Published in Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA), 2013. DOI: http://dx.doi.org/10.1109/ICRA.2013.6630810

Incremental Smoothing vs. Filtering for Sensor Fusion on an Indoor UAV

Sven Lange, Niko Sünderhauf and Peter Protzel

***Abstract*— Our paper explores the performance of a recently proposed incremental smoother in the context of nonlinear sensor fusion for a real-world UAV. This efficient factor graph based smoothing approach has a number of advantages compared to conventional filtering techniques like the EKF or its variants. It can more easily incorporate asynchronous and delayed measurements from sensors operating at different rates and is supposed to be less error-prone in highly nonlinear settings.**

We compare the novel incremental smoothing approach based on iSAM2 against our conventional EKF based sensor fusion framework. Unlike previously presented work, the experiments are not only performed in simulation, but also on a real-world quadrotor UAV system using IMU, optical flow and altitude measurements.

I. Introduction

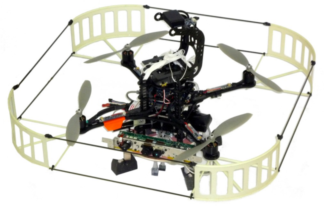
Accurate information of the UAV's system state is essential for a stable flight control. Especially in indoor scenarios with narrow places and no GPS signal, accurate sensor fusion algorithms are even more important for crash-free autonomous navigation. Bayesian filtering algorithms, like the Extended Kalman Filter (EKF) or Unscented Kalman Filter, are state of the art for fusing sensor information in this kind of scenarios. However, as the system becomes highly nonlinear, the Kalman Filter variants are known to be error-prone. This leads to the desire for new sensor fusion techniques.

Factor graph based optimization frameworks like $\mathrm{g}^2\mathrm{o}$ [7] are effectively used for solving large SLAM problems with high accuracy and adequate time consumption. In contrast to the filtering algorithms all previous measurements are used by the framework for state estimation. This leads to rapidly growing information to compute in each time step. So this technique was almost always used for offline estimation in the past.

To overcome the problem of continuously growing optimization problems, so called sliding window filters or also referred to as fixed lag smoothers were used by [12] or [1]. The reached accuracy and computation time of this technique depends closely on the length of the chosen window. Hence, an adequate window size enables this technique for use in on-line applications.

Recently, advances in the field of incremental smoothing led to an adaptive-lag smoother called iSAM2 [6]. Instead of using a fixed size for the optimization problem, it recalculates only a portion of the contained variables. This is realized by using the Bayes tree representation. Based upon this development, the authors of [4] used iSAM2 for sensor fusion in the context of an inertial navigation system. They simulated an aerial vehicle including IMU, GPS and stereo vision measurements, used the iSAM2 algorithm for smoothing and compared it to a common EKF approach. However, only simulation results were presented.

The authors are with the Department of Electrical Engineering and Information Technology, Chemnitz University of Technology, 09111 Chemnitz, Germany. Contact: sven.lange@etit.tu-chemnitz.de Website: http://www.tu-chemnitz.de/etit/proaut


(a)

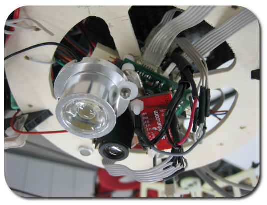

(b)

Fig. 1. (a) shows the multirotor system we use. (b) shows the additional microcontroller board with the optical flow sensor.

While our work is similar to [4], we demonstrate an iSAM2 based sensor fusion algorithm on a real world system and compared it to our conventional EKF based solution. Our focus lies on autonomous indoor navigation with a multirotor UAV, so we did not use GPS measurements. Sensors we use for sensor fusion are: IMU, optical flow and sonar for altitude measurements.

II. System Overview

In the following, we will introduce the hardware configuration which was used for the experiments as well as the filter and smoother setup.

A. Multirotor UAV

The UAV we use for our work is called *Pelican* (see Fig. 1(a)) and is manufactured by Ascending Technologies GmbH, Munich, Germany. This mid-size four-rotor UAV, or quadrocopter, measures $72\,\mathrm{cm}$ in diameter and can carry up to $500\,\mathrm{g}$ of payload for about $20\,\mathrm{min}$.

The quadcopter is equipped with a variety of sensors: Besides the usual accelerometers, gyros and a magnetic field sensor, a pressure sensor and a GPS module provide input for AscTec's sophisticated sensor fusion algorithm and the control loop running at $1\,\mathrm{kHz}$. Like on the smaller Hummingbird system, an *AscTec AutoPilot board* is responsible for data fusion and basic control of the UAV. An additional microcontroller on this board is available for custom programming. We updated it with the firmware coming with the *asctec_mav_framework*[1] for high frequency IMU sensor readings which otherwise are not available. More technical details on the AutoPilot Board, the Hummingbird and the controllers can be found in [3].

