Smartphone Quadrotor Flight Controllers and Algorithms for Autonomous Flight

Tyler Ryan[1] and H. Jin Kim[2]

I. Introduction

Some of the most interesting recent research in quadrotor flight is related to fully autonomous systems, i.e. only using onboard processing and sensing. Many researchers have demonstrated different aspects needed for autonomy and within the last couple of years some have even demonstrated fully autonomous flight. [1], [2] One of the biggest challenges is finding powerful lightweight mobile processors, along with the state estimation and control algorithms that are fast enough to be run on those processors.

In this paper we present some of the results of our use of a smartphone as a quadrotor flight controller. Modern smartphones have all the necessary sensors needed for quadrotor flight and also provide a powerful mobile computing environment. Additionally, smartphone vendors are rapidly increasing the computation performance and, since our work uses the standard Android operating system, software is easily transferred to new phones as they are released.

The contributions of this paper are twofold. First, we describe the algorithms used towards the goal of fully autonomous flight. While many of the algorithms are not new, we have demonstrated that they are suitable for use on our smartphone flight controller. Secondly, this paper presents the first online implementation of a velocity estimation algorithm we recently proposed, [3] which generates image-space feature location prior distributions and then uses Bayesian inference to create "soft" point correspondences and calculate the maximum *a posteriori* velocity and height.

II. System Description

The UAV used in this work is a quadrotor, for which the dynamics have been well studied. In this section we just lay out the model used with respect to the rest of the paper.

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{1a}$$

$$\dot{\boldsymbol{v}} = Rf\boldsymbol{e}_3 \tag{1b}$$

$$\dot{R} = R\boldsymbol{\omega}_\times \tag{1c}$$

$$J\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega}_\times J\boldsymbol{\omega} + \boldsymbol{\tau} \tag{1d}$$

where $\boldsymbol{x}$ is the position, $\boldsymbol{v}$ is the velocity, R is the rotation matrix representing the attitude of the quadrotor with respect to inertial coordinates, J is the rotational inertia matrix, and $\boldsymbol{\omega}$ is the angular velocity in body fixed coordinates. The $\cdot_\times$ operator is used to denote the skew symmetric matrix representing the cross product and the inverse operation is denoted by vex, i.e. $\text{vex}(\boldsymbol{\omega}_\times) = \boldsymbol{\omega}$.

Image kinematics are assumed, for a sufficiently small timestep dt, to follow

$$\boldsymbol{q}(t+dt) - \boldsymbol{q}(t) \simeq L_\omega(t)\boldsymbol{\omega}(t)dt + L_v(t)\frac{\boldsymbol{v}(t)}{z(t)}dt \tag{2}$$

$$L_\omega \triangleq \begin{bmatrix} f^{-1}q_xq_y & -(f+f^{-1}q_x^2) & q_y \\ f+f^{-1}q_y^2 & -f^{-1}q_xq_y & -q_x \end{bmatrix}$$

$$L_v \triangleq \begin{bmatrix} -f & 0 & q_x \\ 0 & -f & q_y \end{bmatrix}$$

where $\boldsymbol{q}(t) \in \mathbb{R}^2$ is a feature point's image coordinate and f is the camera focal length.

III. Hardware

The quadrotor used in our experiments is a Mikrokopter [4] with the flight controller replaced by a Galaxy SIII smartphone. Use of the smartphone as the flight controller has some unique benefits as well as challenges (table I). The biggest benefits are twofold: 1) smartphones provide powerful computation power relative to their size, with upgraded versions regularly being released and requiring minimal, if any, software changes, and 2) the sensor and communication hardware comes prepackaged so the UAV engineer does not have to take the time to integrate several separate subsystems. To communicate with the motors we connect an arduino, which generates the commanded motor control signals, to the phone. In the future this board can also serve to communicate with other external systems, such sonar or laser scanners.

Smarphones also have unique challenges, but these are manageable when properly considered. The potentially most difficult challenge is the lack of a real-time operating system. In our experience, though, the sensor updates happen fast enough that this is not a major problem. Additionally, we label all data with the time that it is generated so time dependent tasks, such as integration or differentiation, are still accurate. The other challenge is lack of control over hardware selection or layout. We have not been limited by this so far, but if there are problems in the future external units can be integrated via the USB port.

IV. State Estimation and Control

In this section we describe the estimation and control algorithms used for flight. Algorithms commonly found in the quadrotor literature are only introduced without detail, but algorithms with significant modifications or not common in the literature are described in more detail.

