Proceedings of the International Multiconference on
Computer Science and Information Technology pp. 783–790
ISBN 978-83-60810-27-9
ISSN 1896-7094

Computationally effective algorithms for 6DoF INS used for miniature UAVs

Jan Floder
VSB – Technical University of Ostrava, Department of Measurement and Control, 17. listopadu 15, 70833 Ostrava, the Czech republic
Email: jan.floder@vsb.cz

***Abstract*—The article aims at 6 degrees of freedom inertial navigation systems for miniature UAVs. It shows a new filter design which replaces standard solutions represented by EKF/UKF filters. The new filter is designed to significantly reduce filter complexity and processing power requirements (but keeping estimation accuracy) to be useful in small embedded systems with minimum processing power.**

I. Introduction

DEVELOPMENT in semiconductor technologies in past several decades brings technologies which used to belong to military/hi-tech domain to everyday use in variety of fields.

This includes also a development of inertial measurement related sensors like MEMS technology sensors (accelerometers, gyrometers), tiny but powerful GPS modules, small magnetometers, barometric sensors and of course fast and small processors. The sensors are used for inertial navigation systems which provide information about vehicle orientation and movement in space. These navigation systems are used for example in autonomous robots, ground, underwater and aerial vehicles.

Although all sensors mentioned above undergo rapid development accuracy is still far beyond their high-end counterparts and traditional technologies. So the sensors themselves used in inertial measurement units (abbreviation IMU) are not able to provide acceptable navigation results over a longer period of time. To solve the issue GPS measurements have to be integrated to the INS measurements to keep navigation results usable (in contrary to INS output errors, GPS measurement errors are not function of time). This solution is known as a GPS aided inertial navigation. The sensor integration and INS information calculation is usually handled by an EKF (Extended Kalman Filter) or an UKF (Unscented Kalman Filter) with adaptive gain. But these filters have drawbacks in certain situations. Major drawback is their relative complexity and requirements of powerful processors.

The article tries to describe and explain ways how replace traditional INS algorithms (represented by EKF, UKF) by algorithms which are more suitable for miniature unmanned aerial vehicles (UAVs) which have limited processing power. The article continues in development of the vector filter algorithm presented in [1].

A. INS

The INS does estimate orientation, acceleration, velocity and position of a vehicle moving anywhere on earth (or close to its surface). This is done by measurement and integration of several motion parameters. There are several types of inertial navigation but the article is aimed at a GPS aided tightly coupled inertial navigation for systems with 6 degrees of freedom of motion. Six degrees of freedom of motion means that vehicles, where the navigation is applied on, can "virtually freely" move and rotate in any direction. The only omitted parameter are huge long term centripetal accelerations because UAVs the article is aimed at do not undergo them.

The motion equations for a UAV are (presented in form prepared for inertial sensors)

$$\begin{aligned}
\dot{\boldsymbol{q}}_{RPY\to ENU} &= \boldsymbol{q}_{RPY\to ENU} * \frac{1}{2}\boldsymbol{\omega}_{RPY} \\
\dot{\boldsymbol{v}}_{ENU} &= -\boldsymbol{g}_{ENU} + \boldsymbol{C}_{RPY\to ENU}(\boldsymbol{q}_{RPY\to ENU})\cdot\boldsymbol{a}_{RPY} \\
\dot{\boldsymbol{p}}_{ECEF} &= \boldsymbol{C}_{ENU\to ECEF}(\boldsymbol{p}_{ECEF})\cdot\boldsymbol{v}_{ENU} \\
\boldsymbol{p}_{LLA} &= \begin{bmatrix} \varphi_{WGS84} \\ \lambda_{WGS84} \\ h_{WGS84} \end{bmatrix} = \boldsymbol{f}_{ECEF\to WGS84}(\boldsymbol{p}_{ECEF})
\end{aligned} \tag{1}$$

Vehicle motion parameters velocity vector $\boldsymbol{v}_{ENU}$ and position vector $\boldsymbol{p}_{ECEF}$ (and $\boldsymbol{p}_{LLA}$) are derived from vehicle acceleration vector $\boldsymbol{a}_{RPY}$ which is measured by a MEMS accelerometer (measurement contains gravity acceleration part which has to be subtracted - $\boldsymbol{g}_{ENU}$). Vehicle orientation quaternion $\boldsymbol{q}_{RPY\to ENU}$ is derived from angular rate vector $\boldsymbol{\omega}_{RPY}$ measured by a MEMS gyrometer.