B. Additional Sensors

We extended the system's configuration with additional hardware. In relation to this work, a custom made microcontroller board based on an ATmega644P which includes an ADNS-3080 optical flow sensor, is of special interest (Fig. 1(b)). Connected to this board is an SRF10 sonar sensor with a beam width of about 60°, a separate IMU and an XBee Pro radio module for communication. The Avago ADNS-3080 optical flow sensor is combined with a small M12 mount camera lens for detecting the UAV's current velocity over ground. The SPI bus is used for interfacing the sensor to our microcontroller which makes the measurements available to our ground station. We already used this configuration for example in [8], [10].

III. System Equations

We describe the current state of the system with the following vector:

$$\mathbf{x} = \begin{bmatrix}\mathbf{v}^B & \mathbf{a}^B & \mathbf{q}^{BW} & \boldsymbol{\omega}^B & h^W & \mathbf{b}_a^{IMU} & \mathbf{b}_\omega^{IMU}\end{bmatrix}^\mathsf{T} \quad (1)$$

Where $\mathbf{v}^B$ and $\mathbf{a}^B$ is the velocity and acceleration of the system within the body frame B relative to the world frame W. Both frames' axes are given in North-East-Down (*NED*) convention. The vector $\mathbf{q}^{BW}$ is a quaternion which describes the rotation between body frame B and world frame W. If $\mathbf{R}^{BW} = R(\mathbf{q}^{BW})$ is the related rotation matrix, a point within the body frame is given through: $\mathbf{p}^B = \mathbf{R}^{BW} \cdot \mathbf{p}^W + \mathbf{t}_W^B$ where $\mathbf{t}_W^B$ is the world frame's translation within the body frame. The following state variable $\boldsymbol{\omega}^B$ represents the turn rate and h^W is the height above ground.

As accelerometers and gyros are known to have biases which can be modeled as random walk processes, we have two additional vectors $\mathbf{b}_a^{IMU}$ and $\mathbf{b}_\omega^{IMU}$ within the state representation for the accelerometer's bias and gyro's bias. Both variables are given within the IMU frame.

Different from commonly used state representations, we chose the velocity and acceleration to be within the body frame, as no global position information is available. Based on this representation, the state vector can directly be used for navigation controllers.

A. State Transition

As state transition, we use a constant turn rate and acceleration *(CTRA)* model, where $\mathbf{x}_{t+1}$ is the predicted state after the time period Δt based on the last known state vector $\mathbf{x}_t$.

$$\mathbf{x}_{t+1} = f^{CTRA}(\mathbf{x}_t, \Delta t) \quad (2)$$

$$\mathbf{x}_{t+1} = \begin{bmatrix} \Delta\mathbf{q} \cdot \mathbf{v}_t^B + (\mathbf{a}_{t+1}^B + \boldsymbol{\omega}_{t+1}^B \times \mathbf{v}_t^B) \cdot \Delta t \\ \Delta\mathbf{q} \cdot \mathbf{a}_t^B + \boldsymbol{\eta}_A \\ \boldsymbol{\Phi}(\boldsymbol{\omega}_{t+1}^B, \Delta t) \cdot \mathbf{q}_t^{BW} \\ \boldsymbol{\omega}_t^B + \boldsymbol{\eta}_{TR} \\ h_t^W + v_z^W \cdot \Delta t \\ \mathbf{b_a} + \boldsymbol{\eta}_{\mathbf{b_a}} \\ \mathbf{b}_{\boldsymbol{\omega}} + \boldsymbol{\eta}_{\mathbf{b}_\omega} \end{bmatrix} \quad (3)$$

Where $\Delta\mathbf{q} = \mathbf{q}_{t+1}^{BW} \cdot (\mathbf{q}_t^{BW})^{-1}$ is the rotation between the consecutive rotating frames. We propagate the new rotation as described in [13, p. 185]. This specialized form of quaternion prediction enforces the unit norm constraint.

$$s = \frac{1}{2} \cdot \|\boldsymbol{\omega}_{t+1}^B\| \cdot \Delta t \quad (4)$$

$$\lambda = 1 - \|\mathbf{q}_t^{BW}\|^2 \quad (5)$$

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \cdot \Delta t \quad (6)$$

$$\boldsymbol{\Phi} = \mathbf{I}_{4\times4} \cdot (\cos(s) + \lambda) - \frac{1}{2} \cdot \boldsymbol{\Omega} \cdot \frac{\sin(s)}{s} \quad (7)$$