[1]PhD candidate, Dept of Mech and Aero Engineering, Seoul National University, ryantr@gmail.com

[2]Associate Professor, Dept of Mech and Aero Engineering, Seoul National University, hjinkim@snu.ac.kr

Benefits	Challenges
• Powerful portable computing environment • Easily upgraded to new phones • Integrated hardware – Fast communication between sensors and logic – Globally synced timestamps	• Not real-time • Sensor events are sometimes delivered late – Event timestamps are still accurate • No control over individual hardware items – e.g. Rolling shuttter cameras

TABLE I: Benefits and challenges with smartphone flight controllers

A. *Attitude State Estimation*

For attitude estimation we have implemented the SO3 observer described in [5], running at approximately 200Hz, which is designed specifically for the sensors commonly used by quadrotors: gyroscopes provide high speed updates and accelerometers provide noisy inertial measurements which can be used to compensate for gyroscope bias. Using just these two sensors there is still one undetermined dimension (in the yaw angle for hover flight). If flying outdoors, a magnetometer can be used to correct this since the observer of [5] is able to filter out the high frequency magnetic disturbance caused by the motors. But in our indoor flight environment the iron used in the building construction causes low frequency distortion in the magnetic field making the magnetometer too unreliable. Currently we are using the Vicon system to correct this last bit of ambiguity, but in the future we will use onboard vision measurements.

Since we are using a smartphone, we also have the option of using the phone's own attitude estimation algorithm. In our case, the Galaxy SIII uses a sensor fusion algorithm provided by iNEMO which is based on an adaptive extended Kalman filter (EKF). While many researchers have reported success using their own EKFs, the iNEMO algorithm is not tuned for quadrotor flight so produces poor results. Figure 1 compares the SO3 observer performance with the Android (iNEMO) observer performance.

B. *Attitude State Control*

The attitude controller is derived from Lyapunov theory. First, define the rotation error metric, $\widetilde{R}$, as

$$\widetilde{R} = (R^d)^\intercal R \tag{3}$$

where R^d is the rotation matrix associated with the desired attitude. For $R^d = R$ we have $\widetilde{R} = I$ so the control law needs to drive $\widetilde{R}$ to the identity matrix. Before proceeding with the control derivation we need to define the symmetric and anti-symmetric parts of a matrix.

$$\Gamma_s(\widetilde{R}) = \frac{1}{2}(\widetilde{R} + \widetilde{R}^\intercal) \tag{4a}$$

$$\Gamma_a(\widetilde{R}) = \frac{1}{2}(\widetilde{R} - \widetilde{R}^\intercal) \tag{4b}$$

$$\widetilde{R} = \Gamma_s(\widetilde{R}) + \Gamma_a(\widetilde{R}) \tag{4c}$$

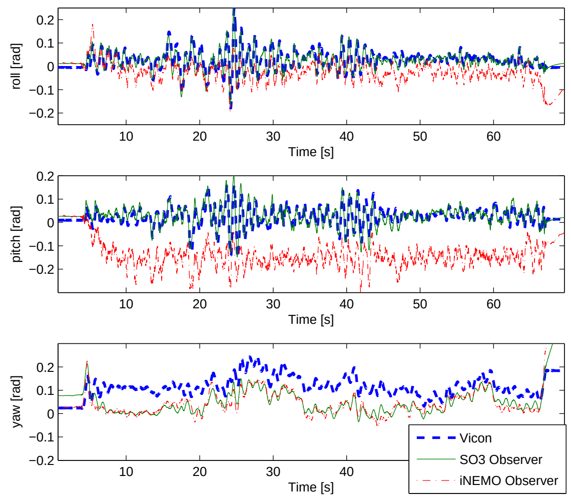


Fig. 1: Observer comparison. "Vicon" is the ground truth data. Bias offsets in the yaw angle are due to different origins. In this data, the magnetometer is used for yaw angle, but due to large variation in the bias caused by the environment we currently do most flights with the Vicon yaw angle.

To derive the attitude control law, we use the Lyapunov function

$$V \triangleq k_R \operatorname{tr}(I - \widetilde{R}) + \frac{1}{2}\boldsymbol{\omega}^\intercal J \boldsymbol{\omega} \tag{5}$$

Since $\widetilde{R}$ is the product of two rotation matrices, $\widetilde{R}$ is also a rotation matrix which ensures that the $\operatorname{tr}(I - \widetilde{R}) \geq 0$ with equality occurring if and only if $\widetilde{R} = I$. Looking at the time derivative,

$$\begin{aligned} \dot{V} &= k_R \operatorname{tr}(-\dot{\widetilde{R}}) + \boldsymbol{\omega}^\intercal J \dot{\boldsymbol{\omega}} \\ &= k_R \operatorname{tr}(-(\dot{R}^d)^\intercal R - (R^d)^\intercal \dot{R}) + \boldsymbol{\omega}^\intercal J \dot{\boldsymbol{\omega}} \end{aligned} \tag{6}$$

Now substitute in the quadrotor dynamics of eqn 1

$$\begin{aligned} \dot{V} &= -k_R \operatorname{tr}((\dot{R}^d)^\intercal R) - k_R \operatorname{tr}(R^d R \boldsymbol{\omega}_\times) + \boldsymbol{\omega}^\intercal(-\boldsymbol{\omega}_\times J \boldsymbol{\omega} + \boldsymbol{\tau}) \\ &= -k_R \operatorname{tr}((\dot{R}^d)^\intercal R) - k_R \operatorname{tr}(\widetilde{R} \boldsymbol{\omega}_\times) + \boldsymbol{\omega}^\intercal \boldsymbol{\tau} \\ &= -k_R \operatorname{tr}((\dot{R}^d)^\intercal R) - k_R \operatorname{tr}((\Gamma_s(\widetilde{R}) + \Gamma_a(\widetilde{R}))\boldsymbol{\omega}_\times) + \boldsymbol{\omega}^\intercal \boldsymbol{\tau} \end{aligned} \tag{7}$$