The quaternion $\boldsymbol{q}_{RPY\to ENU}$ stores rotation between vehicle local frame RPY and earth local frame ENU [2]. The position is calculated in earth global frame ECEF and alternatively in LLA (longitude, latitude, altitude) coordinates (usually the WGS84 Earth model). For coordinate systems and transformations ($\boldsymbol{C}_{RPY\to ENU}$, $\boldsymbol{C}_{ENU\to ECEF}$) consult next section.

As shown above accelerometer and gyrometer are enough to calculate all parameters of the inertial navigation. But this is valid for ideal sensors only. In the real world there are many sources of errors which cripple navigation results. In the equation you can see that acceleration and angular rate are integrated to provide vehicle orientation, velocity and

position. So measurement errors (like offset) are integrated too and do rise in time. That is why magnetic field and GPS measurements have to come to stage.

B. Coordinate Systems and Transformations

The equation 1 represents INS parameters in several coordinate systems. They can be divided in 3 groups.

1. Vehicle local Cartesian coordinate system RPY (Roll-Pitch-Yaw) is fixed to vehicle with center in its center of gravity.

2. Earth local Cartesian coordinate system ENU/NED (East-North-Up / North-East-Down). This is a local coordinate system related to earth.

3. Earth global coordinate system ECEF, LLA (Earth-Centered-Earth-Fixed and Longitude-Latitude-Altitude). The ECEF is a Cartesian coordinate system and the LLA is a spherical (or elliptical) coordinate system.

For complete definitions consult for example [2] and [7].

Coordinate transformations transform a vector represented in one coordinate system to another and vice versa (for example $RPY \rightarrow ENU$), so they can describe vehicle orientation in space (when considering rotations only).

There are several ways to calculate/represent coordinate transformations. The best suited (for INS systems) are [2]:

- Rotation matrix $\boldsymbol{C}_{from \rightarrow to} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$

- Quaternion $\boldsymbol{q}_{from \rightarrow to} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T$

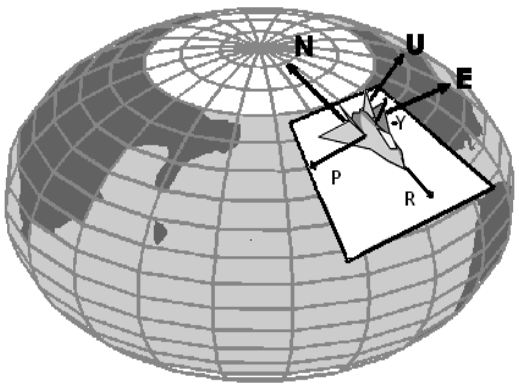

Fig 1. Illustration of coordinate systems relations (RPY, ENU, LLA).

The figure 1 illustrates RPY, ENU and LLA coordinate systems.

II. Sensors

Modern MEMS sensors are small in size (usually below 1cm2) require minimum external components (supply filter capacitor and some output low-pass single pole filters), single voltage supply and consume minimum power (below 1mA). But they also have drawbacks. As said before they are not accurate enough for classical INS.

The main sources of INS error related to MEMS sensors are additive white noise $\boldsymbol{\eta}$, temperature, random and high frequency vibration bias drift and offset $\boldsymbol{b}$, scaling error $\boldsymbol{s}$ and cross axis sensitivity $\boldsymbol{c}$.

A. MEMS Accelerometers

Bearing in mind the equation 1, accelerometer output is transformed from vehicle coordinates (RPY) to local earth coordinates (ENU) and integrated to obtain vehicle velocity (and position). Ignoring possible attitude estimation error $\delta \boldsymbol{q}_{RPY \rightarrow ENU}$ (see section *INS Algorithms*) there are three types of accelerometer errors which corrupt velocity estimation in time. It is a drift & offset, mis-scaling and cross-axis sensitivity. According to practical measurements (3 axis ADXL335) these errors may lead up to 0.3 $m \cdot s^{-2}$ (each axis) ignoring high frequency vibrations drift - in helicopter-like UAVs, where powerful engines are needed to create enough lift, there is a danger of strong vibration bias drift due to motor vibrations (close to resonant frequency of sensor's mechanical parts). The only way to avoid it is a mechanical damping.

