \mathcal{N}(0, \mathcal{Q}) \tag{13}$$
$$\mathcal{Q} = \text{diag}\left[\sigma^2_{^{\mathcal{W}}x} \quad \sigma^2_{^{\mathcal{W}}y} \quad \sigma^2_{^{\mathcal{W}}\dot{x}} \quad \sigma^2_{^{\mathcal{W}}\dot{y}} \quad \sigma^2_{\phi} \quad \sigma^2_{\theta} \quad \right]$$

where $\mathcal{Q}$ is the covariance matrix of the process noise. $\mathcal{N}(0, \mathcal{Q})$ denotes a zero-mean Gaussian noise. σ is standard deviation of the corresponding states. The covariance propagation step follows the standard Kalman Filter procedure. There is an ambiguity for $^{\mathcal{W}}y$ and yaw angle (ψ), as both result in the target appearing to move horizontally in the image and it is a challenge to decouple them without using additional sensors. Therefore, we omit yaw (heading) angle estimation in the EKF states and assume it is controlled independently.

2) Measurement model: Four points $\in \mathbb{R}^3$ that lie on the two side edges of a pole are defined in $\{\mathcal{W}\}$. Two Plücker lines, $^{\mathcal{W}}\mathbf{L}^1$ and $^{\mathcal{W}}\mathbf{L}^2$, are formed and projected onto the image plane. This is shown in Fig. 5.

We partition the measurement into two components: visual $\mathbf{z}_{\text{cam}}$, and inertial, $\mathbf{z}_{\text{IMU}}$. The measurement vector is

$$\mathbf{Z}_k = \begin{bmatrix} \boldsymbol{\ell}_k^1 \\ \boldsymbol{\ell}_k^2 \\ \hline \phi_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} \mathbf{z}_{\text{cam}} \\ \mathbf{z}_{\text{IMU}} \end{bmatrix} \tag{14}$$

where $\boldsymbol{\ell}_k^i \in \mathbb{R}^2$ are the 2D line from the tracker (See (9)). The projected line observation is given by the nonlinear function with (7)

$$\mathbf{z}_{\text{cam}} = \begin{bmatrix} h_{\text{cam}}(^{\mathcal{W}}\mathbf{L}^1, ^{\mathcal{W}}\hat{x}_k, ^{\mathcal{W}}\hat{y}_k, \hat{\phi}_k, \hat{\theta}_k, \mathbf{w}) \\ h_{\text{cam}}(^{\mathcal{W}}\mathbf{L}^2, ^{\mathcal{W}}\hat{x}_k, ^{\mathcal{W}}\hat{y}_k, \hat{\phi}_k, \hat{\theta}_k, \mathbf{w}) \end{bmatrix} \tag{15}$$
$$= \begin{bmatrix} \hat{\mathbf{C}}\, ^{\mathcal{W}}\mathbf{L}^1 \hat{\mathbf{C}}^T \\ \hline \hat{\mathbf{C}}\, ^{\mathcal{W}}\mathbf{L}^2 \hat{\mathbf{C}}^T \end{bmatrix} \tag{16}$$

where $\hat{\mathbf{C}}$ denotes $\mathbf{C}(^{\mathcal{W}}\hat{x}_k, ^{\mathcal{W}}\hat{y}_k, \hat{\phi}_k, \hat{\theta}_k)^{\mathcal{W}}$ and unobservable states, $^{\mathcal{W}}z_k$ and ψ, are omitted. $\mathbf{w}$ is measurement noise with measurement covariance matrix, $\mathcal{R}$

$$\mathbf{w} \sim \mathcal{N}(0, \mathcal{R}) \tag{17}$$
$$\mathcal{R} = \text{diag}\left[\sigma^2_{\alpha 1} \quad \sigma^2_{\beta 1} \quad \sigma^2_{\alpha 2} \quad \sigma^2_{\beta 2} \quad \sigma^2_{\phi} \quad \sigma^2_{\theta}\right]$$

Again, we manually tune these parameters by comparing filter output with Vicon ground truth. We generated the runtime code for (16) using the MATLAB symbolic toolbox and then exporting the C++ code. This model is 19K lines of code but computation time is just 6 μs.