Process noise for the state transition is added by the noise vector $\boldsymbol{\eta} = \begin{bmatrix}\boldsymbol{\eta}_A & \boldsymbol{\eta}_{TR} & \boldsymbol{\eta}_{\mathbf{b_a}} & \boldsymbol{\eta}_{\mathbf{b}_\omega}\end{bmatrix}^\mathsf{T}$, which is modeled as white Gaussian noise with zero mean and standard deviations $\boldsymbol{\sigma} = \begin{bmatrix}\boldsymbol{\sigma}_A & \boldsymbol{\sigma}_{TR} & \boldsymbol{\sigma}_{\mathbf{b_a}} & \boldsymbol{\sigma}_{\mathbf{b}_\omega}\end{bmatrix}^\mathsf{T}$. The parameters $\boldsymbol{\sigma}_A$ and $\boldsymbol{\sigma}_{TR}$ describe the certainty of the assumed CTRA model and $\boldsymbol{\sigma}_{\mathbf{b_a}}$ and $\boldsymbol{\sigma}_{\mathbf{b}_\omega}$ are used for modelling the IMU's random walk for acceleration and turn rate.

B. Measurement Prediction

In the following, we describe our measurement prediction equations used in the smoothing and filtering solution.

1) Accelerometer and Gyro: For predicting the acceleration $\hat{\mathbf{z}}_a^{IMU}$ and turn rate $\hat{\mathbf{z}}_\omega^{IMU}$ measured by the accelerometer and gyroscope, we use a common equation as can be found for example in [2]. The IMU sensor's frame coincides with the body frame, so no additional rotation matrix $\mathbf{R}^{IMU,B}$ or translation vector $\mathbf{t}_{IMU}^B$ is needed. Additionally, as the sensor is located near the center of gravity, there is no need for compensation of lever arm effects.

$$\hat{\mathbf{z}}_a^{IMU} = \mathbf{a}^B + \mathbf{b}_a^{IMU} + \boldsymbol{\omega}^B \times \mathbf{v}^B - \mathbf{q}^{BW} \cdot \mathbf{g}^W \quad (8)$$

$$\hat{\mathbf{z}}_\omega^{IMU} = \boldsymbol{\omega}^B + \mathbf{b}_\omega^{IMU} \quad (9)$$

All variables are known from the state vector except the constant $\mathbf{g}^W$, which is the vector of gravity, given in the world frame.

2) Optical Flow: The optical flow sensor measures the UAV's displacement relative to the ground in values of accumulated pixels per time period. This process is analog to camera based optical flow algorithms using image processing. Essential differences are the small sensor size, fast processing through the algorithm's hardware implementation and the small and light-weight design.

[1]http://www.ros.org/wiki/asctec_mav_framework

Regarding the sensor information's use, we have to notice the sensor's sensibility for rotational changes $\mathbf{z}^O_{\texttt{rot}}$, which adds to the translational part $\mathbf{z}^O_{\texttt{tran}}$. Also of interest is the height above ground h^O within the sensor's frame O to get the correlation between metric velocity information and the measurement in pixels. The sensor frame O is attached to the body frame through the rotation $\mathbf{R}^{OB}$ and translation $\mathbf{t}^B_O$.

$$\hat{\mathbf{z}}^O_{\texttt{oflow}} = \frac{1}{h^O} \cdot \mathbf{K} \cdot (\mathbf{z}^O_{\texttt{rot}} - \mathbf{z}^O_{\texttt{tran}}) \tag{10}$$

$$\mathbf{z}^O_{\texttt{rot}} = (\mathbf{R}^{OB} \cdot \boldsymbol{\omega}^B) \times \begin{bmatrix}0 & 0 & h^O\end{bmatrix}^\intercal \tag{11}$$

$$\mathbf{z}^O_{\texttt{tran}} = \mathbf{R}^{OB} \cdot \mathbf{v}^B \tag{12}$$

$$h^O = \mathbf{R}^{OB}_{(3,1:3)} \cdot (\mathbf{q}^{BW} \cdot \begin{bmatrix}0 & 0 & h^W\end{bmatrix}^\intercal - \mathbf{t}^B_O) \tag{13}$$

$$\mathbf{K} = \mathbf{I}_{2\times2} \cdot f \cdot \frac{\texttt{resolution}\cdot\texttt{mode}}{\texttt{size}} \tag{14}$$

The constant $\mathbf{K}$ can be seen as a kind of intrinsic camera matrix for the optical flow sensor, with the focal length f and a relation to convert the velocity in metric units to pixel units. In our case, we use the ADNS-3080, which has a resolution of 30 px and a sensor size of 60 µm. We use it in the 1600 cpi mode, so $\texttt{mode} = 2$ and $f = 8\,\text{mm}$, which corresponds to the lens we have mounted in front of the sensor.

Worth mentioning is the sensor's information about the measurement quality, which corresponds to the tracked pixels between the successive frames. By experiments, we found that sensor readings are only reliable, if this quality value is larger than 50. Taking this into account, we filter sensor readings out accordingly before using them within the filter or the smoother.

