2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Versatile Distributed Pose Estimation and Sensor Self-Calibration for an Autonomous MAV

Stephan Weiss, Markus W. Achtelik, Margarita Chli, Roland Siegwart

Abstract— **In this paper, we present a versatile framework to enable autonomous flights of a Micro Aerial Vehicle (MAV) which has only slow, noisy, delayed and possibly arbitrarily scaled measurements available. Using such measurements directly for position control would be practically impossible as MAVs exhibit great agility in motion. In addition, these measurements often come from a selection of different onboard sensors, hence accurate calibration is crucial to the robustness of the estimation processes. Here, we address these problems using an EKF formulation which fuses these measurements with inertial sensors. We do not only estimate pose and velocity of the MAV, but also estimate sensor biases, scale of the position measurement and self (inter-sensor) calibration *in real-time*. Furthermore, we show that it is possible to obtain a yaw estimate from position measurements only. We demonstrate that the proposed framework is capable of running entirely onboard a MAV performing state prediction at the rate of 1 kHz. Our results illustrate that this approach is able to handle measurement delays (up to 500ms), noise (std. deviation up to 20 cm) and slow update rates (as low as 1 Hz) while dynamic maneuvers are still possible. We present a detailed quantitative performance evaluation of the real system under the influence of different disturbance parameters and different sensor setups to highlight the versatility of our approach.**

I. Introduction

The research in autonomous micro helicopters is advancing and evolving rapidly leading to great progress over the past few years. However, current solutions lack the robustness and flexibility to general conditions that is required in order to leave the controlled laboratory environments. As a result, we are yet to see truly autonomous flights in general environments without reliance on unrealistic assumptions like uninterrupted GPS signal, perfect communication link to a ground station for off-board processing/control or (almost perfect) pose measurements from motion capture systems (e.g. Vicon). Only after solving these issues, higher level tasks such as autonomous exploration, swarm operation and large trajectory planning can be tackled.

Autonomous flights in unknown environments exclude the use of motion capture systems. Using GPS is not always reliable due to effects like shadowing or multipath in city-like environments. Therefore, commonly used sensors for pose estimation are stereo and monocular cameras as well as laser scanners. Due to the power, weight and computation limitations inherent in the field of MAVs, the sensor-feeds and the algorithms that process them usually yield slow and noisy update rates including delays, and in the case of monocular vision, unknown scale of position measurements. However, an accurate and timely pose and velocity estimate is crucial for proper and safe autonomous flights of a MAV.

Fig. 1: The hexacopter which is used for the experimental analysis during an autonomous flight with the proposed framework

Measurements from GPS or a Leica Total Station[1] might be limited to position information only. In order to perform position control, errors must be transformed from world- to body-fixed coordinates such that the appropriate corrections can be computed. It is therefore essential to have an estimate of the heading (yaw) of the helicopter. Using a magnetic compass for yaw measurements is often not reliable enough and prone to disturbances. In this work, we show that it is possible to estimate yaw from body accelerations and world-fixed position measurements only.

For real systems, it is impractical having to re-run the calibration module every time the sensor suite of the MAV is to be used. It is, however, important to estimate the biases of the IMU and the inter-sensor calibration parameters (between IMU and any additional sensors). In the proposed framework, we apply the findings in [1], [2], [3] and show that we can estimate these calibration parameters *online*. Some arbitrary movements at the beginning of the flight are sufficient for the convergence of these calibration parameters. As a result, the MAV sensor suite is rendered truly power-on-and-go.

The aim of this work is to provide a versatile and modular EKF framework which tackles pose estimation and inter-sensor calibration not only as a theoretical construct, but also realize this on a working MAV system operating in real-time. Our approach achieves state prediction at 1 kHz while the framework is kept modular to allow any kind of position/pose measurements to be incorporated quickly.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7) under grant agreements n. 231855 (sFly) and n.266470 (myCopter). Stephan Weiss and Markus Achtelik are currently PhD students and Margarita Chli is a senior researcher at the ETH Zurich (email: {stephan.weiss, markus.achtelik, margarita.chli}@mavt.ethz.ch). Roland Siegwart is full professor at the ETH Zurich and head of the Autonomous Systems Lab (email: r.siegwart@ieee.org).

[1] or generally, an external system measuring the 3D position of the MAV

II. Related Work

The weight and power constraints on MAVs limit the choices of sensors for accurate MAV state estimation. Hence, in the following review we focus on the use of cameras, GPS and IMU sensors to achieve this task.

A promising approach addressing IMU and camera calibration appears in [3], which was recently refined by Kelly and Sukhatme in [2] as a major step towards power-on-and-go systems. In the latter work, the authors used all information from the two sensors to statistically optimally estimate the 6DoF IMU pose, the inter-sensor calibration, the visual scale factor arising in monocular systems, and the gravity vector in the Vision frame. In [1], we described an incremental improvement to make this approach modular and suitable for multiple sensors. From theory to practice however, there is a big step, hence in this paper, we realize our proposed approach on a real system performing all processing onboard and in real-time, underlining the modularity, flexibility and robustness which are crucial for real world applications.

Real-time visual-inertial navigation using an iterated EKF has been shown in [4] exhibiting a complexity which grows at least quadratically with the number of features, rendering the method unsuitable for large-scale robot navigation. The more efficient approaches in [5] and [6] considered pairwise images for visual odometry and fused the output with inertial measurements in an EKF. The authors also used an additional altimeter for solving for the unknown scale of the vision algorithm. Their proposed EKF approach is linear in the number of features and quadratic in the number of included camera poses. It is worth mentioning, however, that none of these works solve for inter-sensor calibration nor provide the possibility of augmenting the system with additional sensors like the work in this paper.

Along similar lines as the work of Kelly and Sukhatme, are the works in [7] and [8], where the IMU is tightly coupled into an EKF SLAM framework with the extension to new features. While the authors in [7] showed an accurate large outdoor trajectory using their approach on a car, the authors in [8] made use of a hand-held device to estimate the metric depth of a natural scene. Regarding the filtering strategy, these approaches are robust as they take all cross-couplings of the observed features into account. However, the computational cost of EKF SLAM is $O(N^2)$ for N features, preventing long-term runs unless sub-mapping is performed though at the cost of losing cross-couplings. In [9], the camera, IMU and GPS measurements are fused in an EKF SLAM framework, only their formulation leads to a state dimension of 10^4 which is clearly unmanageable on a platform with limited calculation power. Moreover, they assume a known inter-sensor calibration.

In [10], the authors present a commercially available GPS/INS solution also accounting for the car's heading (yaw) angle. Their post-mission (i.e. offline) implementation also estimates the vector between IMU and GPS sensor. The sensors used in that work are not bound to strong weight constraints and are sufficiently precise to project the Earth's rate vector onto the rate gyro to determine heading.