The update step for the filter requires linearization of this line model and evaluation of the two Jacobians

$$H_x = \frac{\partial h_{\text{cam}}}{\partial \mathbf{x}}|_{\hat{\mathbf{x}}(k)}, \quad H_w = \frac{\partial h_{\text{cam}}}{\partial \mathbf{w}} \tag{18}$$

H_x is a function of state that includes the camera projection model. We also use the MATLAB symbolic toolbox and automatic code generation (58K lines of code) for H_x. It takes 30 μs to compute in the C++ implementation.

The remaining observations are the vehicle attitude, directly measured by the onboard IMU ($\mathbf{z}_{\text{IMU}}$) and reported at 70Hz over a serial link. The linear observation model for the attitude is

$$\mathbf{z}_{\text{IMU}} = \begin{bmatrix} \phi_k \\ \theta_k \end{bmatrix} = \mathbf{H}_{\text{IMU}}\, ^{\mathcal{W}}\hat{\mathbf{X}} \tag{19}$$

$$\mathbf{H}_{\text{IMU}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{20}$$

The measurements $\mathbf{z}_{\text{cam}}$ and $\mathbf{z}_{\text{IMU}}$ are available at 100 Hz and 70 Hz respectively. The EKF is synchronous with the 100 Hz vision data and the most recent $\mathbf{z}_{\text{IMU}}$ measurement is used for the filter update. Inputs-outputs of horizontal plane EKF is presented in Fig 7.

3) Simulation: In order to validate the EKF line model and Jacobian, we create a virtual camera observing four points, $\mathbf{P}_i \in \mathbb{R}^3$ and move the camera with sinusoidal motion in 4 DOF ($^{\mathcal{W}}x, ^{\mathcal{W}}y, \phi, \theta$) using the simulation framework of [19]. Note that z and ψ are omitted here since they are difficult to determine using a front facing monocular camera. z is almost unobservable from two vertical lines of the pole, and there is an ambiguity between ψ and y axis

translation. To emulate the errors in line measurements we set the measurement uncertainties to be $\sigma_\alpha = 1^\circ$ and $\sigma_\beta = 4$ pixels in line parameters, α and β, from (9). Estimation results, their confidence boundary, and noise parameters are shown in Fig. 6. We see good quality estimates of position and velocity in the horizontal plane, whilst decoupling the effects of attitude on the projected line parameters. We see that the x-axis estimation is noisier than the y-axis. A closer object moves more in the image plane than one further out.

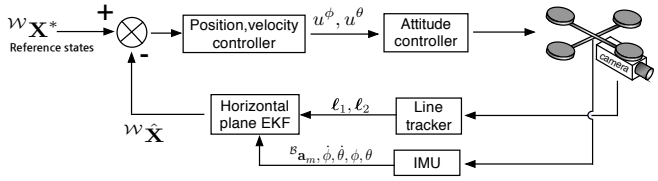


Fig. 7. Input-output of horizontal plane EKF. u^ϕ and u^θ denote control inputs for roll and pitch commands. $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ are tracked 2D lines and ${}^{\mathcal{B}}\mathbf{a}_m$ is onboard acceleration measurement from IMU. Position and velocity controllers are addressed in Section V.

B. Vertical state estimation using KF

We observe altitude directly using a down-facing ultrasonic sensor at 20 Hz, but the update rate is too low for control purposes and any derived velocity signal has too much lag. Therefore we use another Kalman Filter to fuse this with the 70 Hz inertial data which includes vertical acceleration in $\{\mathcal{B}\}$. The sonar sensor is calibrated by least squares to ground truth state estimates. The altitude and z-axis acceleration measurement in $\{\mathcal{B}\}$ are transformed to $\{\mathcal{W}\}$ using $\hat{\phi}$ and $\hat{\theta}$ angles and (12). ${}^{\mathcal{W}}\mathbf{X}^{\text{alt}}$ is the vertical state, $\left[{}^{\mathcal{W}}z, {}^{\mathcal{W}}\dot{z}\right]^T$ and the process model is given by