3) Distance: For measurement prediction of the sonar sensor, we use a simplified model. Because of the beam width, we assume the measurement to be the UAV's height within the world frame.

$$\hat{z}^U_{\texttt{sonar}} = \mathbf{R}^{UB}_{(3,1:3)} \cdot \left(\begin{bmatrix}0 & 0 & h^W\end{bmatrix}^\intercal - (\mathbf{q}^{BW})^{-1} \cdot \mathbf{t}^B_U\right) \tag{15}$$

Where the two constants $\mathbf{R}^{UB}$ and $\mathbf{t}^B_U$ describe the rotation and translation of the ultrasonic range finder relative to the body frame and $\mathbf{R}^{UB}_{(3,1:3)}$ is the third row of the rotation matrix $\mathbf{R}^{UB}$.

4) Compass: Compass measurements are necessary to measure the absolute rotation about the UAV's yaw axis. Currently we use a very simple model which will be replaced in later work.

$$\mathbf{R}^{BW} = R(\mathbf{q}^{BW}) \tag{16}$$

$$\hat{z}_{\texttt{compass}} = \operatorname{atan2}(\mathbf{R}^{BW}_{1,2}, \mathbf{R}^{BW}_{1,1}) \tag{17}$$

The compass measurement corresponds to the UAV's internal prediction of its orientation about the yaw axis, so we extract this information from the orientation representation $\mathbf{q}^{BW}$ of our state vector.

IV. Implementation

For comparing both sensor fusion techniques, we implemented an EKF based and a smoothing based solution. In contrast to common implementations like in [13], [11], we decided to use the CTRA model instead of using the IMU measurements as input for the state transition. This enables us to deal with sensor drop outs or asynchronous measurements which do not match the IMU measurement times, i.e. if the IMU is failing for short time, we still can add new state predictions based on the available sensor information. We think this is a more general approach regarding the smoothing algorithm. Additionally, we plan to use a more sophisticated transition model in future work. Consequently we use the CTRA model also in the EKF algorithm for reasons of comparability. The following sections describe specific implementation details.

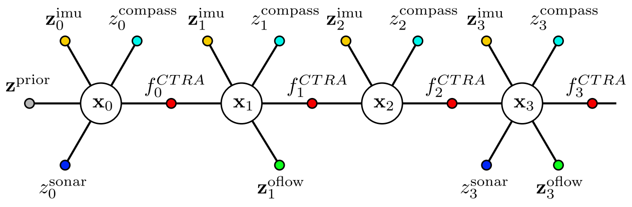


Fig. 2. Factor graph representation as used in our implementation. IMU and compass measurements are always available and sonar and optical flow measurements are available at different frequencies. Note that f_k^{CTRA} is the short form of $f^{CTRA}(\mathbf{x}_k, \Delta t)$

A. EKF

Our filter implementation is build upon the ReBEL Toolkit [13], which is a Matlab implementation of several estimation methods including the EKF. Our state prediction step of the EKF uses the described CTRA transition function with a rate of 100 Hz. Accordingly the following update step is executed with the same frequency. Because of different sensor rates, measurement predictions are made depending on the currently available measurements only. The Jacobi matrices for transferring the covariances of the process noise and sensor noise into the state space are derived symbolically by Matlab's Symbolic Math Toolbox.

B. Incremental Smoothing

We use the novel algorithm *iSAM2* [5] for our smoothing solution. This graph based optimization algorithm is particularly developed for efficient incremental predictions which are needed for on-line use in sensor fusion applications. While frameworks like g^{2}o [7] have to solve the complete optimization problem for concurrent state prediction, the iSAM2 algorithm uses the Bayes tree representation and applies only partial variable recalculations.

For our application we implemented a factor graph for iSAM2, as shown exemplarily in Fig. 2 for the first three states. This gives us the following cost functions:

$$\mathbf{e}^{\text{ctra}}_t = f^{CTRA}(\mathbf{x}_{t-1}, \Delta t) - \mathbf{x}_t \tag{18}$$

$$\mathbf{e}^{\text{imu}}_t = \begin{bmatrix}\hat{\mathbf{z}}^{IMU}_a(\mathbf{x}_t) \\ \hat{\mathbf{z}}^{IMU}_\omega(\mathbf{x}_t)\end{bmatrix} - \mathbf{z}^{\text{imu}}_t \tag{19}$$

$$e^{\text{compass}}_t = \hat{z}_{\texttt{compass}}(\mathbf{x}_t) - z^{\text{compass}}_t \tag{20}$$

$$\mathbf{e}^{\text{oflow}}_t = \hat{\mathbf{z}}^O_{\texttt{oflow}}(\mathbf{x}_t) - \mathbf{z}^{\text{oflow}}_t \tag{21}$$

$$e^{\text{sonar}}_t = \hat{z}^U_{\texttt{sonar}}(\mathbf{x}_t) - z^{\text{sonar}}_t \tag{22}$$

where $\mathbf{z}_t^{\{\dots\}}$ are the sensor's measurement vectors at time step t.