Perhaps the most relevant work is [1], where we developed the theory for our filtering framework. In this paper, we perform an observability analysis for obtaining yaw using a position-only measurement and then focus on the practical implementation and the quantitative evaluation of our framework. The assessment of the performance and practicality of this EKF framework is crucial as it can only be performed following its actual implementation on a real system – the true effect of all the assumptions revealed and any inaccuracies due to modeling/linearization are identified only then. The most relevant caveats are the assumptions of a Gaussian, linear and discrete world. Even though the sensor readings may be well-approximated by Gaussian distributions, they certainly get distorted after applying the non-linear equations. Moreover, discretization effects may lead to instability of the system. Hence, we conduct a thorough quantitative evaluation to reveal consistent operation of a highly robust and modular filtering framework, running online and onboard a MAV.

III. System Setup

The MAV we use is a prototype of a hexacopter from Ascending Technologies[2] (AscTec) as it is shown in Fig. 1. It is a helicopter driven by six rotors, symmetric to the center of mass. The control is performed solely by changing the rotation speed of the rotors. Equipment and behavior is similar to the "AscTec Pelican" quadrocopter, which is described in detail in [11]. The key features are up to 500 g payload, improved vibration damping for the sensors and the Flight Control Unit (FCU) "AscTec Autopilot". This FCU features a complete IMU and two $32\,\mathrm{Bit}$, $60\,\mathrm{MHz}$ ARM-7 microcontrollers used for data fusion and flight control.

The so-called Low Level Processor (LLP) is considered as a black box which manages the hardware and performs IMU sensor data fusion for attitude control. The other microcontroller, the High Level Processor (HLP) is dedicated for custom code. IMU data is provided at an update rate of $1\,\mathrm{kHz}$ via a high speed serial interface from the LLP. In particular, this comprises body accelerations, body angular velocities, magnetic compass, height measured by an air pressure sensor and the estimated attitude of the vehicle. On the HLP, a position controller based on nonlinear dynamic inversion and reference model based setpoint following is implemented which we described in detail in [11].

For the computationally more expensive onboard processing tasks, we outfitted the helicopter with an onboard 1.6 GHz Intel Atom Based embedded computer, available from Ascending Technologies. This computer is equipped with 1 GB RAM, a MicroSD card slot for the operating system, 802.11n WiFi, a Compact Flash slot, USB 2.0 interfaces for external sensors and serial ports for communication to the HLP. We run Ubuntu Linux 10.04 on our onboard

[2] www.asctec.de

computer and use the ROS [3] framework as a middleware for communication, parameters and monitoring of our processes.

Also crucial for the whole system to work is accurate time synchronization between all involved parts. Ground station(s) and Atom-computer are synchronized via the Network Time Protocol (NTP) whereas between Atom-computer and HLP time synchronization packets are exchanged. Transfer delay and jitter are then estimated in a NTP like way.

IV. State Estimation for MAVs

An EKF framework generally consists of a prediction and an update step. As we describe later, we use this to distribute the computational load to different units on the MAV. We first give an overview of the underlying structure of the EKF framework in [1] and then discuss the different measurements a sensor can yield as an update.

A. Inertial Sensor Model

We assume that the inertial measurements contain a certain bias b and white Gaussian noise n. Thus, for the real angular velocities ω and the real accelerations a we have

$$\omega = \omega_m - b_\omega - n_\omega \qquad a = a_m - b_a - n_a \tag{1}$$

The subscript m denotes the measured value. The dynamics of the non-static bias b are modeled as a random process:

$$\dot{b_\omega} = n_{b_\omega} \qquad \dot{b_a} = n_{b_a} \tag{2}$$

B. State Representation

The state of the filter is composed of the position of the IMU p_w^i in the inertial world frame, its velocity v_w^i and its attitude quaternion q_w^i describing a rotation from the inertial to the IMU frame. We also add the gyro and acceleration biases b_ω and b_a as well as a possible measurement scale factor λ. The calibration states are the rotation from the IMU frame to the measurement sensor frame q_i^s and the distance between these two sensors p_i^s. Note that the calibration states can be omitted and set to a calibrated constant making the filter more robust [7]. For completeness we keep them in the filter state. This yields a 24-element state vector X:

$$X = \{p_w^{i^T} \; v_w^{i^T} \; q_w^{i^T} \; b_\omega^T \; b_a^T \; \lambda \; p_i^s \; q_i^s\} \tag{3}$$

The following differential equations govern the state:

$$\dot{p}_w^i = v_w^i \tag{4}$$
$$\dot{v}_w^i = C_{(q_w^i)}^T (a_m - b_a - n_a) - g \tag{5}$$
$$\dot{q}_w^i = \frac{1}{2}\Omega(\omega_m - b_\omega - n_\omega) q_w^i \tag{6}$$

$$\dot{b}_\omega = n_{b_\omega} \quad \dot{b}_a = n_{b_a} \quad \dot{\lambda} = 0 \quad \dot{p_i^s} = 0 \quad \dot{q_i^s} = 0 \tag{7}$$

With g as the gravity vector in the world frame and $\Omega(\omega)$ as the quaternion multiplication matrix of ω. We assume the scale drifts spatially and not temporally, thus $\dot{\lambda} = 0$. Eq. (4) to Eq. (7) already reveal, that the base state prediction only needs very little computation power.

[3] www.ros.org

More details on the calculus of the system propagation and covariance matrices F_d and Q_d, respectively can be found in our previous work [1]. With the discretized error state propagation and error process noise covariance matrices, we can propagate the state as follows:

1) propagate the state variables following Eq. (4) to Eq. (7). For the quaternion, we use the 1^{st} order integration described in [12].
2) calculate F_d and Q_d as proposed in [1]
3) compute the propagated state covariance matrix according to the filter equation

$$P_{k+1|k} = F_d P_{k|k} F_d^T + Q_d \tag{8}$$

Note that the computational burden is on the prediction of the NxN state covariance matrix P with N states. We discuss this later on in Section V-A.

C. Measurement Models

The propagation model discussed above is the core in our EKF framework making the IMU an indispensable sensor of the system. However, we do not see this as restriction since almost all airborne navigation systems are able to bear an IMU. MEMS IMUs are cheap, small and lightweight such that they are very applicable on almost any robotic platform. Around this core sensor, we can have additional sensing modalities to enable robust MAV navigation. In particular, here we analyze a 6DoF pose sensor (e.g. a camera or Vicon system) and a 3DoF position sensor (e.g. GPS or laser tracker). Other sensors can also be used, given the observability analysis confirms the observability of the used states. Fig. 2 depicts the setup with the coordinate frames, sensors and state variables discussed below.

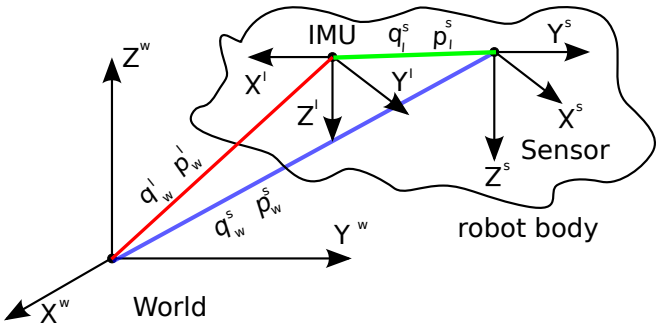

Fig. 2: Setup depicting the robot body with its sensors w.r.t. a world reference frame. The system's state as described in Section IV-B is $X = \{p_w^i \; v_w^i \; q_w^i \; b_\omega \; b_a \; \lambda \; p_i^s \; q_i^s\}$ whereas p_w^s and q_w^s denote the robot's sensor measurements in (a possibly scaled) position and attitude respectively in a world frame. Depending on the type of sensor, only one of the measurements is available (i.e. GPS or laser tracker only measure the 3D position p_w^s whereas a camera or Vicon system measures both).