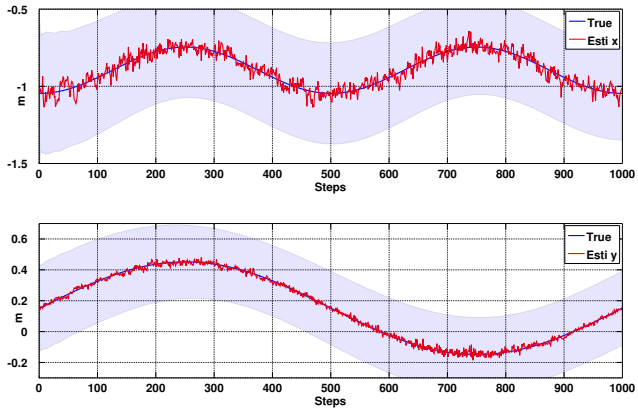
$$ {}^{\mathcal{W}}\hat{\mathbf{X}}^{\text{alt}}_{\langle k+1|k\rangle} = \mathbf{A}^{\text{alt}} \begin{bmatrix} {}^{\mathcal{W}}\hat{z}_{\langle k|k\rangle} \\ {}^{\mathcal{W}}\hat{\dot{z}}_{\langle k|k\rangle} \end{bmatrix} + \mathbf{B}^{\text{alt}} {}^{\mathcal{W}}\ddot{z} + \mathbf{v}^{\text{alt}} \tag{21} $$

$$ \text{where } \mathbf{A}^{\text{alt}} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \quad \mathbf{B}^{\text{alt}} = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} \tag{22} $$

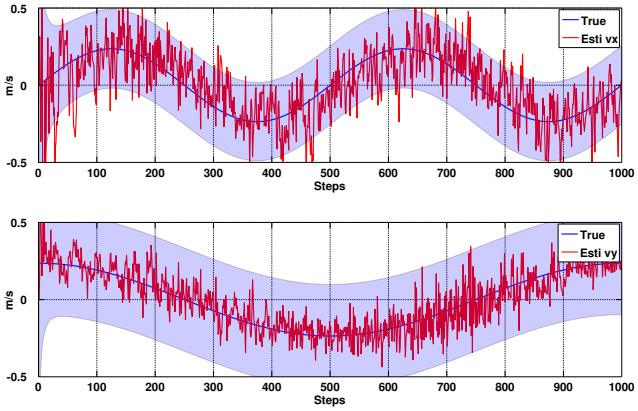
$\mathbf{v}^{\text{alt}}$ is process noise vector of ${}^{\mathcal{W}}z$ and ${}^{\mathcal{W}}\dot{z}$. The covariance matrices of process and measurement noise, $\mathcal{Q}^{\text{alt}}$ and $\mathcal{R}^{\text{alt}}$, are defined in the same way as discussed in IV-A. The observation matrix is $\mathbf{H}^{\text{alt}} = \begin{bmatrix} 1 & 0 \end{bmatrix}$.

V. Controller Design

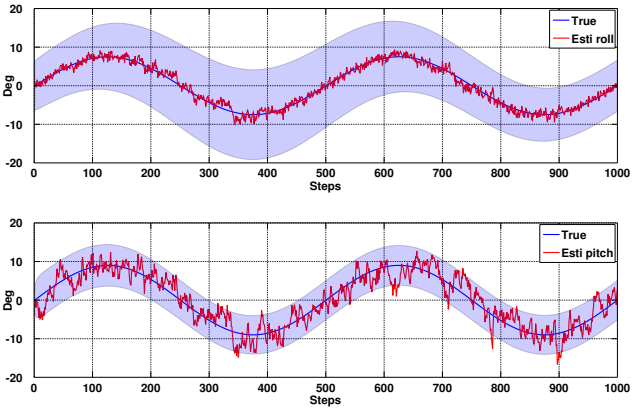
A quadrotor platform is an under-actuated dynamic system that is force actuated and undamped. This makes it challenging to control as high-quality velocity signals are required. The hardware platform we use in this paper has a mass of 1.85 kg and is about 3 times heavier than the one reported in our previous work [20], due to improved onboard image processing capability. This requires more thrust and higher average rotor speeds to lift the platform which leads us to revisit system modeling. The system dynamics are identified by recording command inputs and attitude outputs during manual flight. The position and velocity controllers are designed using these dynamic models in MATLAB and Simulink. This allows us to determine appropriate controller



(a) Simulated ${}^{\mathcal{W}}\hat{x}$ and ${}^{\mathcal{W}}\hat{y}$ position estimation results with 3σ confidence boundary.



(b) Simulated ${}^{\mathcal{W}}\hat{\dot{x}}$ and ${}^{\mathcal{W}}\hat{\dot{y}}$ velocity estimation simulation results with 3σ confidence boundary.



(c) Simulated $\hat{\phi}$ and $\hat{\theta}$ angle simulation results with 3σ confidence boundary.

