Orientation Sensing for Helicopter UAVs under Strict Resource Constraints

Marek Musial Carsten Deeg Volker Remuß Günter Hommel

Technische Universität Berlin
Real-Time Systems & Robotics Group
http://pdv.cs.tu-berlin.de/leute/$\{$*musial,deeg,remuss,hommel*$\}$*.html*

Abstract

For flight control of helicopter UAVs, it is essential to have high-quality 6-DOF attitude measurements. In the course of the miniaturization, the need for reliable orientation sensors conflicts with strict resource constraints. This paper presents the algorithms for orientation calculation used on-board TU Berlin's helicopter UAV MARVIN as an example of "low-resource" orientation measurement. While rotation rate sensors provide short-term data, acceleration and magnetic field sensors are fused for delayed on-line calibration, using GPS to eliminate kinematic acceleration effects. All maths are performed using 32 bit fixed point arithmetics, involving certain "crude" simplifications. The algorithm's performance and computational complexity are compared to a full Kalman Filter fusion of the sensors, justifying the use of the specialized and simplified approach.

1 Introduction

For flight control of autonomous helicopter UAVs, it is essential to have high-quality 6-DOF attitude measurements, including both position and orientation. While it is certainly possible to build a fully autonomous aerodynamically stable fixed-wing UAV without any orientation sensors, helicopters are aerodynamically unstable. Therefore, continuous measuring of the pitch and roll angles and their active control is required even to keep the UAV airborne for more than a few seconds.

In the course of the miniaturization of helicopter UAVs, the need for reliable orientation sensors conflicts with strict resource constraints regarding weight, power consumption, computational demands, and cost. Of-the-shelf IMUs are usually designed for much bigger vehicles and therefore not suited for the use with micro UAVs. Instead, it is highly desirable to utilize light and cheap sensors and have a simple on-board microcontroller evaluate their signals at low computational complexity.

This paper presents the algorithms used by TU Berlin's helicopter UAV MARVIN [3, 6, 7, 8, 9] to fuse a set of sensors for "low-resource" orientation measurement. MARVIN won the 2000 *International Aerial Robotics Competition* [2] in the US ahead of 9 teams from Canada and the United States. The paper motivates why 3 sensor groups are actually necessary under the conditions at hand, explains the data processing in detail and analyses the resulting orientation sensor's performance with regard to the effects of the low-resource optimizations.

The rest of this paper is organized as follows: Section 2 explains what types of sensors are minimally required and present on-board MARVIN, section 3 explains the fusion algorithm for orientation measurement, and section 4 evaluates the algorithm used and compares its performance to standard "state-of-the-art" methods.

2 Sensors

This section explains what types of sensors are minimally required for the task at hand and which sensors are actually used on-board MARVIN.

Available Sensor types relevant to orientation measurement are:

- acceleration sensors,
- rotation rate sensors,
- magnetic field sensors.

Position sensors such as GPS might assist in the sensor fusion. While it is basically sufficient to use rotation rate integration for orientation measurement, small and cheap rotation rate sensors do inevitably lead to unacceptable drift errors due to the integration and require on-line calibration.

Acceleration sensors are basically capable of guiding this on-line calibration process. Unfortunately, they always measure a combination of kinematic and gravitational acceleration. Acceleration measured on-board a helicopter tends to be "tied" to the main rotor axis unless air resistance resulting from horizontal motion comes into play. Therefore, the beginning of an unwanted pitch or roll rotation cannot be registered at all from the readings of a 3D acceleration sensor!

Finally, the acceleration and magnetic field vectors can even ideally resolve only 2 DOF each. Thus, a full set of all of the sensor types mentioned above is strictly required for orientation sensing under the requirements considered here.

With respect to above results, MARVIN carries the following sensors for position and orientation measurement:

- 3 piezo-electric rotation sensors, Murata ENC05E(A)
- 3 semiconductor acceleration sensors, Analog Devices ADXL05
- 3 magnetic field sensors, Speake & Co. Ltd. FGM-1/FGM-2
- Carrier-phase differential GPS receiver, NovAtel RT-2 Millennium GPSCard

The RT-2 DGPS receiver delivers real-time kinematics measurements with roughly 2 cm accuracy at 5 Hz. All sensors are directly connected to the on-board microcontroller, an Infineon SAB80C167. The acceleration and rotation sensors use analog signals converted by on-chip 10 bit converters, the compass outputs PWM signals, and the GPS receiver uses a serial data interface to output position and velocity logs.