1) Measurement from a 6DoF pose sensor: While the measurement of such a sensor has been addressed in [1] (camera), we state the most relevant information for completeness here.

For the possibly scaled camera position measurement p_w^s obtained from the visual algorithm, we have the following measurement model

$$z_p = p_w^s = (p_w^i + C_{(q_w^i)}^T p_i^s)\lambda + n_p \tag{9}$$

with $C_{(q_w^i)}$ as the IMU's attitude in the world frame.

For the rotation measurement we apply the notion of an error quaternion. The vision algorithm yields the rotation from the vision frame to the camera frame q_w^s. We can model this as

$$z_q = q_w^s = q_i^s \otimes q_w^i \tag{10}$$

A non-linear observability analysis as suggested in [13] and done in [2] reveals that all states are observable including the inter-sensor calibration states p_i^s (distance from the IMU to the sensor) and q_i^s (rotation from the IMU to the sensor). This is true as long as the robot excites the IMU's accelerometer and gyroscopes in at least two axes as proved in [2], [3].

2) Measurement from a 3DoF position: Given the modular setup of [1], we can also consider a different measurement sensor than a camera yielding the 6DoF pose. Most relevant to practical applications is a sensor yielding the 3DoF position of the robot in a world frame. Such a sensor might be GPS or a laser tracking system [4].

For the sensor's position measurement p_w^s we have the same measurement model as in Eq. (9) except that the frame indices s now indicate the frame of the 3DoF position sensor and not the camera frame.

Since such a position sensor only measures the position of the robot in a world frame, the rotation of the sensor w.r.t. the IMU is irrelevant. We omit thus the state q_i^s from the system and only keep the translational calibration state p_i^s (i.e. 3D translation from IMU to the sensor).

The non-linear observability analysis reveals that all states are observable including the inter-sensor calibration state p_i^s (distance from the IMU to the sensor) and also the absolute yaw angle of the robot. This is only true if we have excitation in at least two linear directions (i.e. acceleration in at least two axes). This is intuitively clear since the double-integrated IMU accelerometers contrast with the measured position of the measurement sensor. This contrast leads to observability of the yaw angle. As an illustrative example, we assume the IMU readings to yield positive acceleration in x and the robot's attitude is identical with the world frame (i.e. unit rotation). Double-integration of the IMU readings leads to a positive displacement in world x. If the position sensor measures a displacement in positive y in world, this indicates that our assumption of unit-rotation is 90° off in yaw. For disambiguation and full state observability, the observability analysis indicates the need of acceleration in more than one (i.e. minimal two) axes.

V. Implementation of Distributed State Estimation

In [1] we thoroughly developed the theory for our filter framework and recapitulated it in a condensed form above with the addition of the yaw observability using a position-only measurement. In this section, we present how the filter framework is implemented and how its tasks are distributed in our system.

A. Distribution of Processing Tasks

In Section IV-B we noted that the computational burden is in the prediction of the covariance matrix. Note that the measurements only imply a matrix inversion of the size of the number of measurements (e.g. 6×6 for a 6DoF pose sensor). In addition, the update frame-rate is usually much lower than the prediction frame-rate[5]. Here, we define three processing tasks: (a) the pure state prediction as in Eq. (4) to Eq. (7), (b) the covariance prediction as in Eq. (8) and (c) the full update step.

Task (a) is the least computationally demanding while it yields the most recent best guess of the system state (i.e. robot pose). For controlling a MAV, this is the most (time) critical part and should be executed at high rates with the least possible delay to allow fast response to disturbances or to enable dynamic flights. Since IMU data is available with almost no delay at a rate of 1 kHz at the user programmable "high level processor" (HLP) of our MAV (see Section III), we implemented this part on the HLP executed at a rate of 1 kHz. Note that this time critical part is ensured to be executed in real-time which softens some real-time constraints for the remaining tasks being executed on a standard non-real-time operating system.

The computationally more expensive tasks (b) and (c) are implemented separately on the onboard Atom computer. The HLP sends its state predictions to the Atom computer over a high speed serial link, while the Atom computer sends back state corrections every time a measurement update arrives. Crucial for this to work is accurate time synchronization between HLP and Atom computer which we implemented in a lightweight NTP like approach.

As it can be seen in Fig. 3, the current state predicted by the HLP is exchanged only at a rate of 100 Hz. This means that we predict the covariance at 100 Hz while the state is predicted at 1 kHz on the HLP. This is due to bandwidth limitations of the data link between HLP and Atom computer and of the relatively high computation cost for predicting a $N \times N$ covariance matrix. While this approximation might be problematic in theory, it did not cause any problems in practice since linearization errors in the period of 10 ms are negligible.

B. Handling Measurement Delays

An efficient method to handle measurement delays is crucial for robust and accurate state estimation. These delays usually occur due to computationally expensive processing tasks such as image processing on limited onboard hardware. However, the measurement update cannot be performed just when other processing is finished, instead it has to be aligned in time with the state prediction. One of the contributions of this work is a method that compensates for this, ensuring both computational efficiency and theoretical exactness.

[4]We use a Total Station laser tracking system from Leica yielding the 3DoF position with millimeter precision at about 7Hz and a range of over several hundred meters.

[5]In our setup using the Leica laser tracker, we run the prediction at IMU frame-rate of 1 kHz and the update at about 7 Hz.

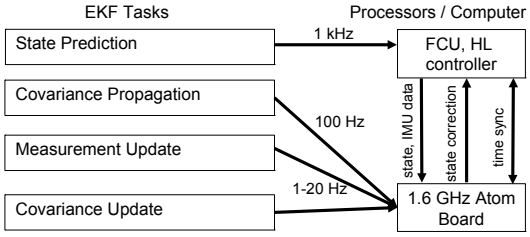

Fig. 3: Distribution of the processing tasks with their execution rates. State prediction, as the most time-critical part for controlling the helicopter, is executed on the IMU microcontroller (HLP) and is guaranteed to run in real-time at 1 kHz. This leaves enough time for the more complex parts to be computed on the Atom computer with a non-real-time operating system. Also very important for the whole system to work is accurate time synchronization.

Since both the HLP and Atom computer are synchronized, the timestamps of all sensors are referring to the same global time. Keeping a buffer of the S past states enables us to apply the obtained measurements at the exact time in the past they were taken. Thus the update is also theoretically exact, despite the delay. After performing the update step, the corrected state in the past is propagated again to the present time. The sequence in Fig. 4 depicts the process.