Fig. 6. Simulated results for horizontal plane state estimation using vision only. We intentionally fly the vehicle following a sinusoidal trajectory and add the line measurement uncertainties, $\sigma_\alpha = 1^\circ$ and $\sigma_\beta = 4$ pixels to simulate the error. Most of the states are within 3σ confidence level. Total simulation time is 10 sec, with sampling rate of 100 Hz.

gains before flying, and to fine tune them with ROS online parameter reconfiguration while flying. For forward motion we are interested in the pitch response where the input is pitch command θ^* (an arbitrary "stick input" in the range -700 to 700) and the output is the IMU pitch angle measurement $\hat{\theta}$ (radians). Actual flight data and the response of our second-order discrete time model which has 0.0135 sec

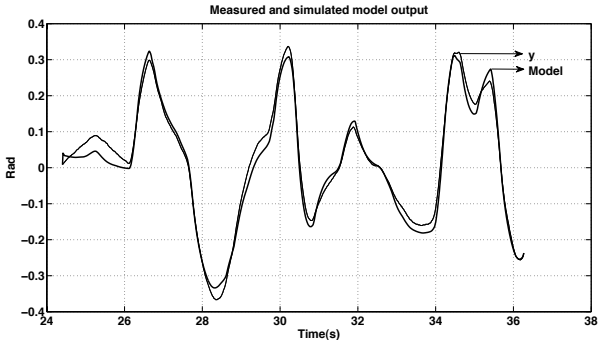


Fig. 8. Actual response, y and simulated model output, Model, for quadrotor pitch axis. This discrete-time model has 2 poles and 0 zeros with 0.0135 sec sample time. Fitting score to the output is 86.93 %.

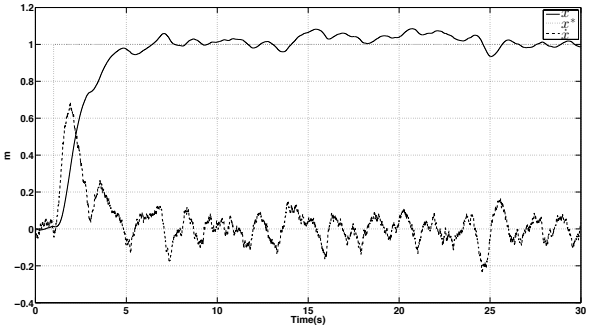


Fig. 10. Simulation results for position and velocity controller. x, x^* and $\dot{x}$ denote the position (with respect to world coordinate along x axis), desired x position and velocity states. Rise time for this position controller is around 5 s and standard deviation is 0.031 m during 5 s∼30 s interval.

sample time

$$G(z) = \frac{8.75 \times 10^{-5} z}{z^2 - 1.784z + 0.804} \tag{23}$$

is compared in Fig. 8. The model exhibits a rise time of 0.461 sec and a settling time of 0.8 sec and is a function of the attitude controller embedded in the MikroKopter flight controller. The roll axis model is similar due to the vehicle's symmetry.

A. xy-horizontal control

Fig. 9 shows the control structure for translational motion. An inner proportional velocity controller and an outer PI position controller run at 100 Hz, synchronized with the camera and the horizontal EKF. By making a small angle assumption, the force acting on the quadrotor along the x-axis of $\{\mathcal{W}\}$ can be modeled as ${}^{\mathcal{W}}f_x = g \sin\theta$ [20] [21]. A large acceleration noise, $\mathcal{N}_{\ddot{x}} \sim (0, 0.5\,\mathrm{m/s^2})$, is added to the system.

We model communications delay inherent in the serial telemetry data transmission between the onboard PC and the MikroKopter flight controller. This is estimated based on packet size and baud rate. We also model processing delay which lumps together: image acquisition time (exposure time and image transport), line feature extraction and EKF execution. Image transport time over the USB bus is difficult to determine and we choose a reasonable maximum delay based on the camera sampling rate, 2 samples $\sim 20\,\mathrm{ms}$. Fig. 10 shows simulation results for horizontal position, x, and velocity, $\dot{x}$, with the above-mentioned time delay models and horizontal dynamics model.