3 Fusion Algorithm

This section presents the sensor fusion algorithm used on-board MARVIN to calculate the orientation information from the reading of the sensor groups mentioned in the previous section.

All maths for orientation sensing is performed using 32 bit fixed point arithmetics, because the on-board controller does not have an FPU and floating point calculation in software results in high computational and memory cost (the latter mainly for the library routines). On the other hand, integrating the sensor interfaces and all software up to the flight controller on a single chip provides a very easy and robust system design under the resource constraints that have to be met.

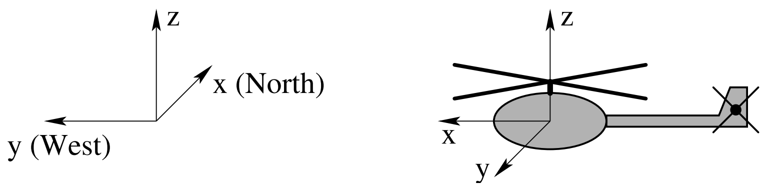


Figure 1: Base and helicopter coordinate systems

3.1 State Description

For calculations involving coordinates, three different coordinate frames are used: The *base coordinate system* (BCS) and the *helicopter coordinate system* (HCS) are defined according to figure 1. Additionally, the *magnetic base coordinate system* (MCS) resembles BCS but with x to Magnetic North instead of True North. In this section, the superscripts ${}^{\text{BCS}}a$, ${}^{\text{MCS}}b$, and ${}^{\text{HCS}}c$ will be used to denote the system to which any $a, b, c \in \mathbb{R}^3$ refer.

The state description for orientation calculation consists of the vectors

$$ {}^{\text{HCS}}g \in \mathbb{R}^3, \qquad {}^{\text{HCS}}N \in \mathbb{R}^3 \tag{1} $$

the first of which is supposed to reflect the direction of the earth's gravity field vector and the second to lie in the MCS-x-z-plain (N stands for **north**).

From these two vectors, MCS base vectors can easily be calculated:

$$ \begin{aligned} {}^{\text{HCS}}\text{MCS}_z &= -\frac{{}^{\text{HCS}}g}{|{}^{\text{HCS}}g|} && (2)\\ {}^{\text{HCS}}\text{MCS}_y &= \frac{{}^{\text{HCS}}N \times {}^{\text{HCS}}g}{|{}^{\text{HCS}}N \times {}^{\text{HCS}}g|} && (3)\\ {}^{\text{HCS}}\text{MCS}_x &= {}^{\text{HCS}}\text{MCS}_y \times {}^{\text{HCS}}\text{MCS}_z && (4) \end{aligned} $$

yielding the orthonormal orientation matrix for MCS in HCS, or HCS in MCS after transposition.

In the MARVIN system, the convention for compact representation of the UAV's orientation is a vector $\alpha' \in \mathbb{R}^3$ of Euler angles for rotating MCS about the sequence of axes z, y', x'' to arrive at HCS. Formally, using orthonormal rotation matrizes:

$$ {}^{\text{MCS}}\text{HCS} = R_z(\alpha'_3) \cdot R_y(\alpha'_2) \cdot R_x(\alpha'_1) \tag{5} $$

This can be unambigiously solved in α' for all orientations where HCS_x is not exactly vertical using:

$$ \alpha' = \begin{pmatrix} \text{atan2}({}^{\text{MCS}}\text{HCS}_{3,2}, {}^{\text{MCS}}\text{HCS}_{3,3}) \\ \arcsin(-{}^{\text{MCS}}\text{HCS}_{3,1}) \\ \text{atan2}({}^{\text{MCS}}\text{HCS}_{2,1}, {}^{\text{MCS}}\text{HCS}_{1,1}) \end{pmatrix} \tag{6} $$