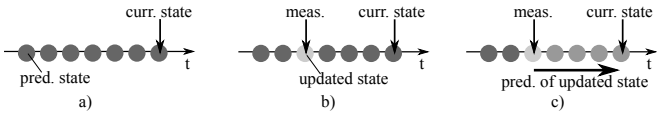

Fig. 4: Method to handle a time-delayed measurement. a) a given situation in time, the robot controller uses the latest propagated state as reference. b) a delayed measurement arrives and corrects the corresponding buffered state in the past. c) from the corrected state on, we correct all states in the buffer until the most recent one by propagating the corrected state according to the system's propagation equations Eq. (4) to Eq. (7).

Propagating the corrected state to the most recent state is computationally inexpensive for the state itself, but expensive for the covariance propagation. However, unless the most recent state uncertainty is used in other algorithms, there is no need to propagate the state covariance matrix to the present time. In fact, it is sufficient to have an up-to-date state covariance at the point in time where the latest measurement arrived. To minimize the computational effort and to avoid computation spikes, we suggest the following: as a measurement arrives, update the state and covariance at the time of the measurement in the past and then only recompute the state according to Fig. 4 until the present time, which is not expensive.

During regular state prediction, each time we predict a new state, we propagate the covariance matrix in the past to the next state in the past. That is, according to Fig. 5 while a state prediction e.g. x_6 is made, the propagation of the state covariance $P_{3|2}$ is computed.

If the measurements had a constant update rate and delay, the covariance prediction would be up-to-date at the correct point in the past where the latest measurement belongs to. This way, we distribute the covariance prediction optimally within the time between two measurements and have thus equally distributed processor load. In practice, the measurement delays vary slightly so we have to ensure an up-to-date covariance matrix by applying additional covariance prediction steps or ignoring redundant ones. This overhead is small in practical applications.

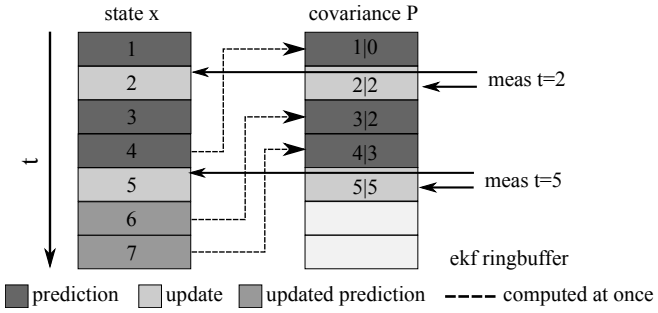

Fig. 5: For the computationally efficient state covariance handling we use two ring-buffers: one for the states and one for the covariance. Each has its separate data pointer. At a certain time t the state buffer pointer represents the present time, whereas the covariance buffer pointer points at the time the measurement corresponds to. Both state and covariance are updated at that point of time and the state is propagated to its current data pointer (present time) according to Fig. 4. As further states get propagated, the covariance is propagated accordingly in the past (t=4 to t=7). Upon new measurement, covariance and state in the past get updated and again the state is propagated to its current data pointer.

C. Handling Measurement Updates in Position Control

In the presence of low measurement update rates, noticeable correction steps occur, which in practice disturbs control of the MAV resulting in sudden excitations. Therefore, we still correct the state on the HLP in one step and perform state propagation from this correction, but we distribute the correction on the position controller over a constant period of time. In practice, smoothing the correction over 100 ms proved to be a good trade-off between precise setpoint following and smooth motion of the MAV. The results at a low update rate of 1 Hz can be seen in the right part of Fig. 9: while the state estimate gets corrected immediately every second, the motion of the helicopter still stays smooth. For larger corrections, such as for loop closure events, correction smoothing would not be sufficient. Therefore, we suggest performing a correction without smoothing and at the *same* time set the setpoint of the controller to the new corrected position. This avoids sudden movements and the large correction can be handled by a higher level task.

D. Modular Design

As suggested in Section IV-C, our framework has a core consisting of the propagation model using the dynamics acquired by the IMU and can be augmented with different types of measurement updates, i.e. this module remains unchanged using different sensors for measurement updates. This setup requires a modular software design capable of adapting to different and multiple sensors. As this framework, our ROS-based software implementation consists of a prediction core and allows different and multiple measurement modules to be added. Fig. 6 depicts the software architecture. In fact, to use another sensor to the system, it is only necessary to define the initialization values of the filter, the linearized measurement matrix H such that the estimated measurement $\hat{z} = H\vec{x}$ with $\vec{x}$ as the state vector and calculate the residual

$r = z - \hat{z}$. This renders our approach versatile and reusable for a wide range of applications and different sensors.

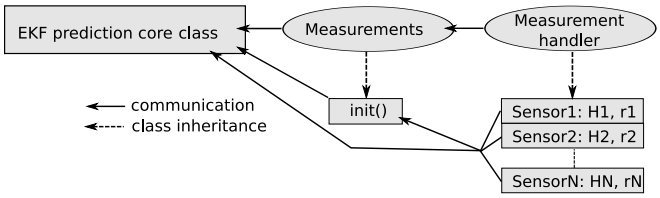

Fig. 6: Software architecture of the proposed framework. As described in Section IV-B, the core part of the framework is the EKF prediction module which remains unchanged using different measurement sensors. We add a base class to handle measurements in general including an initialization routine. A dedicated measurement handler class represents then the different measurement sensors. Each sensor has its specific update – it is only necessary to set the initialization values, define the linearized measurement matrix H and calculate the residual $r = z - \hat{z}$ for each sensor added to the system.

VI. Results

This section presents our experimental evaluation of the proposed framework and implementation. We give a quantitative evaluation using a Vicon motion capturing system providing us ground truth. We also use the Vicon system to provide measurements for our experiments to cancel out potential failures or systematic errors of a pose-estimator such as (Visual-)SLAM. To obtain measurements of similar quality as such algorithms, we reduce the rate of the Vicon measurements and add substantial noise and delay[6]. With this approach, we are able to relate the filter performance with exact values for each disturbance parameter: noise, delay and frame-rate.

For each test, we change one of these parameters. The default values are measurements at 10Hz, Vicon standard noise (mm accuracy) and Vicon standard delay (negligible). The distance p_i^s from the IMU to the additional sensor is measured as p_i^c = [0.015, -0.009, -0.028] and the corresponding attitude q_i^s as unit rotation. We conduct two main experiments: stationary hovering and a dynamic trajectory flight with all degrees of freedom of the helicopter involved. For all experiments, the state estimates (position, velocity and yaw) from the filter are used directly as input for the PID position controller [11] of the helicopter. A summary of the results can be seen in the accompanied video[7]

A. Stationary Hovering

First, we show the performance of the framework under hovering conditions. This is important to evaluate since the filter needs motion in order for all the states to be observable (Section IV). Observability ensures no drift on these states. For this experiment, we let the vehicle hover at a height of $1\,\mathrm{m}$ for $30\,\mathrm{s}$. We then compute the RMS error between the filter estimate and ground truth and between ground truth and the desired setpoint. The latter is done in order to show the performance of the whole system including the position controller. In particular, we make the following evaluations:

1) Ground truth against filter output for:
   a) position p_w^i and orientation q_w^i in Roll Pitch Yaw (rpy) convention
   b) scale λ of the position measurement in %
   c) position p_i^s and orientation q_i^s in Roll Pitch Yaw (rpy) convention of the sensor[8] w.r.t. the IMU.
2) Ground truth against controller setpoint: $x = 0$, $y = 0$, $z = 1\,\mathrm{m}$, $yaw = 0\,\mathrm{rad}$. In the tables in the appendix, this is indicated by the second vector in the columns for p_w^i and q_w^i.