B. Vertical control

We use altitude, ${}^{\mathcal{W}}z$, and altitude rate, ${}^{\mathcal{W}}\hat{\dot{z}}$, from the vertical axis Kalman Filter as feedback for the thrust command u_k^h such that

$$u_k^h = K_p({}^{\mathcal{W}}z^* - {}^{\mathcal{W}}\hat{z}) + K_i e^h + K_d {}^{\mathcal{W}}\hat{\dot{z}}_{\langle k|k\rangle} + u_k^{\text{off}} \tag{24}$$

where ${}^{\mathcal{W}}\hat{\dot{z}}_{\langle k|k\rangle}$ is vertical axis velocity estimate at time k and e^h is the integral error over time. The steady state thrust u_k^{off} required for hover is a function of battery voltage since the output power of a battery varies over flying with the same weight of the vehicle. We use a cubic polynomial model driven by the measured Lithium-Polymer terminal voltage [20].

VI. Experimental results

We present two sets of experimental results for state estimation and controller performance while hovering. Three different visual sampling rates were tested including 30 Hz, 60 Hz and 100 Hz while the controller gains remain fixed. The performance can be seen in the associated video clip with additional details available online[1].

A. Hardware and software system set up

All processing is performed by an onboard single board computer (SBC) with a dual core 1.9 GHz Celeron CPU running Ubuntu Linux and Robot Operating System (ROS). Weight, including additional 4 GB RAM and compact flash, is 314 g and consumes 20 W. With this configuration, the quadrotor is able to fly up to 16 mins. The camera is a low-cost high-speed Playstation EyeToy connected via USB. This CMOS camera has a rolling shutter which is problematic on a moving platform. We are able to adjust essential camera options (such as exposure, frame rate, gain etc.) through the USB driver. The IMU provides $[\Phi, \dot{\Phi}, \mathbf{a}]$ where Φ is the roll-pitch-yaw angles $[\phi, \theta, \psi]$, $\dot{\Phi}$ the RPY angle rates and $\mathbf{a}$ the 3-axis acceleration. This rate is limited to 70 Hz due to the RS232 protocol bandwidth. Fig. 11 shows the hardware, software configuration and timing profiling for the corresponding ROS node. Different colors denote different sampling rates and arrows denote data flow at a given frequency. Each box is an individual ROS node implemented using C++. Precision Time Protocol (PTP) is utilized for time synchronization between the onboard computer and the Vicon data logger.

B. State estimation

State estimation is evaluated by computing the root mean square (RMS) error between filter estimation and ground truth as measured by the Vicon system. Table I summarizes the RMS error for state estimation, at all sample rates. For the 60 Hz and 100 Hz cases, we consider a flight period from 10 –70 sec. For the 30 Hz case we consider only a very short

[1] https://code.google.com/p/cyphy-pole-quad

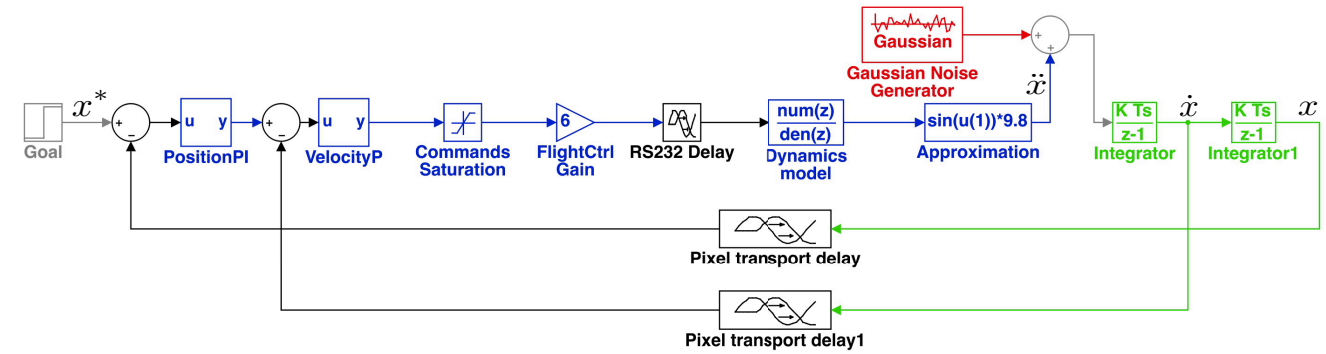


Fig. 9. x-axis position controller with outer PI position loop and inner proportional velocity loop. Communication (8.3 ms) and computation (20 ms) delays are included along with acceleration noise, $\sigma_{\ddot{x}} \sim 0.5\,\mathrm{m/s^2}$.