As the result of orientation sensing, actually the Euler angle vector α for the z, y', x''-rotation from BCS to HCS is needed. Given the the local magnetic deviation d, it is

$$ {}^{\text{BCS}}\text{MCS} = R_z(-d) \tag{7} $$

(negative sign resulting from mathematical versus compass direction of rotation) such that

$$
\begin{aligned}
{}^{\mathrm{BCS}}\mathrm{HCS} &= {}^{\mathrm{BCS}}\mathrm{MCS} \cdot {}^{\mathrm{MCS}}\mathrm{HCS} && (8)\\
&= R_z(-d) \cdot R_z(\alpha_3') \cdot R_y(\alpha_2') \cdot R_x(\alpha_1') && (9)\\
&= R_z(\alpha_3' - d) \cdot R_y(\alpha_2') \cdot R_x(\alpha_1') && (10)
\end{aligned}
$$

Thus, the final representation of the UAV's orientation will be

$$\alpha = (\alpha_1', \alpha_2', \alpha_3' - d)^{\mathrm{T}} \tag{11}$$

It remains to be seen how the state vectors ${}^{\mathrm{HCS}}g$ and ${}^{\mathrm{HCS}}N$ can be determined resp. updated.

3.2 On-Line Calibration

For drift-free on-line calibration of the orientation measurement, the accelerometer and compass readings have to be used. The accelerometer readings need to be adjusted by eliminating the effects of the dynamic acceleration, which in turn can only be estimated for some time in the past based on the DGPS output. On-board MARVIN, a DGPS position fix is obtained every $T = 200$ ms. Therefore, the estimation of the dynamic acceleration (which is naturally carried out in the Newtonian BCS frame) can be calculated at time t according to:

$$
\begin{aligned}
{}^{\mathrm{BCS}}v(t - \frac{T}{2}) &= {}^{\mathrm{BCS}}s(t) - {}^{\mathrm{BCS}}s(t-T) && (12)\\
{}^{\mathrm{BCS}}a(t-T) &= {}^{\mathrm{BCS}}v(t - \frac{3}{2}T) - {}^{\mathrm{BCS}}v(t - \frac{T}{2}) && (13)
\end{aligned}
$$

thus yielding an estimation delayed by T.

Now, a calibration reference can be calculated for time $t-T$ based upon the sensor readings a_s and B_s for acceleration and magnetic field, respectively:

$$
\begin{aligned}
{}^{\mathrm{HCS}}g^*(t-T) &= {}^{\mathrm{HCS}}a_s(t-T) + {}^{\mathrm{HCS}}\mathrm{BCS}(t-T) \cdot {}^{\mathrm{BCS}}a(t-T) && (14)\\
{}^{\mathrm{HCS}}N^*(t-T) &= {}^{\mathrm{HCS}}B_s(t-T) && (15)
\end{aligned}
$$

From these two vectors, a "corrected" past Euler vector $\alpha^*(t-T)$ is calculated according to (2-4, 6, 11). Compared to the stored past Euler vector $\alpha(t-T)$, the detected calibration error for time $t-T$ is:

$$\Delta\alpha(t-T) = \alpha(t-T) - \alpha^*(t-T) \tag{16}$$

Please note that a stored orientation frame ${}^{\mathrm{HCS}}\mathrm{BCS}(t-T)$ must be used in (14) to transform the dynamic acceleration from the BCS to the HCS frame. This stored frame is affected by the calibration error as well. However, overall convergence of the calibration process is not impeded by this for moderate calibration errors.

3.3 State Update

After having looked at the state description and the calculation of the calibration error, the update process of the two state vectors ${}^{\mathrm{HCS}}g$ and ${}^{\mathrm{HCS}}N$ remains to be clarified.

As the drift-free orientation calculation described in the previous section is afflicted with a delay of T, the rotation sensors are required to incorporate "short-term" rotations into the measurement. Actually, the two state vectors, which conveniently refer to HCS, have to be rotated according to the rotation sensors' readings $\omega(t) \in \mathbb{R}^3$. Additionally, due to the Euler nature of the calibration error determined above, it is an appealing (while only approximately correct) simplification to combine the sensor readings and the calibration error into one single 3D rotation.