All these experiments are performed with a 6DoF pose sensor (e.g. camera) and a 3DoF position sensor (e.g. Leica Total Station) in configurations as described in Section IV. We denote these setups as with (w/) and without (w/o) attitude measurements respectively in the tables in the appendix.

Table I shows the results of distorting the signal with noise. Note that the applied Gaussian noise gets distorted by the non-linear system becoming sub-optimal for EKF processes. We apply different noise levels on position and attitude ranging from a standard deviation of $5\,\mathrm{mm}$ and $0.5°$, to $20\,\mathrm{cm}$ and $2°$, respectively. In the same table, it is also visible that the inter-sensor distance p_i^s is the least accurate estimate. The observability analysis requires this state to be non-zero. We assume that the small magnitude of our chosen p_i^s together with the low excitations in hovering may in practice not be sufficient for this state to converge fully. The results of low noise with- and large noise without attitude update can also be seen in Fig. 7.

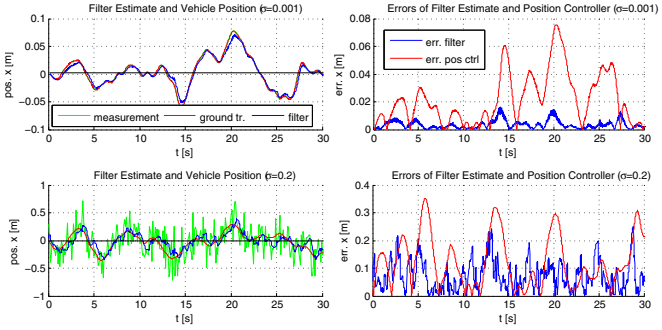

Fig. 7: Performance comparison for the "hovering" experiment with ($\sigma = 0.001\,\mathrm{m}$) w/ attitude update (top) and ($\sigma = 0.2\,\mathrm{m}$) w/o attitude update (bottom). Left: ground truth position of the helicopter (red), the filter estimate (blue) and the measurement the filter was updated with. Right: error of the filter w.r.t. ground truth (blue) and error of the desired trajectory w.r.t. ground truth (red) and setpoints. This is the overall error including the state estimator and the position controller of the helicopter. Note the different scaling of the plots.

In Table II we see the expected effect of delay introduced to the measurements. In our implementation, we have a state buffer of $2.5\,\mathrm{s}$. In theory, adding a delay of $500\,\mathrm{ms}$ should thus have minimal influence on the filter performance. This is evident from the results in Table II. The slight RMS increase in position p_w^i and attitude q_w^i comes from the fact that for the

[6]The tools we used can be found at `http://www.ros.org/wiki/vicon_bridge`

[7]A high resolution version is available at `http://www.youtube.com/watch?v=12x53pCkFoI`

[8]e.g. camera or laser scanner. In case of an external tracking system (Vicon, laser tracker), this is the pose of the reflector(s)

current state, the last measurement update was up to $500\,\mathrm{ms}$ in the past. During this time, the filter state "integrated away" using the IMU prediction model only. Again, the state p_i^s seems to be barely observable in this configuration.

In Table III we list the influence of the measurement rate on the filter performance. In theory, as long as we have information, the filter should converge. Again, the very slight increase of the RMS in position p_w^i and attitude q_w^i comes from the time between measurement updates where the filter can use the IMU prediction model only.

For all the experiments above, the scale estimate is very accurate. One may expect differently given that the scale is a hidden state and thus may compensate on the real system for linearization and non-Gaussian effects. We assume that the low excitations in hovering mode reduce these effects drastically while still giving enough information to actually observe the state.

B. Dynamic Flight

To show the performance of the filter and the whole system for dynamic flights, we let the helicopter fly a trajectory of the shape of a tilted ellipse (Fig. 8), keeping a track speed of $1\,\mathrm{m/s}$ while performing a full $360°$ turn around the z-axis. We use this trajectory as input for the reference-model based setpoint following described in [11]. Based on the given position input, this model generates the accelerations and velocities (left part of Fig. 8) such that the helicopter is physically able to stay on the trajectory. The accelerations are used for feed-forward control whereas an error controller finally compensates for disturbances, keeping the helicopter on the desired trajectories for velocity and position.

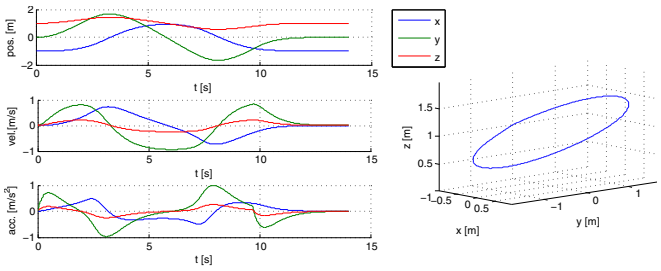

Fig. 8: The trajectory used for the dynamic flight experiments. The top and right plot show the desired position. Middle and bottom plots show the velocities and necessary accelerations to keep an absolute track speed of 1 m/s.

Table IV shows the results of different noise levels similar to the experiment in hovering mode. The values lie in the same range as for the hovering experiment if not slightly lower. Since the pose of the filter is a direct measure, this result is expected. Moreover, the inter-sensor calibration states seem to have a lower RMS value as well compared to the hovering experiments. In dynamic flight, the requirements of having angular velocities and linear accelerations in at least two axes are fulfilled. As a rule of thumb, it is good to have as much excitation of the system as the Nyquist theorem is still fulfilled given the measurement sensor reading rate. Thus, Table V and Table VI yield the expected results. Naturally, because of the issue of "integrating away" of the states on large delays or low update rates, the RMS values of the position are higher in dynamic flight than while hovering. The last two tests in Table VI may reflect the above issue of not fulfilling Nyquist's theorem. Hence the RMS rises accordingly. The results of two dynamic flights with 10 Hz and only 1 Hz update rate can also be seen in Fig. 9.

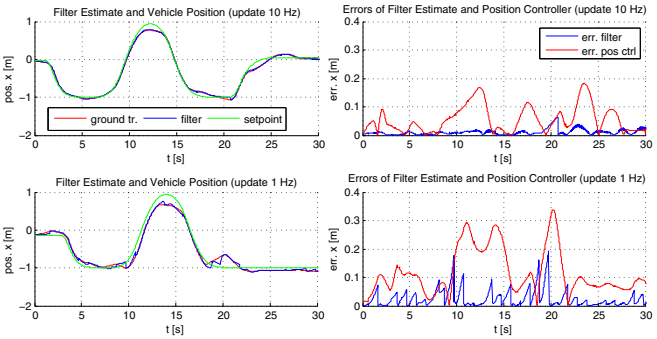

Fig. 9: Performance comparison for the "ellipse" experiment with 10 Hz (top) and only 1 Hz (bottom) update rate. Left: desired trajectory (green), ground truth position of the helicopter (red) and the filter estimate (blue). Right: error of the filter w.r.t. ground truth (blue) and error of the desired trajectory w.r.t. ground truth (red) and setpoints. This is the overall error including the state estimator and the position controller of the helicopter.