The covariances $\mathbf{\Sigma}^{\{\dots\}}$ connected to each measurement cost function are composed of the sensor's noise parameters. The optical flow factor's covariance for instance is $\mathbf{\Sigma}^{\text{oflow}} = \text{diag}(\boldsymbol{\sigma}^2_{\text{oflow}})$. Of special interest is the CTRA factor's covariance which is modeled as process noise for the acceleration, turn rate and the bias terms: $\mathbf{\Sigma}'^{\text{ctra}} = \text{diag}(\left[\boldsymbol{\sigma}^2_A, \boldsymbol{\sigma}^2_{TR}, \boldsymbol{\sigma}^2_{\mathbf{b_a}}, \boldsymbol{\sigma}^2_{\mathbf{b_\omega}}\right])$. Therefore it has to be converted into the state space first by using a transition function like the unscented transformation or as we do it, with the Jacobian $\mathbf{J} = \frac{\partial f^{CTRA}}{\partial \boldsymbol{\eta}}$ and $\mathbf{\Sigma}^{\text{ctra}} = \mathbf{J}\mathbf{\Sigma}'^{\text{ctra}}\mathbf{J}^\top$.

The iSAM2 algorithm can be configured by several parameters, where we chose the relinearization threshold to be 0.01 and left the wildfire threshold unchanged at 1×10^{-5}. As optimization method we chose *dogleg*. Currently all derivatives are implemented as numerical derivatives, which will change in the future.

As UAV measurements arrive, initially a new system state vertex is created by the CTRA transition edge, where the available sensors' information is connected by unary constraints (see Fig. 2). The following incremental update operation of iSAM2 includes relinearization and adjusts the state predictions accordingly. Even if the marginals are not mandatory for controlling the UAV, we extract them after every update step for the concurrent state prediction. Concurrent means, that we always use the latest prediction available, which is the usual way sensor fusion works in online applications.

V. Results

We compared the smoothing against the filtering technique for two scenarios. First, a simulation was used to validate the implementation and test the performance with known noise parameters. Afterwards a real world experiment was performed with the previously described UAV system.

A. Simulation

The simulation is based on a Matlab Simulink model, which implements a point mass model simulating the quadrocopter and all sensors which are used on the real system. An overlying position controller controls the simulated model with the following inputs: At time $t = 1\,\text{s}$ the position setpoint changes to $(x, y, z) = (2, 0, 1)$ and at $t = 3\,\text{s}$ it is modified to $(x, y, z) = (2, 1, 1)$. The simulated sensor measurements are gathered at a frequency of $100\,\text{Hz}$ and are corrupted with a zero-mean Gaussian noise for all sensors.

For the accelerometer and gyroscope measurements we used a standard deviation of $\boldsymbol{\sigma}_a = 30\,\text{mg}$ and $\boldsymbol{\sigma}_\omega = 2°\,\text{s}^{-1}$ and for the corresponding random walk model within the state transition, we used $\boldsymbol{\sigma}_{\mathbf{b}_a} = 0.1\,\text{mg}/\sqrt{\text{Hz}}$ and $\boldsymbol{\sigma}_{\mathbf{b}_\omega} = 1\,°/\sqrt{\text{h}}$. The sonar sensor's noise is modeled with $\sigma_{\text{sonar}} = 1\,\text{cm}$ and the optical flow sensor's noise is $\boldsymbol{\sigma}_{\text{oflow}} = 10\,\text{px/s}$. For the compass we use $\sigma_{\text{compass}} = 2°$.

We compared the smoothing and filtering results to our ground truth data for the complete state vector with special interest in the resulting error for rotation, velocity and height

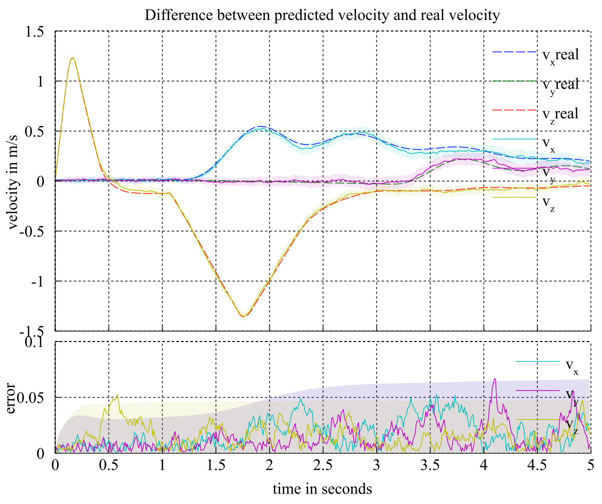


Fig. 3. Simulation results of iSAM2 for the concurrent velocity in all three dimensions. Real values from simulation are shown as dotted lines and the 3σ bound is shown as transparent patch for each dimension in the corresponding color.