B. MEMS Gyrometers

Gyrometer outputs are integrated to obtain attitude information from angular rate. The most important error, which is usually estimated and compensated by the INS filter, is a bias drift $\boldsymbol{b}_g$. For modern gyrometers overall drift does not usually exceed 0.5 deg/s (MLX90609) but initial bias may differ on every run so it is worthy to estimate it. There may also be a danger of a high frequency vibration drift but as long as resonant frequencies of gyrometer mechanical parts are higher compared to accelerometers the danger is smaller.

C. Magnetometers

Modern magneto-resistive or magneto-inductive magnetometers are size and power consumption friendly too. But in their case another type of errors is most important. It is a hard iron distortion and a soft iron distortion. The distortions are caused by close sources of magnetic field (like DC motor magnets) or magnetic materials (chassis, shielding, etc.). Without compensation these errors lead in overall error that may exceed even 20 degrees. But ways how to compensate these errors are not subject of the article.

D. Miniature GPS Modules

GPS modules just receive and process navigation data from geostationary satellites (and in some cases additional navigation data from ground stations too). They are also small (especially models with integrated antennas) and may weight even less than 20 grams. They output actual position and velocity information with accuracy of several meters (RMS) for position and tenths of meters per second for 2D/3D velocity (later GPS chipsets, like μBlox-5, provide 3D velocity estimations).

III. INS Algorithms

At first a standard tightly coupled INS based on the multiplicative EKF (MEKF) design is presented. It was chosen

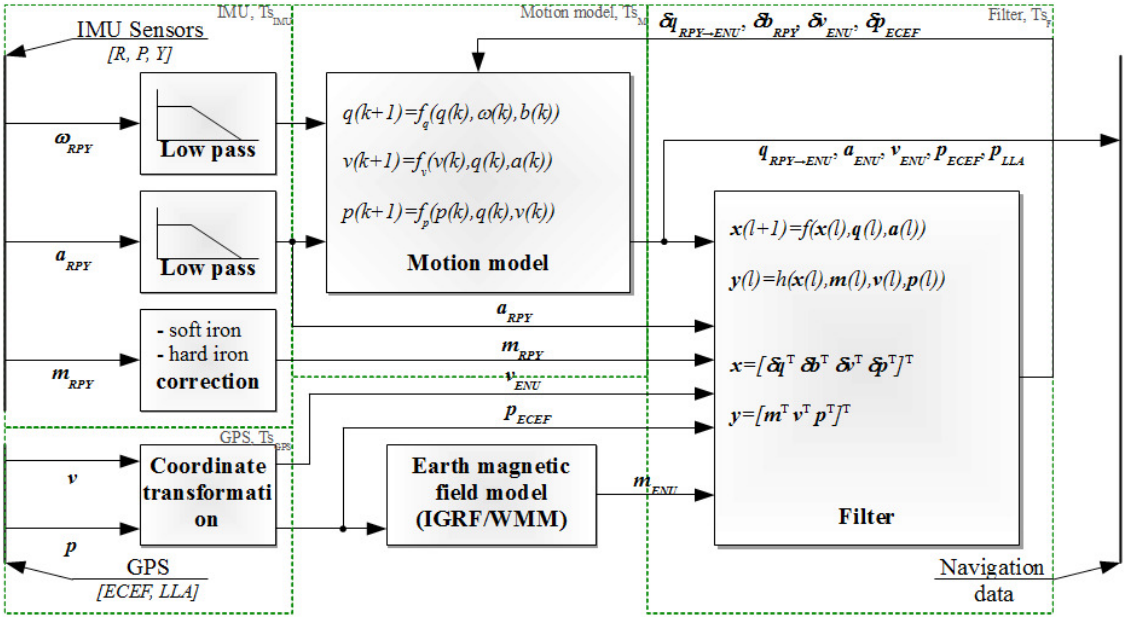

Fig 2. Tightly coupled GPS aided INS (based on error model) block diagram.

due to its elegance (relatively "linear" model) and "simplicity" (sparse matrices reduce number of arithmetic operations needed for calculations) which enables it to be used in non-powerful processors unlike some other solutions (like for example GP-UKF). The MEKF also represents a standard navigation solution which is used as a reference to the navigation algorithm proposed in this article. The figure 2. shows a tightly coupled INS diagram.