Before this occurs, some low-pass-filtering is applied to the calibration error. This allows to reduce the effect of sensor noise on the orientation data output while maintaining a comfortably high compensation rate in case of significant persistant drift of the integrated rotation readings:

$$\overline{\Delta\alpha}(t) = (1 - f_1) \cdot \overline{\Delta\alpha}(t - T_c) + f_1 \cdot \Delta\alpha(t - T) \tag{17}$$

Here, T_c denotes the time increment for every calculation cycle, which is $T_c = 1/20$ s for the MARVIN system. The actual filter factor used is $f_1 = 1/32$.

Now, the angle vector for the rotation update computes as:

$$\beta = f_2 \cdot \overline{\Delta\alpha}(t) - \omega(t) T_c \tag{18}$$

Here, $f_2 < 1$ is a weighting factor necessary to prevent overcompensation and divergence. The value used is $f_2 = 1/32$ as well. For stability, $f_2 \leq T_c/T$ is strictly required, otherwise the error compensation during the time interval $[t - T \ldots t]$ might exceed the actual error at $t - T$.

For accuracy reasons set forth in the next section and connected to the integer arithmetics used, the final rotation update is performed according to the approximate law:

$$\begin{aligned} {}^{\mathrm{HCS}}g(t) &= R_{P_t(x)}(\beta_{P_t(x)}) \cdot R_{P_t(y)}(\beta_{P_t(y)}) \cdot R_{P_t(z)}(\beta_{P_t(z)}) \cdot {}^{\mathrm{HCS}}g(t - T_c) \qquad (19) \\ {}^{\mathrm{HCS}}N(t) &= R_{P_t(x)}(\beta_{P_t(x)}) \cdot R_{P_t(y)}(\beta_{P_t(y)}) \cdot R_{P_t(z)}(\beta_{P_t(z)}) \cdot {}^{\mathrm{HCS}}N(t - T_c) \qquad (20) \end{aligned}$$

with $P_t : \{x;y;z\} \rightarrow \{x;y;z\}$ some *random* permutation of the axes, drawn anew in every computation cycle. This imposes an arbitrary random order on the rotations that should ideally occur concurrently.

In order to compensate for accumulated errors resulting from the periodic application of this update rule, the following renormalization is applied once every 128 calculation cycles. It adjusts the vectors to unit length and ensures that they are perpendicular:

$$\begin{aligned} {}^{\mathrm{HCS}}g(t)_{\mathrm{norm}} &= \frac{{}^{\mathrm{HCS}}g(t)}{|{}^{\mathrm{HCS}}g(t)|} \qquad (21) \\ {}^{\mathrm{HCS}}N(t)_{\mathrm{norm}} &= \frac{{}^{\mathrm{HCS}}N(t) - ({}^{\mathrm{HCS}}g(t) \cdot {}^{\mathrm{HCS}}N(t))\,{}^{\mathrm{HCS}}g(t)}{|{}^{\mathrm{HCS}}N(t) - ({}^{\mathrm{HCS}}g(t) \cdot {}^{\mathrm{HCS}}N(t))\,{}^{\mathrm{HCS}}g(t)|} \qquad (22) \end{aligned}$$