and the violation of the corresponding 3σ bounds. Fig. 3 shows the resulting velocity of the smoother's concurrent predictions. As can be seen, the prediction follows the real state as expected. The resulting error compared to the ground truth is generally constrained within the 3σ bounds. The performance is similarly for the orientation, as can be seen in Fig. 4 as well as for the height, turn rate, acceleration and biases.

The comparison between the smoothing results and the EKF's filtering results is shown in Fig. 5 for the velocity values. It shows nearly no differences between both methods. The mean error of the filtering result, compared to ground truth is shown in Table I, which is identical to the smoothing result. A higher precision of the smoother can only be reached, if we use offline smoothing (batch optimization) as shown by the *final* error curve.

For closer examination of the smoother's mean velocity error, we did several runs with different acceptable delay times. This means, we did not use the concurrent smoothing result, but the result from a constant time before. The results are plotted within Fig. 6. Depending on the application, the delayed prediction should be preferred to get a higher accuracy.

B. Real World

Our second experiment evaluates real world data, gathered by our UAV system, for validating the smoother's and filter's performance. We used our low-cost tracking system [9] that reaches almost millimeter accuracy. It has a standard deviation for the position measurement of about $2.5\,\text{mm}$ in a common distance of about $1.5\,\text{m}$ to the camera, but we need it to measure the velocity, which results in stronger noise values.

For data acquisition, we moved the system two times along a squared trajectory of about $0.7\,\text{m}$ by $0.7\,\text{m}$ in different heights. The same sensors were used as in the

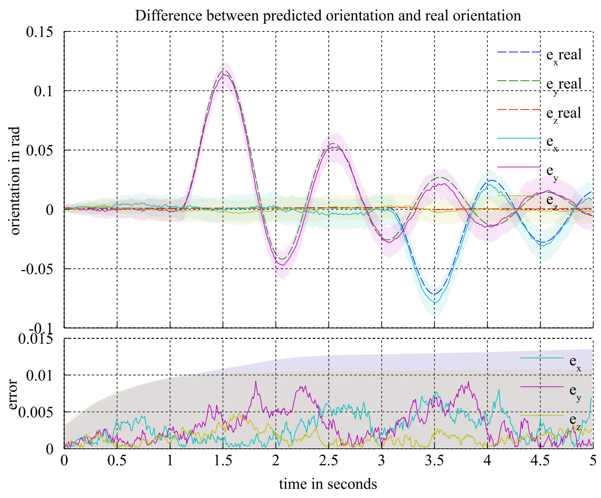


Fig. 4. Simulation results of iSAM2 for the concurrent orientation as Euler angle representation. Real values from simulation are shown as dotted lines and the 3σ bound is shown as transparent patch for each dimension in the corresponding color.

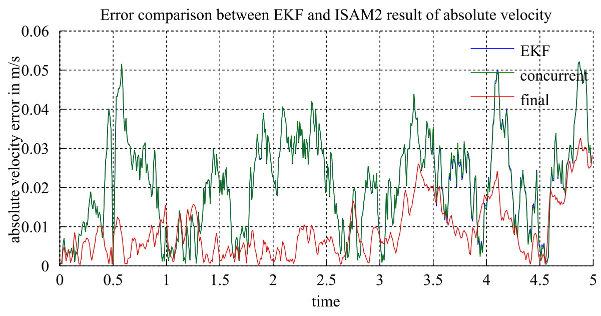


Fig. 5. Error comparison between EKF, concurrent and final solution for the simulation experiment. The error is computed between absolute velocity of ground truth and absolute velocity of state prediction. Notice that the curves for the EKF and concurrent smoothing solution (blue and green) are indistinguishable.

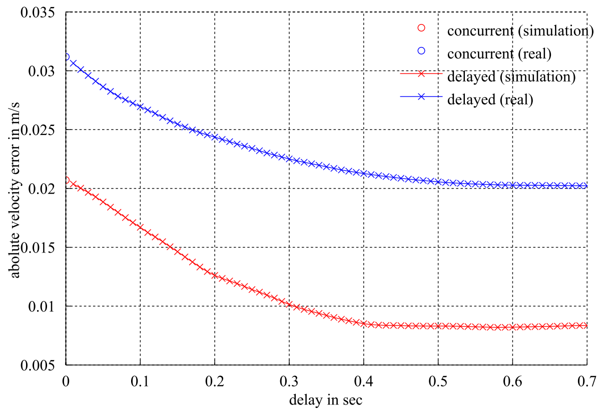


Fig. 6. The diagram shows the smoother's performance if used with different delay times. Plotted are the errors for the complete simulation and real world sequence over the used delay time. Here the circles at delay time 0 mark the concurrent results and each cross marks the mean error for a complete run with the specific delay time. This is also referred to as fixed-lag smoothing where the lag corresponds to the delay time.