Taking advantage of the structure of $\Gamma_s(\widetilde{R})$, $\Gamma_a(\widetilde{R})$, and $\boldsymbol{\omega}_\times$, we know $\operatorname{tr}(\Gamma_s(\widetilde{R})\boldsymbol{\omega}_\times) = 0$ and $\Gamma_a(\widetilde{R})\boldsymbol{\omega}_\times = -2\boldsymbol{\omega}^\intercal \operatorname{vex}(\Gamma_a(\widetilde{R}))$, where the vex() operator is defined in section II. This gives the time derivative,

$$\dot{V} = -k_R \operatorname{tr}((\dot{R}^d)^\intercal R) + \boldsymbol{\omega}^\intercal \left(k_R \operatorname{vex}(\widetilde{R} - \widetilde{R}^\intercal) + \boldsymbol{\tau}\right) \tag{8}$$

From this, define the control law

$$\boldsymbol{\tau} \equiv -k_R \operatorname{vex}(\widetilde{R} - \widetilde{R}^\intercal) - S\boldsymbol{\omega} \tag{9}$$

$$\dot{V} = -k_R \operatorname{tr}((\dot{R}^d)^\intercal R) - \boldsymbol{\omega}^\intercal S \boldsymbol{\omega} \tag{10}$$

where S is a positive definite weighting matrix. If we assume $\dot{R}^d = 0$, then $\dot{V}$ is negative semi-definite. Using the control law defined in eqn 9, the only invariant set on $\dot{V} = 0$ is $\widetilde{R} = I$ and $\boldsymbol{\omega} = 0$ so by LaSalle's invariance principle we can state that the system is asymptotically stable.

In practice, $\dot{R}^d$ is defined by the outer loop controller and is generally not zero. We could adjust the outer loop controller gain to ensure eqn 9 is negative definite but in our experiments eqn 9 is almost always negative, with only small, brief deviations where it is above zero, so for simplicity no checks are made on $\dot{R}^d$ with respect to eqn 10.

C. Translation State Estimation

The translation state observer uses a standard Kalman filter with time updates running at 200Hz. For measurements, we have separated velocity and position as two separate measurements. Velocity measurements are estimated using the procedure described in section V. Currently, we use a Vicon system for position measurements at 10Hz with 1cm standard deviation noise artificially added. We are in the process of implementing algorithms to use the onboard vision system for position measurement.

D. Translation State Control

For control we follow the common practice of assuming attitude commands are tracked perfectly and, combined with thrust, can be treated as an input. More precisely, the net acceleration vector is considered the input,

$$\begin{bmatrix}\dot{\boldsymbol{p}}\\ \dot{\boldsymbol{v}}\end{bmatrix} = \begin{bmatrix}I & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{p}\\ \boldsymbol{v}\end{bmatrix} + \begin{bmatrix}0\\ I\end{bmatrix}\widehat{\boldsymbol{a}} + \begin{bmatrix}0\\ I\end{bmatrix}\boldsymbol{d} \tag{11}$$

where $\widehat{\boldsymbol{a}}$ is the acceleration command and $\boldsymbol{d}$ is acceleration due to disturbances, including tracking error by the attitude controller. To improve system robustness, both to disturbances and to measurement noise, we use an $\mathcal{H}_\infty$ controller modified to allow for user tuning. This controller is described in [6].

V. Velocity Estimation

In this section we summarize the velocity estimation algorithm we developed in [3], which can be run very quickly and is independent of any position estimation routines. The algorithm transforms prior distributions for the velocity and height to an image-space prior distribution for each feature location found in the previous image and then computes the maximum *a posteriori* estimate for the velocity and height based on the feature points found in the current image. Note that no feature descriptors or explicit feature matching are used. In [3], we showed that the algorithm described in this section was both faster and more accurate than explicit feature matching.

A. Establishing Correspondence Probabilities

In this section we describe how we determine the probabilistic, or "soft," correspondence between points in image 1, $Q_1 = \{\boldsymbol{q}_{1,i}\}_{i=1}^{N_1}$, and points in image 2, $Q_2 = \{\boldsymbol{q}_{2,j}\}_{j=1}^{N_2}$, where N_1 and N_2 are the total number of points found in the respective images. To reduce notational clutter, we will assume $\boldsymbol{\omega} = 0$ throughout this section; the extension to an arbitrary $\boldsymbol{\omega}$ is obvious.