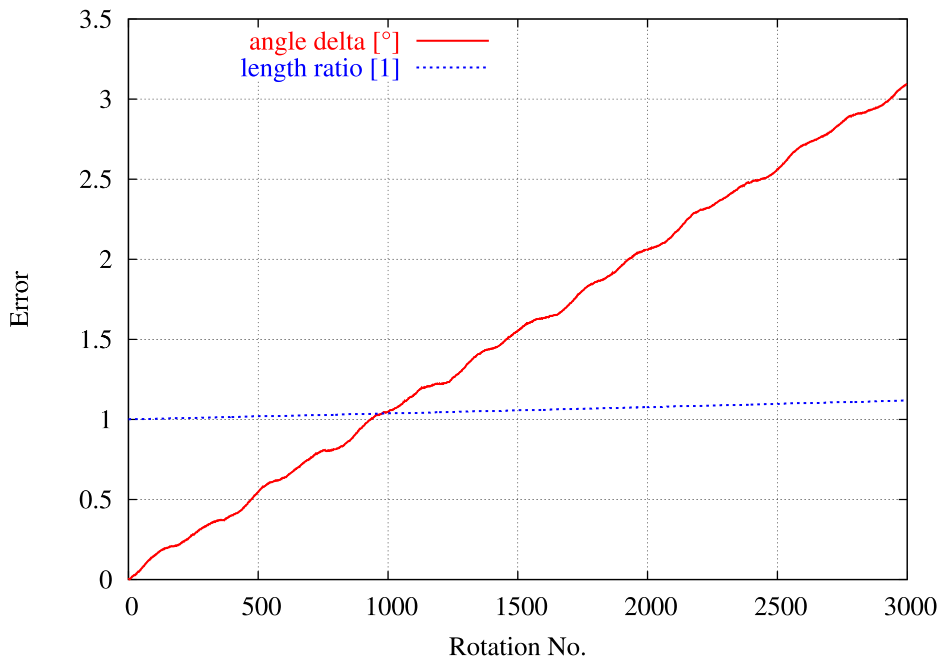
Finally, the orientation output $\alpha(t)$ is derived from the two state vectors according to (2-4, 6, 11).

4 Analysis

In this section of the paper, the MARVIN orientation fusion algorithm described in the previous section is analyzed for the effects of the deliberately chosen simplifications, and for its accuracy and computational cost compared to a Kalman Filter as the usual "state-of-the-art" method.

Rotation Scheme	**multiply/divide**	**add/sub**	**sin/cos**	**square root**
random β rotation (*correct*)	8	9	0	2
subsequent x, y, z rotations (*used*)	2	2	1	0

Table 1: Longest computation path comparison for "correct" and "incorrect" implementations of ${}^{\mathrm{HCS}}g$ and ${}^{\mathrm{HCS}}N$ rotation



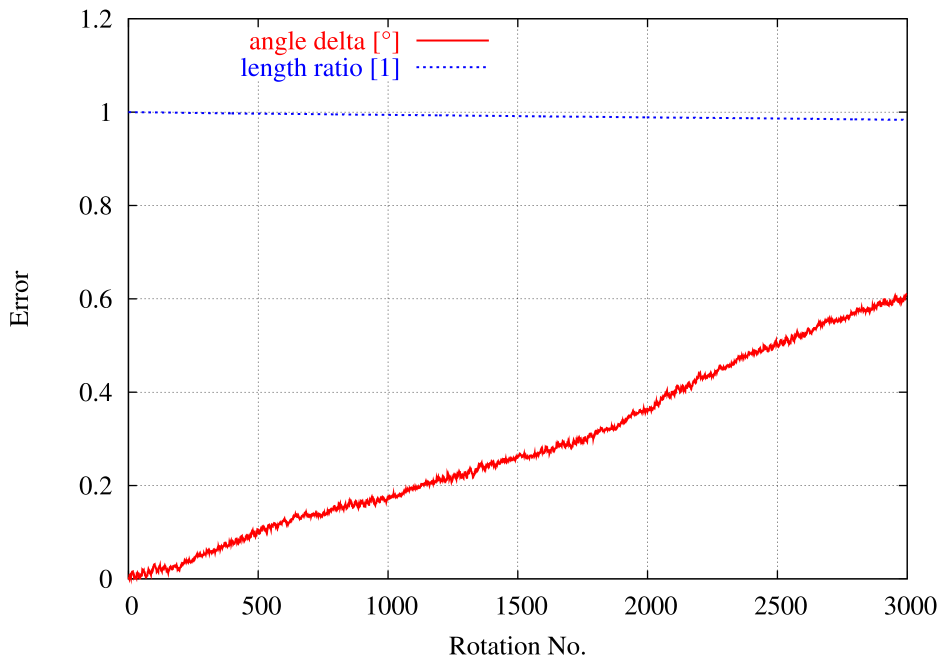


Figure 2: "Random β rotation" (top) and "subsequent x, y, z rotation" (bottom) errors using 32 bit integer arithmetics

4.1 Simplifications

Even when assuming error-free sensor readings, there are three simplification issues in the algorithm:

- The acceleration estimation according to (13, 12) gives only an average over an interval, not a point-of-time acceleration.
- The combination of the measured rotation and the calibration error according to (18) uses (slightly) incorrect axes for the calibration part.
- The state vector rotation according to (19, 20) is mathematically different from the true "concurrent" rotation about an axis parallel to β by an angle of $|\beta|$.