VII. Conclusions

Driven by the need for truly autonomous and robust navigation of MAVs, this paper presents a system for online inter-sensor calibration and 6DoF pose estimation in an implementation that runs entirely onboard a MAV (with an ATOM $1.6\,\mathrm{GHz}$ processor). Our results demonstrate the power and modularity of our framework using different sensors in addition to an IMU, while an observability analysis reveals that yaw estimation is possible from 3Dof position measurements only. The distributed EKF architecture we propose allows a MAV state prediction at $1\,\mathrm{kHz}$ for accurate control, while time-delays are compensated such that exact filter updates are ensured in a way that computational load is optimally distributed across updates. Our thorough quantitative evaluation shows that our approach is highly robust against noise, delay and slow measurement readings.

VIII. Acknowledgments

The authors would like to thank Simon Lynen for his help with producing the video and Prof. Daniela Rus and Prof. Nicholas Roy for hosting Stephan Weiss and Markus Achtelik in summer 2011 at CSAIL, MIT.

Appendix

In the following, detailed results from the experiments are shown. The values denote the RMS error between the filter estimate and ground truth, except for the second vector in the columns for p_w^i and q_w^i which denote the RMS error between setpoint and ground truth. (w/) and (w/o) denote experiments with and without attitude measurements respectively (see Section IV-C.1, Section IV-C.2).

A. Hovering

σ_p[m] σ_q[rad]	p_w^i[m]	q_w^i[rad (rpy)]	λ[%]	p_i^s[m]	q_i^s[rad]
0.001 0.001 w/ att.	$\begin{bmatrix}0.006\\0.003\\0.001\end{bmatrix}$ $\begin{bmatrix}0.017\\0.019\\0.005\end{bmatrix}$	$\begin{bmatrix}0.003\\0.010\\0.019\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.015\end{bmatrix}$	0.3	$\begin{bmatrix}0.052\\0.032\\0.010\end{bmatrix}$	$\begin{bmatrix}0.025\\0.019\\0.000\end{bmatrix}$
0.005 0.009 w/ att.	$\begin{bmatrix}0.011\\0.014\\0.003\end{bmatrix}$ $\begin{bmatrix}0.028\\0.032\\0.006\end{bmatrix}$	$\begin{bmatrix}0.006\\0.007\\0.010\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.016\end{bmatrix}$	0.4	$\begin{bmatrix}0.085\\0.024\\0.050\end{bmatrix}$	$\begin{bmatrix}0.026\\0.010\\0.000\end{bmatrix}$
0.010 0.018 w/ att.	$\begin{bmatrix}0.014\\0.008\\0.006\end{bmatrix}$ $\begin{bmatrix}0.038\\0.025\\0.008\end{bmatrix}$	$\begin{bmatrix}0.007\\0.009\\0.019\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.022\end{bmatrix}$	0.4	$\begin{bmatrix}0.054\\0.052\\0.029\end{bmatrix}$	$\begin{bmatrix}0.031\\0.018\\0.000\end{bmatrix}$
0.020 0.036 w/ att.	$\begin{bmatrix}0.017\\0.012\\0.010\end{bmatrix}$ $\begin{bmatrix}0.056\\0.036\\0.013\end{bmatrix}$	$\begin{bmatrix}0.006\\0.059\\0.034\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.033\end{bmatrix}$	0.4	$\begin{bmatrix}0.020\\0.049\\0.026\end{bmatrix}$	$\begin{bmatrix}0.023\\0.042\\0.000\end{bmatrix}$
0.050 0.036 w/ att.	$\begin{bmatrix}0.034\\0.025\\0.026\end{bmatrix}$ $\begin{bmatrix}0.104\\0.043\\0.033\end{bmatrix}$	$\begin{bmatrix}0.011\\0.016\\0.009\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.015\end{bmatrix}$	0.6	$\begin{bmatrix}0.016\\0.029\\0.082\end{bmatrix}$	$\begin{bmatrix}0.022\\0.010\\0.000\end{bmatrix}$
0.100 0.036 w/o att.	$\begin{bmatrix}0.055\\0.054\\0.039\end{bmatrix}$ $\begin{bmatrix}0.098\\0.099\\0.062\end{bmatrix}$	$\begin{bmatrix}0.021\\0.025\\0.230\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.229\end{bmatrix}$	0.8	$\begin{bmatrix}0.005\\0.054\\0.113\end{bmatrix}$	$\begin{bmatrix}0.000\\0.000\\0.000\end{bmatrix}$
0.200 0.036 w/o att.	$\begin{bmatrix}0.087\\0.098\\0.080\end{bmatrix}$ $\begin{bmatrix}0.150\\0.166\\0.096\end{bmatrix}$	$\begin{bmatrix}0.045\\0.016\\0.190\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.202\end{bmatrix}$	0.8	$\begin{bmatrix}0.035\\0.057\\0.058\end{bmatrix}$	$\begin{bmatrix}0.000\\0.000\\0.000\end{bmatrix}$

TABLE I: Measurement Noise @ 10 Hz vs. RMS Error

delay [ms]	p_w^i[m]	q_w^i[rad (rpy)]	λ[%]	p_i^s[m]	q_i^s[rad]
0	$\begin{bmatrix}0.005\\0.011\\0.002\end{bmatrix}$ $\begin{bmatrix}0.029\\0.023\\0.006\end{bmatrix}$	$\begin{bmatrix}0.012\\0.041\\0.008\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.015\end{bmatrix}$	0.3	$\begin{bmatrix}0.041\\0.010\\0.004\end{bmatrix}$	$\begin{bmatrix}0.034\\0.007\\0.000\end{bmatrix}$
50	$\begin{bmatrix}0.006\\0.012\\0.004\end{bmatrix}$ $\begin{bmatrix}0.035\\0.026\\0.008\end{bmatrix}$	$\begin{bmatrix}0.007\\0.020\\0.070\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.065\end{bmatrix}$	0.3	$\begin{bmatrix}0.028\\0.002\\0.064\end{bmatrix}$	$\begin{bmatrix}0.021\\0.071\\0.001\end{bmatrix}$
100	$\begin{bmatrix}0.011\\0.015\\0.005\end{bmatrix}$ $\begin{bmatrix}0.036\\0.036\\0.009\end{bmatrix}$	$\begin{bmatrix}0.012\\0.052\\0.025\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.024\end{bmatrix}$	0.3	$\begin{bmatrix}0.037\\0.005\\0.069\end{bmatrix}$	$\begin{bmatrix}0.015\\0.026\\0.000\end{bmatrix}$
200	$\begin{bmatrix}0.009\\0.016\\0.004\end{bmatrix}$ $\begin{bmatrix}0.036\\0.036\\0.006\end{bmatrix}$	$\begin{bmatrix}0.016\\0.077\\0.009\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.016\end{bmatrix}$	0.3	$\begin{bmatrix}0.031\\0.010\\0.076\end{bmatrix}$	$\begin{bmatrix}0.015\\0.006\\0.000\end{bmatrix}$
500	$\begin{bmatrix}0.017\\0.021\\0.005\end{bmatrix}$ $\begin{bmatrix}0.066\\0.080\\0.012\end{bmatrix}$	$\begin{bmatrix}0.010\\0.059\\0.040\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.045\end{bmatrix}$	0.4	$\begin{bmatrix}0.031\\0.002\\0.020\end{bmatrix}$	$\begin{bmatrix}0.040\\0.040\\0.001\end{bmatrix}$