A. Error Model

The MEKF INS filter, which can be found for example at [4] is based on an error model where the motion model itself is calculated alone and only model errors are subject of estimation (by the MEKF). So using accelerometer and gyrometer measurement and equation 1 the INS estimates vehicle orientation in space $\tilde{\boldsymbol{q}}_{RPY\to ENU}$, velocity in local earth coordinates $\tilde{\boldsymbol{v}}_{ENU}$ and vehicle position on earth $\tilde{\boldsymbol{p}}_{ECEF}$ (and in more common longitude, latitude and altitude coordinate system $\tilde{\boldsymbol{p}}_{LLA}$). Because there are errors which corrupt INS outputs, estimation errors are calculated according to [4]

$$\begin{aligned}
\delta\dot{\boldsymbol{q}}_{RPY\to ENU} &= \frac{1}{2}\delta\boldsymbol{b}_{RPY}\times\delta\boldsymbol{q}_{RPY\to ENU} - \frac{1}{2}\delta\boldsymbol{b}_{RPY} - \frac{1}{2}\boldsymbol{\eta}_g \\
\delta\dot{\boldsymbol{b}}_{RPY} &= \boldsymbol{\eta}_b \\
\delta\dot{\boldsymbol{v}}_{ENU} &= -2\boldsymbol{C}(\tilde{\boldsymbol{q}}_{RPY\to ENU})\boldsymbol{a}_{RPY}\times\delta\boldsymbol{q}_{RPY\to ENU} - \boldsymbol{C}(\tilde{\boldsymbol{q}}_{RPY\to ENU})\boldsymbol{\eta}_a \\
\delta\dot{\boldsymbol{p}}_{ECEF} &= \boldsymbol{C}_{ENU\to ECEF}(\boldsymbol{p}_{ECEF})\delta\boldsymbol{v}_{ENU}
\end{aligned} \tag{2}$$

The filter is constructed of 4 state vectors (12 state variables) which serve to estimate: attitude error vector $\delta\boldsymbol{q}_{RPY\to ENU}$ which contains only vector part of the quaternion because for small errors the scalar part is ≈ 1 – see [4], gyrometer bias drift $\delta\boldsymbol{b}_{RPY}$, velocity estimation error $\delta\boldsymbol{v}_{ENU}$ and position estimation error $\delta\boldsymbol{p}_{ECEF}$. $\boldsymbol{\eta}_g$ for a gyrometer, $\boldsymbol{\eta}_b$ for a gyrometer bias drift, $\boldsymbol{\eta}_a$ for an accelerometer represent sensor white noise acting on the error model. The filter is updated (every step) by a magnetometer measurement $\boldsymbol{m}_{RPY}$ [4] which is compared to expected magnetic filed vector $\boldsymbol{m}_{ENU}$ for given location,

$$|\boldsymbol{m}_{RPY}| = \boldsymbol{A}(\delta\boldsymbol{q}_{RPY\to ENU})\cdot\left(\boldsymbol{C}(\tilde{\boldsymbol{q}}_{RPY\to ENU})^T\cdot|\boldsymbol{m}_{ENU}|\right) \tag{3}$$

$$\boldsymbol{A}(\delta\boldsymbol{q}) = \begin{bmatrix} 1 & 2\delta\boldsymbol{q}[3] & -2\delta\boldsymbol{q}[2] \\ -2\delta\boldsymbol{q}[3] & 1 & 2\delta\boldsymbol{q}[1] \\ 2\delta\boldsymbol{q}[2] & -2\delta\boldsymbol{q}[1] & 1 \end{bmatrix} \tag{4}$$