The first simplification is somewhat inevitable and about the best thing one can do considering the nature of the GPS data. The second one has proven acceptable on the one hand and will appear particularly reasonable on the other hand after the analysis of the third one (see below), because the number of rotations actually performed is minimized.

The third simplification requires more reflection to be justified. The critical property of the β rotation in every computation cycle is a relatively small rotation angle $|\beta|$, which minimizes the error caused by the sequential instead of concurrent execution of the rotation components. The concurrent rotation about a random axis requires far more complex computation than 3 subsequent rotations around BCS axes. Two already optimized implementations of the β-rotation of a single $\mathbb{R}^3$ vector, one formally correct and the other as described in the previous section, are compared in table 1 with respect to the number of basic operations on the *longest path* in the computation trees. These numbers reflect the *maximum number of subsequent operations* carried out on some input value to determine a component of the result vector, providing a suitable measure of expected inaccuracy due to rounding errors. (Due to the iterative calculation of square roots, these have been listed seperately. Otherwise, their effect on the accuracy of the "random β rotation" would be overestimated.)

Obviously, the "correct" implementation is prone to far more rounding error than the simplified one, especially because of four times the number of chained multiplications and divisions. To back this code-level comparison also practically, simulations of the two algorithms using 32 bit integer arithmetics have been carried out on a PC to evaluate the accumulating error with a proper "random β rotation" computed in 64 bit floating point. Figure 2 shows the accumulating angle error (angular deviation of calculated and true vector) and the accumulating length error (relation of calculated and true vectors' lengths) for the two algorithms. In this experiment, 3000 subsequent rotations about 7 different angles between 0.03° and 1.6° per axis (average of 0.50°) are performed. This range of rotation angles is in the order of magnitude found in the MARVIN case with 20 orientation computations per second. The angles used in this experiment are only an example, of course, but the result is not qualitatively different for any other set of angles.

The figure shows that the "mathematically incorrect" implementation actually yields significantly better results. Evidently, the factor 4 derived from table 1 is a satisfactory estimation of the relation of the resulting errors. Thus, the error caused by "incorrect" sequential rotating is clearly dominated by the rounding error.

4.2 Accuracy

For comparison of MARVIN's orientation fusion algorithm with a Kalman filter [4], such a filter has been implemented not for the on-board microcontroller but only for simulation purposes on different computer platforms.