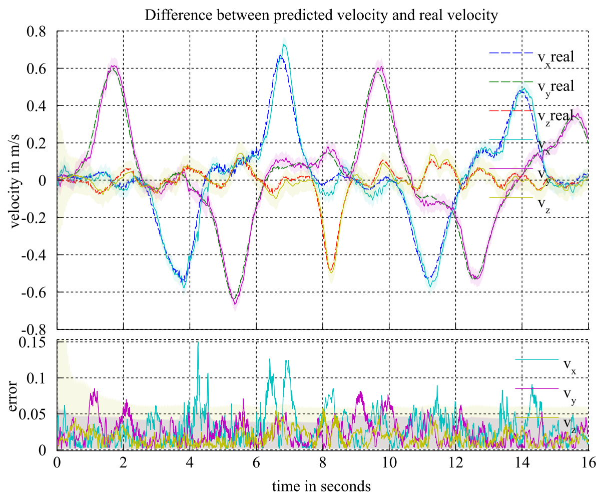


Fig. 7. Real world results of iSAM2 for the concurrent velocity in all three dimensions. Real values from the tracking system are shown as dotted lines and the 3σ bound is shown as transparent patch for each dimension in the corresponding color.

simulation environment, but this time with different measurement frequencies for the sonar and the optical flow sensor. New sonar measurements are available at a rate of about 20 Hz and optical flow measurements arrive with 50 Hz, but with 5 % filtered outliers. Outliers are filtered by the sensor's quality information as described previously. The used standard deviations are the same as in the simulation environment.

The result for the velocity including the error between the ground truth, is plotted in Fig. 7 for the smoother and in Fig. 8 for the EKF. Similarly to the simulation results, there are only minor differences between both variants. For better visualization, the error of the absolute velocity is plotted together with the final result in Fig. 9. As can be seen, there are only minor differences between the concurrent prediction of the smoother and the prediction of the EKF. Only the final result performs clearly better, which can also be seen in Table I.

Another visualization of this result is given in Fig. 10. The boxplot illustrates the strong similarity again. In contrast to the final result, no meaningful differences can be discovered between concurrent result and the EKF result.

TABLE I

MEAN ERRORS OF ABSOLUTE VELOCITY

	EKF	**Concurrent**	**Final**
Simulation	$0.0206\,\mathrm{m\,s^{-1}}$	$0.0206\,\mathrm{m\,s^{-1}}$	$0.0087\,\mathrm{m\,s^{-1}}$
Real world	$0.0309\,\mathrm{m\,s^{-1}}$	$0.0306\,\mathrm{m\,s^{-1}}$	$0.0205\,\mathrm{m\,s^{-1}}$

VI. CONCLUSIONS

We described a complete sensor fusion algorithm for our UAV system using two different approaches, the commonly used EKF and a new incremental smoothing approach based on iSAM2. Our real-world results confirm the simulated

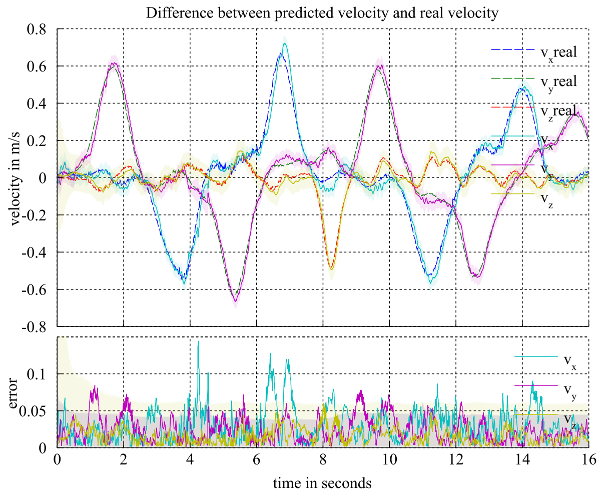


Fig. 8. Real world results of EKF for the concurrent velocity in all three dimensions. Real values from the tracking system are shown as dotted lines and the 3σ bound is shown as transparent patch for each dimension in the corresponding color.