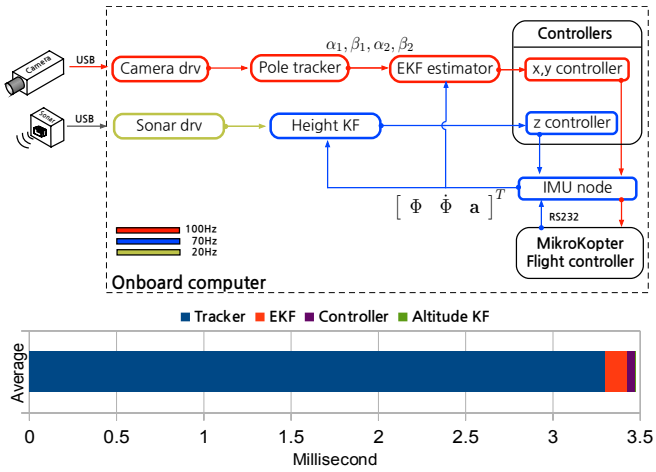


Fig. 11. Software system diagram. Different colors denote corresponding sampling rates. The software is implemented using ROS and all processing is finished within 3.5 ms on average. The tracker utilizes 55% of the CPU. EKF includes line model and Jacobian calculation. Each step takes 6 μs and 30 μs respectively.

TABLE I

RMS ERROR FOR ESTIMATOR ACCURACY

State variable	30 Hz	60 Hz	70 Hz	100 Hz	units
$\mathcal{W}\hat{x}$	0.4544	0.0465	—	0.0284	m
$\mathcal{W}\hat{y}$	0.2256	0.0480	—	0.0071	m
$\mathcal{W}\hat{z}$	—	—	0.0089	—	m
$\mathcal{W}\hat{\dot{x}}$	1.0610	0.0747	—	0.0941	m/s
$\mathcal{W}\hat{\dot{y}}$	1.5214	0.0529	—	0.0335	m/s
$\mathcal{W}\hat{\dot{z}}$	—	—	0.0586	—	m/s
$\hat{\phi}$	7.37	1.72	—	1.44	Deg
$\hat{\theta}$	5.44	1.46	—	1.38	Deg
Interval	5∼9	10∼70	10∼70	10∼70	sec

period of flight from 5 – 9 sec after which the tracker lost the target (see video). The performance of the horizontal 100 Hz and the vertical 70 Hz estimators are shown in Fig. 12.

C. Controller performance

Control performance is evaluated by computing the RMS error between the goal position (x, y and z) and ground truth as measured by the Vicon system. The performance of the controller is shown in Fig. 12. Interestingly, although the x-axis velocity estimation is noisy, the control performance for this axis is not significantly worse than for the y-axis. This implies that the quadrotor plant acts like a low-pass filter. Table II summarizes the RMS error for position control, at

TABLE II

RMS ERROR FOR CONTROL PERFORMANCE

State variable	30 Hz	60 Hz	70 Hz	100 Hz	units
$\mathcal{W}\hat{x}$	0.7538	0.0658	—	0.0444	m
$\mathcal{W}\hat{y}$	0.1505	0.0603	—	0.0232	m
$\mathcal{W}\hat{z}$	—	—	0.0613	—	m
$\mathcal{W}\hat{\dot{x}}$	0.5185	0.0559	—	0.0522	m/s
$\mathcal{W}\hat{\dot{y}}$	0.3916	0.0630	—	0.0180	m/s
$\mathcal{W}\hat{\dot{z}}$	—	—	0.0278	—	m/s
Interval	5∼9	10∼70	10∼70	10∼70	sec

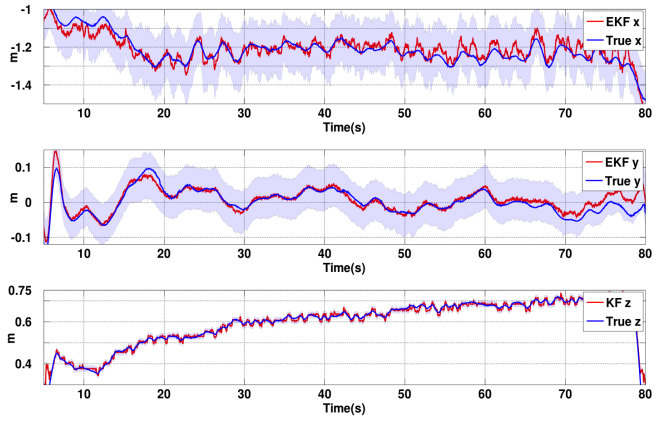
all sample rates, and over the same flight time intervals as above.