The correspondence matrix C is the matrix with individual entries c_{ij} defined as the probability that point $\boldsymbol{q}_{1,i}$ in image 1 corresponds to point $\boldsymbol{q}_{2,j}$ in image 2. Mathematically this is written as

$$c_{ij} = p_{c_{ij}}\left(\boldsymbol{q}_{2,j} - \boldsymbol{q}_{1,j} + \boldsymbol{n}_{ij} = L_v\frac{\boldsymbol{v}}{z}dt\right) \tag{12}$$

where $\boldsymbol{n}_{ij}$ is measurement noise. Since there are finitely many possible correspondences for $\boldsymbol{q}_{2,j}$ (one of the points in Q_1 or no correspondence), $p_{c_{ij}}$ defines a discrete distribution. To find the MAP velocity we need to find a way to efficiently compute $p_{c_{ij}}$ using only the prior distributions of $\boldsymbol{v}$ and z.

1) Distribution of the Change in Position - 1D case: Defining $\boldsymbol{d} = L_v dt \boldsymbol{v} z^{-1}$, we need to find its distribution. First consider the 1D case:

$$\begin{aligned} d_x &= \frac{\delta_x}{z} \\ \delta_x &\triangleq q_x v_z dt - f v_x dt \end{aligned} \tag{13}$$

From this we see that d_x is distributed as the ratio of normally distributed random variables, which does not result in another normally distributed random variable.

Instead we use a normal approximation. The random variable $\Delta_x Z^{-1}$, from which $\delta_x z^{-1}$ is drawn, does not have well-defined moments so we use a two step method to produce the normal approximation: 1) define a new variable $\widehat{z}$ such that $\delta_x \widehat{z}^{-1} \simeq \delta_x z^{-1}$ and $\widehat{z}$ has well-defined moments, and 2) compute the moments of $\delta_x \widehat{z}^{-1}$ to make the normal distribution approximation. First, assuming $|\mu_z - z| < |\mu_z|$ we can write $\delta_x z^{-1}$ using a Taylor series expansion about μ_{δ_x} and μ_z.

$$\begin{aligned} \frac{\delta_x}{z} =& \frac{\mu_{\delta_x} + (\delta_x - \mu_{\delta_x})}{\mu_z + (z - \mu_z)} \\ =& \sum_{k=0}^{\infty} \frac{\mu_{\delta_x}(\mu_z - z)^k}{\mu_z^{k+1}} + \\ & \sum_{k=1}^{\infty} \frac{\partial^k\{\delta_x z^{-1}\}}{\partial\delta_x \partial^k z^{-1}} \frac{(\delta_x - \mu_{\delta_x})(z - \mu_z)^k}{(k+1)!} \end{aligned} \tag{14}$$

Proof of convergence for this approximation is given in [3].

Trying to take the expectation of eqn 14 directly would result in a divergent infinite sum. Instead, to create a new random variable with well-defined moments that closely approximates $\delta_x z^{-1}$, we truncate the Taylor series in equation 14 to the first k' terms.

$$\begin{aligned} \frac{\delta_x}{\widehat{z}} =& \sum_{k=0}^{k'} \frac{\mu_{\delta_x}(\mu_z - z)^k}{\mu_z^{k+1}} + \\ & \sum_{k=1}^{k'} \frac{\partial^k\{\delta_x z^{-1}\}}{\partial x \partial^k z^{-1}} \frac{(\delta_x - \mu_{\delta_x})(z - \mu_z)^k}{(k+1)!} \end{aligned} \tag{15}$$

Since this is now a finite sum, the expectation is well-defined and computed by

$$\mathbb{E}\left(\frac{\Delta_x}{\widehat{Z}}\right) = \sum_{k=0}^{\frac{k'}{2}} \frac{\mu_{\delta_x}\mathbb{E}\left((\mu_z - z)^k\right)}{\mu_z^{k+1}} = \frac{\mu_{\delta_x}}{\mu_z} + \sum_{j=1}^{\frac{k'}{2}} \frac{\mu_{\delta_x}\sigma_z^{2j}(2j-1)!!}{\mu_z^{2j+1}} \tag{16}$$

If $\sigma_z \ll \mu_z$ the summands in eqn 16 will quickly approach zero and k' can be chosen to give both an accurate and stable[1] approximation.

We also need to repeat this for the second moment, for which it is easiest to first look at z^{-2} and then use, by independence, $\mathbb{E}(\Delta_x^2\widehat{Z}^{-2}) = \mathbb{E}(\Delta_x^2)\mathbb{E}(\widehat{Z}^{-2})$.

$$\begin{aligned}\frac{1}{z^2} &= \frac{1}{(\mu_z + (z-\mu_z))^2} \\ &= \frac{1}{\mu_z^2} - \frac{2(z-\mu_z)}{\mu_z^3} + \frac{3!(z-\mu_z)^2}{\mu_z^4 2} - \dots \\ &= \sum_{k=0}^{\infty} \frac{(k+1)(\mu_z - z)^k}{\mu_z^{k+2}} \end{aligned} \tag{17}$$

$$\mathbb{E}\left(\frac{1}{\widehat{Z}^2}\right) = \sum_{k=0}^{k'} \frac{(k+1)\mathbb{E}\left((\mu_z - z)^k\right)}{\mu_z^{k+2}} = \frac{1}{\mu_z^2} + \sum_{j=1}^{\frac{k'}{2}} \frac{(2j+1)\sigma_z^{2j}(2j-1)!!}{\mu_z^{2j+2}} \tag{18}$$

Proof of convergence for eqn 17 is also given in [3].