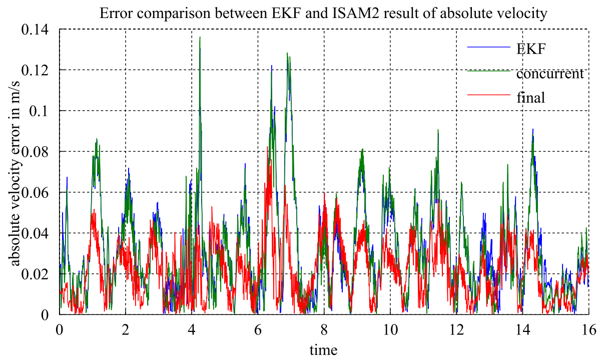


Fig. 9. Error comparison between EKF, concurrent and final solution for the real world experiment. The error is computed between absolute velocity of ground truth and absolute velocity of state prediction. Notice that the curves for the EKF and concurrent smoothing solution (blue and green) are nearly indistinguishable.

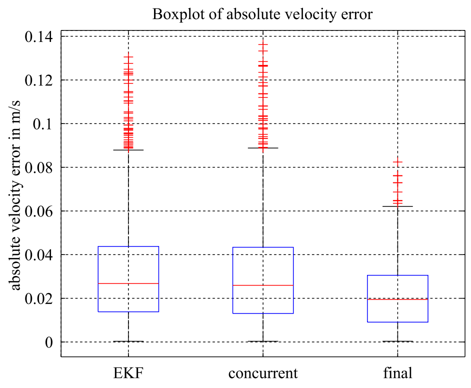


Fig. 10. Boxplot comparing the absolute velocity errors for the real world experiment.

findings of [4] that the accuracy of the concurrent smoothing and the EKF state estimation are almost equal, even though we did not have global position information. Tuning iSAM2's parameters for the relinearization threshold or the wildfire parameter, or changing the underlying optimization method between Powell's dog-leg and Gauss-Newton does not change this fact significantly. Only when using a delayed smoothing result instead of the concurrent predictions, there is a growing gain of accuracy compared to the EKF's results.

To our knowledge, our paper is the first that describes the application of the incremental smoothing approach for sensor fusion on a real world system. Although this did not reveal any gain in accuracy, the smoothing approach bears further potential advantages like easy incorporation of asynchronous and delayed measurements or sensor drop-outs. In future work we will explore the influence and relevance of these advantages to real-world dynamic systems. The detection of outliers which occur in the optical flow measurements in case of repeating ground patterns is another challenge for both the filtering and smoothing algorithms. Furthermore, we think the smoother can more conveniently be extended to incorporate a tightly coupled mapping, localization, or SLAM component.

References

[1] Tue-Cuong Dong-Si and A.I. Mourikis. Motion Tracking with Fixed-lag Smoothing: Algorithm and Consistency Analysis. In *Proc. of Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[2] Jay Farrell. *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill, Inc., New York, NY, USA, 2008.

[3] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus. Energy-efficient Autonomous Four-rotor Flying Robot Controlled at 1 kHz. In *Proc. of Intl. Conf. on Robotics and Automation (ICRA)*, 2007.

[4] V. Indelman, S. Williams, M. Kaess, and F. Dellaert. Factor Graph Based Incremental Smoothing in Inertial Navigation Systems. In *Proc. of Intl. Conf. on Information Fusion (FUSION)*, 2012.

[5] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental Smoothing and Mapping with Fluid Relinearization and Incremental Variable Reordering. In *Proc. of Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[6] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree. *Intl. Journal of Robotics Research*, 2012.

[7] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A General Framework for Graph Optimization. In *Proc. of Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[8] S. Lange, N. Sünderhauf, P. Neubert, S. Drews, and P. Protzel. Autonomous Corridor Flight of a UAV Using a Low-Cost and Light-Weight RGB-D Camera. In *Proc. of Intl. Symposium on Autonomous Mini Robots for Research and Edutainment (AMiRE)*, 2011.

[9] Sven Lange and Peter Protzel. Cost-Efficient Mono-Camera Tracking System for a Multirotor UAV Aimed for Hardware-in-the-Loop Experiments. In *Proc. of Intl. Multi-Conference on Systems, Signals and Devices (SSD)*, 2012.

[10] Sven Lange, Niko Sünderhauf, and Peter Protzel. A Vision Based Onboard Approach for Landing and Position Control of an Autonomous Multirotor UAV in GPS-Denied Environments. In *Proc. of Intl. Conf. on Advanced Robotics (ICAR)*, 2009.

[11] Jacob Willem Langelaan. *State Estimation for Autonomous Flight in Cluttered Environments*. PhD thesis, Department of Aeronautics and Astronautics, Stanford University, 2006.

[12] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding Window Filter with Application to Planetary Landing. *J. Field Robotics*, 27(5):587–608, 2010.

[13] Rudolph van der Merwe. *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*. PhD thesis, Oregon Health & Science University, 2004.