VII. Conclusions and Future Works

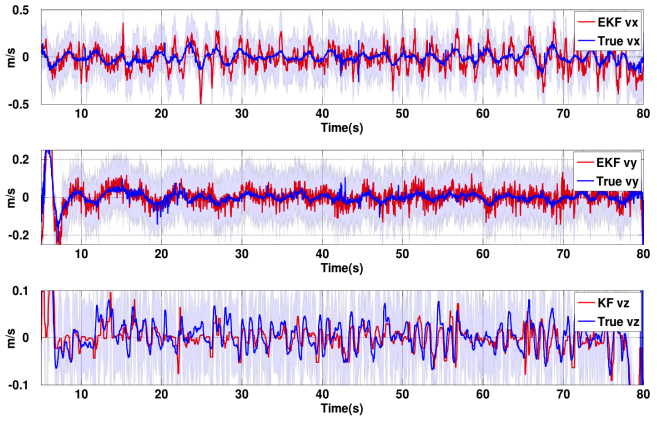
We have demonstrated a high-speed monocular visual servoing system that comprises onboard vision and inertial measurements. We like to think of this as a small step toward replicating the performance of a motion capture system, but onboard. While we have made progress on the sample rate, there remains a large gap in terms of camera resolution (QVGA versus 4 million pixels), camera quality (rolling versus global shutter), and the advantages of having the cameras fixed in the world, not vibrating in 6 DoF. Of course you get what you pay for, and we are comparing position estimators that cost USD 40 with one that costs **four** orders of magnitude more. The result is a larger measurement variance in our system compared to a motion capture system, but that is sufficient for this application.

Experimental results demonstrate comparable performance for our proposed estimator and a motion capture system. We see that control performance improves significantly with visual sampling rate. We note that the use of high-speed vision promises some important other benefits. From a vision perspective, for example, the scene change from frame to frame is small, which means that simple visual features (with poor invariance properties) should work well. From a control viewpoint, a high sample rate is important for rejection disturbance and lag compensation in velocity estimation.

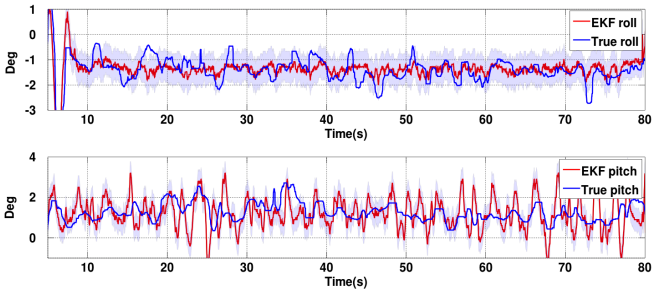
We have a large program of ongoing work. Firstly, in order to disambiguate between lateral translation and yaw motion as perceived by the front camera, we are integrating a visual compass with a down facing camera into the estimator framework. The sonar works to only 2 m, so we are looking at integrating visual odometry along the vertical



(a) Experimental results for position estimation and control with 3σ confidence boundary. -1.2 m, 0 m and 0.6 m are the desired position for ${}^{\mathcal{W}}\hat{x}$, ${}^{\mathcal{W}}\hat{y}$ and ${}^{\mathcal{W}}\hat{z}$. Note that z-axis is inverted for visualization.



(b) Experimental results for velocity estimation, ${}^{\mathcal{W}}\hat{\dot{x}}$, ${}^{\mathcal{W}}\hat{\dot{y}}$ and ${}^{\mathcal{W}}\hat{\dot{z}}$, with 3σ confidence boundary. Desired velocity is taken as the output of the position controller as shown Fig. 9.



(c) Experimental results for roll $\hat{\phi}$ and pitch $\hat{\theta}$ attitude estimation with 3σ confidence boundary.

Fig. 12. Experimental results while hovering. Horizontal states and attitude are 100 Hz whereas vertical states are 70 Hz. Each state is shown in different scale for better visualization.

axis of the structure to estimate height. We are working to improve the sub-pixel accuracy of the line tracker to improve distance sensitivity along the camera optical axis, and also to improve the robustness with respect to background clutter [22]. Currently our tracker relies on control performance to keep the lines at the same location from frame to frame. Using information from the estimated state would improve tracking robustness. We will integrate this into a teleoperation framework so that a remote pilot can specify the position loop set-points which will allow a non-skilled person to fly a quadrotor close to a structure beyond line of sight. Finally we need to move to outdoor field testing and an example real world vertical infrastructure inspection task.