The normal distribution approximation for d_x is then defined by

$$\begin{aligned} p_{d_x} &\sim \mathcal{N}\left(\mu_{d_x}, \sigma_{d_x}^2\right) \\ \mu_{d_x} &= \mathbb{E}(\Delta_x Z^{-1}) \\ \sigma_{d_x}^2 &= \mathbb{E}(\Delta_x^2 Z^{-2}) - \mathbb{E}^2(\Delta_x Z^{-1}) \end{aligned} \tag{19}$$

2) Distribution of the Change in Position - 2D case: Extending the results of the previous section to the 2D case is straightforward. The marginal distributions p_{d_x} and p_{d_y} are directly calculated using eqn 19, which just leaves the covariance to be determined. To calculate this we first need to find $\mathbb{E}(\delta_x\delta_y)$. From eqn 13,

$$\begin{aligned} \delta_x\delta_y &= q_x q_y dt^2 v_z^2 - q_x f dt v_y v_z - \\ &\quad q_y f dt v_x v_z + f^2 v_x v_y dt^2 \\ \mathbb{E}(\Delta_x\Delta_y) &= q_x q_y dt^2 \sigma_{v_z}^2 + \mu_{\delta_x}\mu_{\delta_y} \end{aligned} \tag{20}$$

Using this and the assumption of independence between $\Delta_x\Delta_y$ and Z^{-1}, the covariance is

$$\begin{aligned} \mathrm{cov}(D_x, D_y) &= \mathbb{E}(\Delta_x\Delta_y Z^{-2}) - \mathbb{E}(\Delta_x Z^{-1})\mathbb{E}(\Delta_y Z^{-1}) \\ &= (q_x q_y dt^2\sigma_{v_z}^2 + \mu_{\delta_x}\mu_{\delta_y})\mu_{z^{-2}} - \mu_{\delta_x}\mu_{\delta_y}\mu_{z^{-1}}^2 \\ &= q_x q_y dt^2 \sigma_{v_z}^2 \mu_{z^{-2}} + \mu_{\delta_x}\mu_{\delta_y}(\mu_{z^{-2}} - \mu_{z^{-1}}^2) \end{aligned} \tag{21}$$

[1]Stable in the sense that the value is locally nearly independent of the specific choice of k'

and the normal approximation for the joint distribution is

$$\begin{aligned} p_d &\sim \mathcal{N}(\boldsymbol{\mu}_d, \Sigma_d) \\ \boldsymbol{\mu}_d &= \begin{bmatrix}\mu_{d_x} & \mu_{d_y}\end{bmatrix}^\intercal \\ \Sigma_d &= \begin{bmatrix} \sigma_{d_x}^2 & \mathrm{cov}(D_x, D_y) \\ \mathrm{cov}(D_x, D_y) & \sigma_{d_y}^2 \end{bmatrix} \end{aligned} \tag{22}$$

3) Correspondence Probabilities: Using the normal approximation of $\boldsymbol{d}$ from the previous section, we are now ready to calculate the correspondence probability in eqn 12. Using Baye's rule, we can write this as

$$c_{ij} = \frac{p_{d_i}(\boldsymbol{q}_{2,j} - \boldsymbol{q}_{1,i})p_f(\boldsymbol{q}_{1,i})}{\sum_{k=1}^{N_1} p_{d_k}(\boldsymbol{q}_{2,j} - \boldsymbol{q}_{1,k})p_f(\boldsymbol{q}_{1,k})} \tag{23}$$

where $p_f(\boldsymbol{q}_{1,i})$ is the probability that the feature located at $\boldsymbol{q}_{1,i}$ really exists, which we assume is uniformly distributed so cancels out of eqn 23.

To account for measurement error in $\boldsymbol{q}_{2,j}$, assumed be be normally distributed, we need to integrate over the measurement distribution.

$$c_{ij} = \alpha_j \int_{\mathbb{R}^2} p_{d_i}(\boldsymbol{q} - \boldsymbol{q}_{1,i}|\boldsymbol{q}) p_{q_{2,j}}(\boldsymbol{q}) d\boldsymbol{q} \tag{24}$$