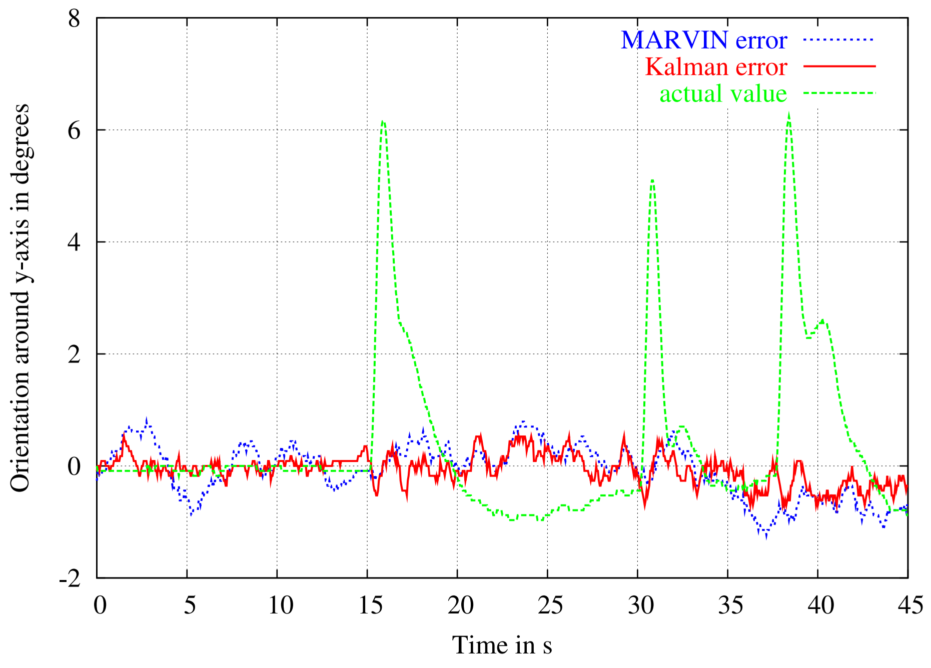


Figure 3: Deviation of orientation around y-axis of MARVIN's algorithm and the EKF together with the real orientation

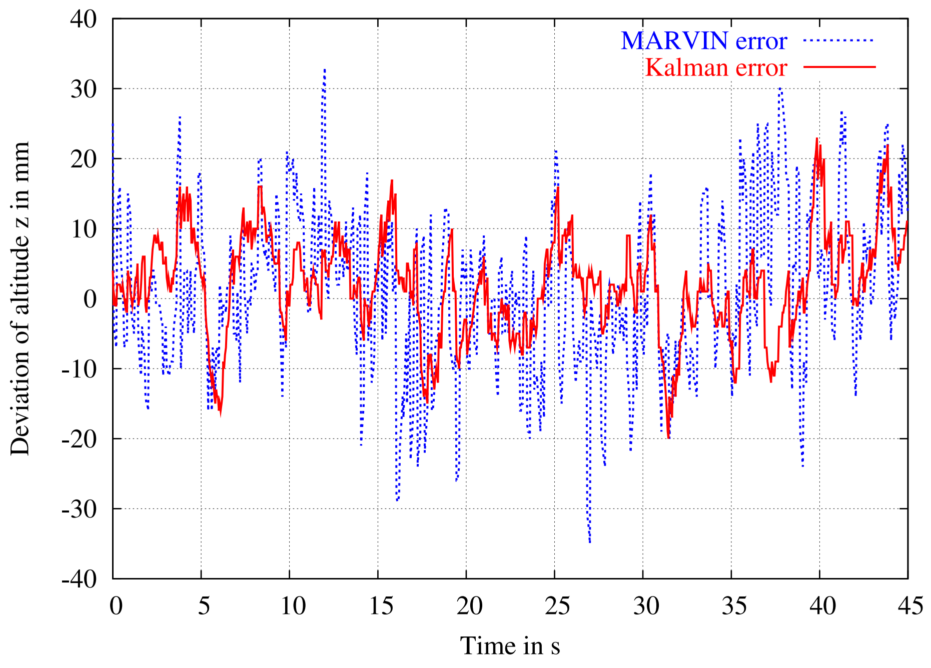


Figure 4: Deviation of vertical position of both algorithms

The implemented Extended Kalman Filter (EKF) combines all described sensors (GPS, acceleration, rotation and compass), which results in a 12D measurement vector. They are used to estimate a 15D state vector which contains position, velocity, acceleration, orientation and angular velocity of the helicopter.

For the prediction of the state vector a simple model of a rigid body is used. The prediction is performed by the Kalman-Bucy-Filter [5] algorithm for continous systems together with numerical integration over time using the approach of Runge-Kutta. The update of the state vector using the measurements is perfomred using a common EKF approach.

There are no simplifications of the Kalman algorithm. For implementation of the algorithm a 15×15-matrix has to be inverted, which is done numerically.

Figure 3 shows the resulting deviations for the orientation of the helicopter around its y-axis produced by MARVIN's integer algorithm described above and the EKF. These plots are created using a simulated flight and emulating the sensors with known errors such as noise. Together with the two filters the actual orientation is shown. The peaks in the orientation indicate flight maneuvers since they mean an acceleration along the helicopter's x-axis.