ACKNOWLEDGMENT

The authors would like to thank Aaron Mcfadyen for assistance while using the Vicon system.

REFERENCES

[1] S. Kim, M. Spenko, S. Trujillo, B. Heyneman, D. Santos, and M. Cutkosky, "Smooth Vertical Surface Climbing With Directional Adhesion," *Robotics, IEEE Transactions on*, vol. 24, pp. 65 –74, Feb. 2008.

[2] C. Balaguer, A. Gimenez, and A. Jardon, "Climbing Robots' Mobility for Inspection and Maintenance of 3D Complex Environments," *Autonomous Robots*, vol. 18, pp. 157–169, 2005.

[3] G. C. Haynes, A. Khripin, G. Lynch, J. Amory, A. Saunders, A. A. Rizzi, and D. E. Koditschek, "Rapid Pole Climbing with a Quadrupedal Robot," in *IEEE International Conference on Robotics and Automation*, pp. 2767–2772, May 2009.

[4] I. Sa and P. Corke, "100Hz Onboard Vision for Quadrotor State Estimation," in *Australasian Conference on Robotics and Automation*, Dec 2012.

[5] G. M. Hoffmann, *Autonomy for Sensor-Rich Vehicles: Interaction between Sensing and Control Actions*. PhD thesis, Stanford Univ., 2008.

[6] D. Mellinger and V. Kumar, "Minimum Snap Trajectory Generation and Control for Quadrotors," in *IEEE International Conference on Robotics and Automation*, pp. 2520– 2525, May 2011.

[7] S. Lupashin, A. Schö andllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *IEEE International Conference on Robotics and Automation*, pp. 1642 –1648, May 2010.

[8] A. Bachrach, A. de Winter, R. He, G. Hemann, S. Prentice, and N. Roy, "RANGE - robust autonomous navigation in GPS-denied environments," in *IEEE International Conference on Robotics and Automation*, pp. 1096–1097, May 2010.

[9] S. Weiss and R. Siegwart, "Real-Time Metric State Estimation for Modular Vision-Inertial Systems," in *IEEE International Conference on Robotics and Automation*, May 2011.

[10] S. Grzonka, G. Grisetti, and W. Burgard, "A Fully Autonomous Indoor Quadrotor," *Robotics, IEEE Transactions on*, vol. 8, pp. 90–100, Feb. 2012.

[11] M. Cutler, B. Michini, and J. P. How, "Lightweight Infrared Sensing for Relative Navigation of Quadrotors," in *International Conference on Unmanned Aircraft Systems*, 2013.

[12] J. Solà, T. Vidal-Calleja, J. Civera, and J. M. Montiel, "Impact of Landmark Parametrization on Monocular EKF-SLAM with Points and Lines," *International Journal of Computer Vision*, vol. 97, pp. 339–368, May 2012.

[13] A. Bartoli and P. Sturm, "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," *Computer Vision and Image Understanding*, vol. 100, no. 3, pp. 416–441, 2005.

[14] R. Mahony and T. Hamel, "Image-based visual servo control of aerial robotic systems using linear image features," *Robotics, IEEE Transactions on*, vol. 21, pp. 227 – 239, Apr. 2005.

[15] A. Bartoli and P. Sturm, "The 3D line motion matrix and alignment of line reconstructions," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I–287, IEEE, 2001.

[16] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[17] M. Fischler and R. Bolles., "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, 1981.

[18] H. Durrant-Whyte, "Introduction to Estimation and the Kalman Filter," tech. rep., The University of Sydney, Jan. 2001.

[19] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in MATLAB*. Springer, 2011.

[20] I. Sa and P. Corke, "System Identification, Estimation and Control for a Cost Effective Open-Source Quadcopter," in *IEEE International Conference on Robotics and Automation*, pp. 2202– 2209, May 2012.

[21] G. P. Tournier, M. Valenti, J. P. How, and E. Feron, "Estimation and control of a quadrotor vehicle using monocular vision and moire patterns," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, pp. 21–24, 2006.

[22] I. Sa and P. Corke, "Improved line tracker using IMU and Vision for visual servoing," in *Australasian Conference on Robotics and Automation (Submitted)*, 2013.