TABLE II: Measurement Delay @ 10 Hz vs. RMS Error

update [Hz]	p_w^i[m]	q_w^i[rad (rpy)]	λ[%]	p_i^s[m]	q_i^s[rad]
20	$\begin{bmatrix}0.004\\0.012\\0.001\end{bmatrix}$ $\begin{bmatrix}0.026\\0.018\\0.005\end{bmatrix}$	$\begin{bmatrix}0.012\\0.065\\0.026\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.032\end{bmatrix}$	0.3	$\begin{bmatrix}0.041\\0.009\\0.003\end{bmatrix}$	$\begin{bmatrix}0.034\\0.025\\0.000\end{bmatrix}$
10	$\begin{bmatrix}0.005\\0.011\\0.001\end{bmatrix}$ $\begin{bmatrix}0.029\\0.023\\0.006\end{bmatrix}$	$\begin{bmatrix}0.012\\0.041\\0.008\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.014\end{bmatrix}$	0.3	$\begin{bmatrix}0.041\\0.010\\0.005\end{bmatrix}$	$\begin{bmatrix}0.034\\0.007\\0.000\end{bmatrix}$
5	$\begin{bmatrix}0.005\\0.012\\0.002\end{bmatrix}$ $\begin{bmatrix}0.021\\0.032\\0.005\end{bmatrix}$	$\begin{bmatrix}0.012\\0.043\\0.018\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.016\end{bmatrix}$	0.3	$\begin{bmatrix}0.039\\0.009\\0.009\end{bmatrix}$	$\begin{bmatrix}0.040\\0.017\\0.000\end{bmatrix}$
2	$\begin{bmatrix}0.009\\0.014\\0.003\end{bmatrix}$ $\begin{bmatrix}0.032\\0.030\\0.008\end{bmatrix}$	$\begin{bmatrix}0.013\\0.046\\0.022\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.021\end{bmatrix}$	0.3	$\begin{bmatrix}0.039\\0.008\\0.013\end{bmatrix}$	$\begin{bmatrix}0.041\\0.023\\0.000\end{bmatrix}$
1	$\begin{bmatrix}0.019\\0.016\\0.005\end{bmatrix}$ $\begin{bmatrix}0.045\\0.039\\0.009\end{bmatrix}$	$\begin{bmatrix}0.012\\0.019\\0.005\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.017\end{bmatrix}$	0.3	$\begin{bmatrix}0.040\\0.016\\0.024\end{bmatrix}$	$\begin{bmatrix}0.029\\0.003\\0.000\end{bmatrix}$

TABLE III: Measurement Update Rate vs. RMS Error

B. Dynamic Flight

σ_p[m] σ_q[rad]	p_w^i[m]	q_w^i[rad (rpy)]	λ[%]	p_i^s[m]	q_i^s[rad]
.001 .001 w/o att.	$\begin{bmatrix}0.003\\0.002\\0.001\end{bmatrix}$ $\begin{bmatrix}0.049\\0.055\\0.007\end{bmatrix}$	$\begin{bmatrix}0.024\\0.027\\0.040\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.033\end{bmatrix}$	0.4	$\begin{bmatrix}0.024\\0.016\\0.015\end{bmatrix}$	$\begin{bmatrix}0.000\\0.000\\0.000\end{bmatrix}$
.005 .009 w/ att.	$\begin{bmatrix}0.014\\0.011\\0.005\end{bmatrix}$ $\begin{bmatrix}0.059\\0.076\\0.015\end{bmatrix}$	$\begin{bmatrix}0.014\\0.026\\0.015\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.028\end{bmatrix}$	1.1	$\begin{bmatrix}0.041\\0.015\\0.017\end{bmatrix}$	$\begin{bmatrix}0.022\\0.019\\0.000\end{bmatrix}$
.005 .009 w/o att.	$\begin{bmatrix}0.014\\0.009\\0.005\end{bmatrix}$ $\begin{bmatrix}0.082\\0.086\\0.015\end{bmatrix}$	$\begin{bmatrix}0.022\\0.029\\0.029\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.034\end{bmatrix}$	0.5	$\begin{bmatrix}0.016\\0.021\\0.037\end{bmatrix}$	$\begin{bmatrix}0.000\\0.000\\0.000\end{bmatrix}$
.010 .018 w/ att.	$\begin{bmatrix}0.016\\0.010\\0.005\end{bmatrix}$ $\begin{bmatrix}0.070\\0.036\\0.011\end{bmatrix}$	$\begin{bmatrix}0.016\\0.024\\0.015\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.026\end{bmatrix}$	1.1	$\begin{bmatrix}0.037\\0.017\\0.025\end{bmatrix}$	$\begin{bmatrix}0.018\\0.006\\0.000\end{bmatrix}$
.010 .018 w/o att.	$\begin{bmatrix}0.016\\0.012\\0.007\end{bmatrix}$ $\begin{bmatrix}0.071\\0.056\\0.017\end{bmatrix}$	$\begin{bmatrix}0.021\\0.025\\0.049\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.039\end{bmatrix}$	0.6	$\begin{bmatrix}0.017\\0.028\\0.067\end{bmatrix}$	$\begin{bmatrix}0.000\\0.000\\0.000\end{bmatrix}$
.020 .036 w/ att.	$\begin{bmatrix}0.017\\0.017\\0.012\end{bmatrix}$ $\begin{bmatrix}0.061\\0.052\\0.020\end{bmatrix}$	$\begin{bmatrix}0.015\\0.028\\0.025\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.039\end{bmatrix}$	1.2	$\begin{bmatrix}0.031\\0.034\\0.025\end{bmatrix}$	$\begin{bmatrix}0.016\\0.012\\0.000\end{bmatrix}$
.020 .036 w/o att.	$\begin{bmatrix}0.018\\0.014\\0.011\end{bmatrix}$ $\begin{bmatrix}0.072\\0.075\\0.016\end{bmatrix}$	$\begin{bmatrix}0.021\\0.028\\0.038\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.037\end{bmatrix}$	0.7	$\begin{bmatrix}0.016\\0.040\\0.062\end{bmatrix}$	$\begin{bmatrix}0.000\\0.000\\0.000\end{bmatrix}$
.050 .036 w/ att.	$\begin{bmatrix}0.033\\0.033\\0.020\end{bmatrix}$ $\begin{bmatrix}0.105\\0.088\\0.026\end{bmatrix}$	$\begin{bmatrix}0.014\\0.027\\0.018\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.026\end{bmatrix}$	1.2	$\begin{bmatrix}0.018\\0.044\\0.026\end{bmatrix}$	$\begin{bmatrix}0.016\\0.014\\0.000\end{bmatrix}$
.050 .035 w/o att.	$\begin{bmatrix}0.033\\0.035\\0.020\end{bmatrix}$ $\begin{bmatrix}0.098\\0.108\\0.024\end{bmatrix}$	$\begin{bmatrix}0.023\\0.029\\0.060\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.050\end{bmatrix}$	0.7	$\begin{bmatrix}0.018\\0.024\\0.062\end{bmatrix}$	$\begin{bmatrix}0.000\\0.000\\0.000\end{bmatrix}$