Solving this integral gives

$$\begin{aligned} c_{ij} &= \frac{\alpha_j\sqrt{|\Sigma_a|}}{2\pi\sqrt{|\Sigma_d\Sigma_n|}} e^{-\frac{f}{2}} \\ f &\triangleq -\boldsymbol{\mu}_a^\intercal\Sigma_a^{-1}\boldsymbol{\mu}_a + (\boldsymbol{\mu}_d + \boldsymbol{q}_{1,i})^\intercal\Sigma_d^{-1}(\boldsymbol{\mu}_d + \boldsymbol{q}_{1,i}) + \\ &\quad \boldsymbol{\mu}_{q_{2,j}}^\intercal \Sigma_n^{-1}\boldsymbol{\mu}_{q_{2,j}} \\ \boldsymbol{\mu}_a &\triangleq \Sigma_a(\Sigma_d^{-1}(\boldsymbol{\mu}_d + \boldsymbol{q}_{1,i}) + \Sigma_n^{-1}\boldsymbol{\mu}_{q_{2,j}}) \\ \Sigma_a &\triangleq (\Sigma_d^{-1} + \Sigma_n^{-1})^{-1} \end{aligned} \tag{25}$$

The remaining unknown, α_j, is found by requiring

$$\sum_{i=1}^{N_1} c_{ij} = 1 \tag{26}$$

Finally, to account for the possibility of a point not having a correspondence, we introduce the virtual point $\boldsymbol{q}_{1,N_1+1}$ in Q_1 whose time integrated position, $\boldsymbol{d}_{N_1+1}$, follows a uniform probability distribution. Note that eqn 26 should now sum over $i = 1 \dots N_1 + 1$.

B. MAP Velocity and Height Estimation

Using the correspondence results of the previous section, estimation of the MAP velocity and height is a straightforward application of standard Bayesian inference tools. Our goal is to find the velocity and height that maximizes the joint probability distribution, i.e.

$$\boldsymbol{v}^*, z^* = \arg\max_{\boldsymbol{v},z} p(\boldsymbol{v}, z|Q_1, Q_2, C) \tag{27}$$

where $\boldsymbol{v}^*$ and z^* are the respective maximizers.

We can use Baye's rule to rearrange this into distributions that are more easily defined.

$$
\begin{aligned}
&p(\boldsymbol{v}, z|Q_1, Q_2, C) = p(\boldsymbol{v}, z|Q_1, Q_2, C, z)p(z|Q_1, Q_2, C) \\
&= \frac{p(Q_2|Q_1, \boldsymbol{v}, z, C)p(\boldsymbol{v}|Q_1, C, z)}{p(Q_2|Q_1, z, C)} \frac{p(Q_2|Q_1, z, C)p(z|Q_1, C)}{p(Q_2|Q_1, C)} \\
&= \frac{p(Q_2|Q_1, \boldsymbol{v}, z, C)p(\boldsymbol{v}|Q_1, C, z)p(z|Q_1, C)}{p(Q_2|Q_1, C)} \qquad (28)
\end{aligned}
$$

Under independence assumptions we can write

$$p(\boldsymbol{v}|Q_1, C, z) = p_v(\boldsymbol{v}) \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma_v) \qquad (29)$$

$$p(z|Q_1, C) = p_z(z) \sim \mathcal{N}(\mu_z, \sigma_z^2) \qquad (30)$$

where p_v and p_z are the respective prior distributions. Further assuming that the measurement of each point in Q_2 is independent of the other points in Q_2 we can write

$$
\begin{aligned}
&p(Q_2|Q_1, \boldsymbol{v}, z, C) = \Pi_{j=1}^{N_2} p_n(\boldsymbol{n}_j)^{1-c_{N_1+1,j}} a^{c_{N_1+1,j}} \qquad (31) \\
&\boldsymbol{n}_j = \sum_{i=1}^{N_1} c_{ij} \left(\boldsymbol{q}_{2,j} - \boldsymbol{q}_{1,i} - L_{\omega,i}\boldsymbol{\omega} dt - L_{v,i}\frac{\boldsymbol{v}}{z}dt \right) \\
&p_n \sim \mathcal{N}(0, \Sigma_n)
\end{aligned}
$$

where $\boldsymbol{n}_j$ is measurement noise and a is the uniform *a priori* probability density of a point having no correspondence. This seems similar to $c_{N_1+1,j}$, but recall that $c_{N_1+1,j}$ is the probability that the point at $\boldsymbol{q}_{2,j}$ is associated with the virtual point in Q_1, i.e. the *a posteriori* probability that the specific point $\boldsymbol{q}_{2,j}$ has no real correspondence.

Rather than directly optimizing eqn 27 it is mathematically more convenient to optimize the log probability, which has the same maximizers. The optimization objective function, J, then becomes

$$
\begin{aligned}
J &= \ln p(Q_2|Q_1, \boldsymbol{v}, z, C) + \ln p_v(\boldsymbol{v}) + \ln p_z(z) + \ldots \\
&= -\frac{1}{2} \left[\sum_{j=1}^{N_1} (1 - c_{N_1+1,j}) \boldsymbol{n}_j^\intercal \Sigma_n^{-1} \boldsymbol{n}_j + \right. \\
&\qquad \left. (\boldsymbol{v} - \boldsymbol{\mu}_v)^\intercal \Sigma_v^{-1} (\boldsymbol{v} - \boldsymbol{\mu}_v) + \frac{(z - \mu_z)^2}{\sigma_z^2} \right] + \ldots \qquad (32)
\end{aligned}
$$

where the omitted terms do not contain any variables being optimized over.