Figure 4 depicts the same situation for the vertical position of the helicopter. MARVIN's fusion algorithm uses simple linear extrapolation of GPS position data between GPS measurements.

During the whole time both algorithms produce comparable results. Even when there are true movements, neither the EKF nor MARVIN's algorithm are affected by this. The resulting deviations do not change due to movements.

Since the microcontroller of MARVIN does not have an FPU, it is not possible to port the tested EKF directly to be used on-board. And even if there were an FPU, the execution time of the EKF is much higher than for MARVIN's algorithm: On an Intel Pentium 4 with $2.8GHz$, the calculation of the EKF takes 15 ms but MARVIN's algorithm only takes 9 μs. So complexity is about 1700 times higher for the EKF and much too high for the computational power on-board.

Comparing the quality of both approaches please keep in mind another aspect: For simulation the sensors are emulated with exactly the same noise as expected by the EKF. This means the results will be slighty worse in reality for the EKF than presented here.

For the future it is planned to extend the model inside the EKF with small parts of the helicopter model. So the filter can make better predictions as it knows how a helicopter will move due to inputs from the controller. But the problem of the necessary computational power will even increase. This means it will not be used on MARVIN's microcontroller.

5 Conclusion

This paper shows that UAV orientation sensing with an accuracy suitable for helicopter flight control is possible under strict resource constraints. Furthermore, in certain circumstances often encountered in micro UAVs it might be advisable to use simplified computations and integer arithmetics instead of theoretically superior but computationally far more demanding methods like the Kalman filter.

The work described in this paper has been funded in part by the European Community in the Information Society Technologies Program under contract COMETS IST-2001-34304 [1].

References

[1] COMETS consortium. Real-time coordination and control of multiple heterogeneous unmanned aerial vehicles. IST-2001-34304, 2002–2005. http://www.comets-uavs.org/.

[2] Association for Unmanned Vehicles International. International Aerial Robotics Competition: The Robotics Competition of the Millennium. http://avdil.gtri.gatech.edu/AUVS/2000AerialRoboticsCompInfo.html/2000EventInfo.html.

[3] Real Time Systems & Robotics Group. MARVIN – An autonomously operating flying robot. http://pdv.cs.tu-berlin.de/MARVIN/. TU Berlin.

[4] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME Journal of Basic Engineering*, 82:35–45, 1960.

[5] R.E. Kalman and R.S. Bucy. New results in linear filter and prediction theory. *Journal of Basic Engineering*, pages 95–108, 1961.

[6] M. Knoke, C. Reinicke, V. Remuß, M. Musial, U. W. Brandenburg, G. Hommel. Entwicklung einer Orientierungssensorik für einen fliegenden Roboter. In *Robotik 2000, Tagung Berlin 29. und 30. Juni 2000*, VDI-Bericht Nr. 1552, pages 537–544, Düsseldorf, 2000. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, VDI-Verlag. ISBN 3-18-091552-8.

[7] G. Hommel M. Musial, U. W. Brandenburg. Das Kommunikationskonzept für MARVIN, den autonom fliegenden Erkundungsroboter der TU Berlin. In *Robotik 2000, Tagung Berlin 29. und 30. Juni 2000*, VDI-Bericht Nr. 1552, pages 545–551, Düsseldorf, 2000. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, VDI-Verlag. ISBN 3-18-091552-8.

[8] Marek Musial, Uwe Wolfgang Brandenburg, Günter Hommel. Cooperative Autonomous Mission Planning and Execution for the Flying Robot MARVIN. In Enrico Pagallo, Frans Groen, Tamio Arai, et al., editors, *Intelligent Autonomous Systems 6*, pages 636–646, Amsterdam, Berlin, Oxford, et al., 2000. IOS Press and Ohmsha. ISBN 1-58603-078-7.

[9] Marek Musial, Uwe Wolfgang Brandenburg, Günter Hommel. Inexpensive system design: The flying robot MARVIN. In *Unmanned Air Vehicle Systems: Sixteenth International UAVs Conference*, pages 23.1–23.12, Bristol, UK, 2001. University of Bristol. ISBN 0-86292-517-7.