TABLE IV: Measurement Noise @ 10 Hz vs. RMS Error

delay [ms]	p_w^i[m]	q_w^i[rad (rpy)]	λ[%]	p_i^s[m]	q_i^s[rad]
50	$\begin{bmatrix}0.009\\0.007\\0.004\end{bmatrix}$ $\begin{bmatrix}0.060\\0.080\\0.007\end{bmatrix}$	$\begin{bmatrix}0.021\\0.029\\0.050\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.045\end{bmatrix}$	0.500	$\begin{bmatrix}0.010\\0.043\\0.018\end{bmatrix}$	$\begin{bmatrix}0.025\\0.032\\0.000\end{bmatrix}$
100	$\begin{bmatrix}0.007\\0.009\\0.004\end{bmatrix}$ $\begin{bmatrix}0.064\\0.092\\0.006\end{bmatrix}$	$\begin{bmatrix}0.025\\0.044\\0.014\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.034\end{bmatrix}$	0.700	$\begin{bmatrix}0.006\\0.008\\0.052\end{bmatrix}$	$\begin{bmatrix}0.021\\0.024\\0.000\end{bmatrix}$
200	$\begin{bmatrix}0.025\\0.027\\0.006\end{bmatrix}$ $\begin{bmatrix}0.149\\0.068\\0.013\end{bmatrix}$	$\begin{bmatrix}0.030\\0.041\\0.036\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.038\end{bmatrix}$	1.000	$\begin{bmatrix}0.016\\0.014\\0.031\end{bmatrix}$	$\begin{bmatrix}0.010\\0.039\\0.000\end{bmatrix}$
500	$\begin{bmatrix}0.095\\0.170\\0.030\end{bmatrix}$ $\begin{bmatrix}0.264\\0.400\\0.036\end{bmatrix}$	$\begin{bmatrix}0.024\\0.063\\0.050\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.066\end{bmatrix}$	1.100	$\begin{bmatrix}0.018\\0.021\\0.030\end{bmatrix}$	$\begin{bmatrix}0.011\\0.046\\0.000\end{bmatrix}$

TABLE V: Measurement Delay @ 10 Hz vs. RMS Error

update [Hz]	p_w^i[m]	q_w^i[rad (rpy)]	λ[%]	p_i^s[m]	q_i^s[rad]
10	$\begin{bmatrix}0.009\\0.006\\0.005\end{bmatrix}$ $\begin{bmatrix}0.078\\0.064\\0.012\end{bmatrix}$	$\begin{bmatrix}0.020\\0.032\\0.020\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.025\end{bmatrix}$	0.8	$\begin{bmatrix}0.011\\0.009\\0.013\end{bmatrix}$	$\begin{bmatrix}0.019\\0.011\\0.000\end{bmatrix}$
5	$\begin{bmatrix}0.007\\0.006\\0.004\end{bmatrix}$ $\begin{bmatrix}0.050\\0.060\\0.008\end{bmatrix}$	$\begin{bmatrix}0.023\\0.040\\0.042\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.047\end{bmatrix}$	0.8	$\begin{bmatrix}0.007\\0.019\\0.024\end{bmatrix}$	$\begin{bmatrix}0.022\\0.024\\0.000\end{bmatrix}$
2	$\begin{bmatrix}0.016\\0.015\\0.005\end{bmatrix}$ $\begin{bmatrix}0.094\\0.068\\0.015\end{bmatrix}$	$\begin{bmatrix}0.012\\0.026\\0.031\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.031\end{bmatrix}$	0.8	$\begin{bmatrix}0.030\\0.014\\0.049\end{bmatrix}$	$\begin{bmatrix}0.013\\0.007\\0.000\end{bmatrix}$
1	$\begin{bmatrix}0.050\\0.032\\0.009\end{bmatrix}$ $\begin{bmatrix}0.177\\0.100\\0.020\end{bmatrix}$	$\begin{bmatrix}0.026\\0.040\\0.059\end{bmatrix}$ $\begin{bmatrix}-\\-\\0.063\end{bmatrix}$	0.8	$\begin{bmatrix}0.014\\0.029\\0.123\end{bmatrix}$	$\begin{bmatrix}0.016\\0.024\\0.000\end{bmatrix}$

TABLE VI: Update rate vs. RMS Error

REFERENCES

[1] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[2] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *International Journal of Robotics Research (IJRR)*, vol. 30, no. 1, pp. 56–79, 2011.

[3] F. Mirzaei and S. Roumeliotis, "A Kalman Filter-Based Algorithm for IMU-Camera Calibration: Observability Analysis and Performance Evaluation," *IEEE Transactions on Robotics and Automation*, vol. 24, no. 5, pp. 1143 –1156, 2008.

[4] D. Strelow and S. Singh, "Motion estimation from image and inertial measurements," *International Journal of Robotics Research (IJRR)*, vol. 23, no. 12, p. 1157, 2004.

[5] S. Roumeliotis, A. Johnson, and J. Montgomery, "Augmenting inertial navigation with image-based motion estimation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[6] A. Mourikis, N. Trawny, S. Roumeliotis, A. Johnson, A. Ansar, and L. Matthies, "Vision-aided inertial navigation for spacecraft entry, descent, and landing," *IEEE Transactions on Robotics (T-RO)*, vol. 25, no. 2, pp. 264–280, 2009.

[7] E. Jones, "Large scale visual navigation and community map building," Ph.D. dissertation, University of California at Los Angeles, 2009.

[8] P. Pinies, T. Lupton, S. Sukkarieh, and J. D. Tardós, "Inertial aiding of inverse depth SLAM using a monocular camera," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[9] M. Bryson, M. Johnson-Roberson, and S. Sukkarieh, "Airborne smoothing and mapping using vision and inertial sensors," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[10] T. Ford, J. Neumann, P. Fenton, M. Bobye, and J. Hamilton, "Oem4 inertial: A tightly integrated decentralised inertial/gps navigation system," NovAtel Inc, Tech. Rep., 2001.

[11] M. W. Achtelik, M. C. Achtelik, S. Weiss, and R. Siegwart, "Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[12] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," University of Minnesota, Dept. of Computer Science and Engineering, Tech. Rep. 2005-002, 2005.

[13] R. Hermann and A. Krener, "Nonlinear controllability and observability," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 728 – 740, 1977.