Eqn 32 is quadratic in $\boldsymbol{v}$ which, if z is known, admits the unique solution

$$
\begin{aligned}
\boldsymbol{v}^{*\intercal} &= \left(\frac{1}{z}\boldsymbol{s}_1^\intercal + \boldsymbol{\mu}_v^\intercal \Sigma_v^{-1} \right) \left(\frac{1}{z^2} S_2 + \Sigma_v^{-1} \right)^{-1} \qquad (33) \\
\boldsymbol{s}_1^\intercal &\triangleq \sum_{j=1}^{N_2} (1 - c_{N_1+1,j}) \left(\sum_{i=1}^{N_1} c_{ij} (\boldsymbol{q}_{2,j} - \boldsymbol{q}_{1,i} - \right. \\
&\qquad \left. L_{\omega,i}\boldsymbol{\omega}) \right)^\intercal \Sigma_n^{-1} A_j \\
S_2 &\triangleq \sum_{j=1}^{N_2} (1 - c_{N_1+1,j}) A_j^\intercal \Sigma_n^{-1} A_j \\
A_j &\triangleq dt \sum_{i=1}^{N_1} c_{ij} L_{v,i}
\end{aligned}
$$

From eqn 33 we can see that when the feature measurement noise is large, and, hence, $\boldsymbol{s}_1$ and S_2 are small, the MAP estimate of $\boldsymbol{v}$ will be close to the prior $\boldsymbol{\mu}_v$. As the measurement noise decreases relative to Σ_v, the MAP estimate approaches the least-squares estimate using the visual information only, as expected.

For the general optimization over both $\boldsymbol{v}$ and z, we can use eqn 33 to reduce the numerical optimization problem to that of a single scalar, z. Rewriting eqn 32 using constant matrices,

$$
\begin{aligned}
J = &-\frac{1}{2} \left[s_0 - \frac{2}{z}\boldsymbol{s}_1^\intercal \boldsymbol{v} + \frac{1}{z^2} \boldsymbol{v}^\intercal S_2 \boldsymbol{v} + \right. \\
&\left. (\boldsymbol{v} - \boldsymbol{\mu}_v)^\intercal \Sigma_v^{-1} (\boldsymbol{v} - \boldsymbol{\mu}_v) + \frac{(z - \mu_z)^2}{\sigma_z^2} \right] + \ldots \qquad (34) \\
s_0 \triangleq &\sum_{j=1}^{N_2} (1 - c_{N1+1}) \left(\sum_{i=1}^{N_1} c_{ij} (\boldsymbol{q}_{2,j} - \boldsymbol{q}_{1,i} - L_{\omega,i}\boldsymbol{\omega}) \right)^\intercal \Sigma_n^{-1} \\
&\left(\sum_{i=1}^{N_1} c_{ij} (\boldsymbol{q}_{2,j} - \boldsymbol{q}_{1,i} - L_{\omega,i}\boldsymbol{\omega}) \right)
\end{aligned}
$$

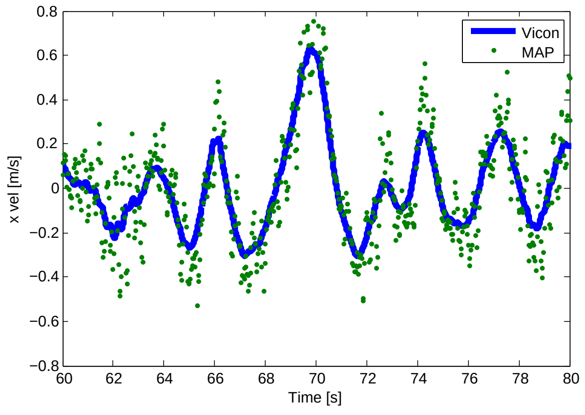
, we can easily apply any common numerical optimization technique. For the work here, we simply perform a coarse grid search over the range of expected heights ($\mu_z \pm 3\sigma_z$) and then use a golden section search [7] to refine the estimate. Most of the computational effort is in building up s_0, $\boldsymbol{s}_1$, and S_2 so the specific choice of numerical optimization technique is not critical.

VI. Experimental Results

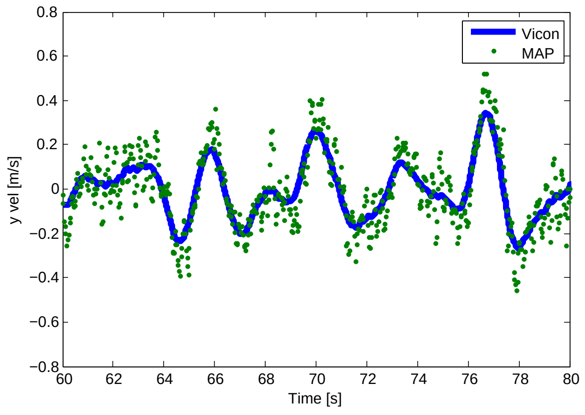
We have implemented the above algorithms on a Galaxy SIII smartphone. All computations are done onboard using only onboard sensors for nearly everything. We are still in the process of researching and implementing onboard position and yaw measurement algorithms so the current results use Vicon to simulation those results. The Vicon data is limited to 10Hz and has 1cm standard deviation noise artificially added. Since our previous analysis [3] showed that the velocity estimation algorithm performed well with relatively few feature points, we use an image size of 320px $\times$ 240px.

Figure 2 shows that the velocity estimation algorithm outlined in section V works well for a sample fight. Note that little effort was spent to tune the Kalman filter for this flight so better overall results could be achieved with a more accurate Kalman filter. There will always be some error, though, so the velocity estimation algorithm of section V is still useful.

Figures 3 and 4 show the computation performance of the smartphone. In figure 3 we see that most velocity estimates are ready in less than 30ms. This is not only the image processing time, but also any overhead due the operating system or other running processes such as the attitude observer and controller, translation observer and controller, etc. Figure 4 shows that the current flight controller is only using around 50% of the available cpu power. This suggests



(a) x-vel



(b) y-vel

Fig. 2: Velocity estimation performance. "Vicon" is the velocity measured using the Vicon system. "MAP" is the onboard velocity measurement using the algorithm described in this section V. For this flight, the height was held relatively constant so z and $\dot{z}$ plots are omitted.

that there is still sufficient resources available to support onboard position estimation.

An example flight video is available at `http://db.tt/2I7LwMEg`

VII. Conclusion

In this work we showed some of the algorithms necessary to enable autonomous flight using a smartphone flight controller. Additionally, we demonstrated an online implementation of a velocity estimation algorithm that we recently proposed. We are currently in the process of researching and implementing algorithms for position estimation and control which will enable us to remove the last elements of reliance on external sensing.

Acknowledgement

This work was supported by the New and Renewable Energy Program of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea government Ministry of Knowledge Economy.

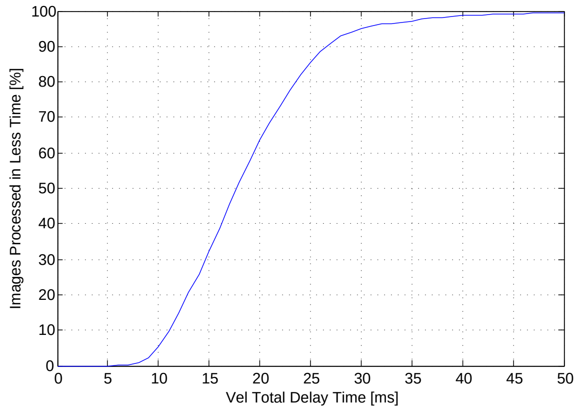


Fig. 3: Total vision delay time. The horizontal axis represents the time elapsed between image capture and the velocity estimate being ready, which includes all vision processing steps plus operating system overhead due to other processing. The vertical axis is the percentage of images with less delay than the respective horizontal axis value, e.g. 95% of the images had a total delay of less than 30ms.

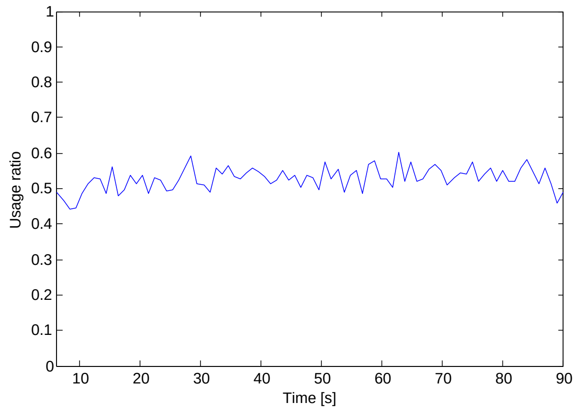


Fig. 4: Average cpu usage, over all 4 cores, during flight

References

[1] M. Achtelik, S. Weiss, and R. Siegwart, "Onboard imu and monocular vision based control for mavs in unknown in-and outdoor environments," in *Robotics and automation (ICRA), 2011 IEEE international conference on*. IEEE, 2011, pp. 3056–3063.

[2] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision," *Autonomous Robots*, vol. 33, no. 1-2, pp. 21–39, 2012.

[3] T. Ryan and H. J. Kim, "Image-space prior distributions and bayesian inference for robust UAV velocity estimation from video sequences," in *under review*.

[4] Mikrokopter website. [Online]. Available: http://www.mikrokopter.de/ucwiki/en/MikroKopter

[5] T. Hamel and R. Mahony, "Attitude estimation on SO3 based on direct inertial measurements," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 2170–2175.

[6] T. Ryan and H. J. Kim, "PD-tunable control design for a quadrotor," in *AIAA Guidance, Navigation, and Control (GNC), 2013 Conference on*. AIAA, 2013.

[7] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, 3rd ed. Springer, 2008.