When a GPS measurement is available the filter is also updated by a measured velocity vector in ENU frame $\boldsymbol{v}_{ENU}$ and a position vector $\boldsymbol{p}_{ECEF}$ in ECEF frame.

$$\begin{bmatrix} \boldsymbol{v}_{ENU} \\ \boldsymbol{p}_{ECEF} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{v}}_{ENU} + \delta\boldsymbol{v}_{ENU} \\ \tilde{\boldsymbol{p}}_{ECEF} + \delta\boldsymbol{p}_{ECEF} \end{bmatrix} \tag{5}$$

These estimated INS solution errors have to be used to correct INS data (the INS data are calculated using equation 1). The correction is

$$\begin{aligned}
&\tilde{\boldsymbol{q}}_{RPY\to ENU} = \tilde{\boldsymbol{q}}_{RPY\to ENU} * \delta\boldsymbol{q}_{RPY\to ENU} \to \delta\boldsymbol{q}_{RPY\to ENU} = 0 \\
&\tilde{\boldsymbol{v}}_{ENU} = \tilde{\boldsymbol{v}}_{ENU} + \delta\boldsymbol{v}_{ENU} \to \delta\boldsymbol{v}_{ENU} = 0 \\
&\tilde{\boldsymbol{p}}_{ECEF} = \tilde{\boldsymbol{p}}_{ECEF} + \delta\boldsymbol{p}_{ECEF} \to \delta\boldsymbol{p}_{ECEF} = 0
\end{aligned} \tag{6}$$

The errors can be transformed from the filter to INS parameters estimations and the errors themselves are set to zero [4], [5] after (every) filter step. The $\tilde{\boldsymbol{q}} * \delta\boldsymbol{q}$ may be rewritten as (derived from [4])

$$\tilde{\boldsymbol{q}} * \delta\boldsymbol{q} = \begin{bmatrix} -\tilde{\boldsymbol{q}}[2] & -\tilde{\boldsymbol{q}}[3] & -\tilde{\boldsymbol{q}}[4] \\ \tilde{\boldsymbol{q}}[1] & -\tilde{\boldsymbol{q}}[4] & \tilde{\boldsymbol{q}}[3] \\ \tilde{\boldsymbol{q}}[4] & \tilde{\boldsymbol{q}}[1] & -\tilde{\boldsymbol{q}}[2] \\ -\tilde{\boldsymbol{q}}[3] & \tilde{\boldsymbol{q}}[2] & \tilde{\boldsymbol{q}}[1] \end{bmatrix} \delta\boldsymbol{q} + \tilde{\boldsymbol{q}} \tag{7}$$

According to the general rule in quaternion calculations the quaternion $\tilde{\boldsymbol{q}}$ should be normalized always after several calculations to avoid accumulation of floating-point round-off errors (valid especially for 32-bit floating-point).

B. (M)EKF/UKF

The error model merits of relatively simple and quite "linear" equations, and sparse matrices when using EKF. But besides this there is one more advantage. Because motion model and error estimation are separated they can be calculated with different sampling periods (bearing in mind S-N-K theorem) – motion model can be calculated for example 100 times per second and the error estimation just 20 times per second. This does reduce arithmetic operations requirements even more. The MEKF is an EKF variant where the attitude estimation error quaternion is multiplied (not added) to the model [5] when every other filter aspect remains the same.

The EKF and UKF (adaptive versions) equations are well known and can be found among literature. The UKF has one disadvantage and this is that it cannot take advantage of spare matrices. Some example INS solution can be found for example at [6] or [9].

C. Vector Filter

First of all complete explanation of the vector filter is too long to fit in the article so it is described only briefly. Some basic principles are described in [1]. The EKF/UKF filter is a working solution proven by time and experience (used over 40 years) but for purposes of miniature UAVs there are some drawbacks. Embedded systems for these UAVs tend to be very cheap, small and light and all processing (basic signal processing, navigation and control) is usually done by a single System on Chip processor which should also have minimum power consumption. This means that processing power for the navigation is significantly reduced. Besides this hurdle EKF/UKF configureability is somewhat limited (it is easy to alter $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices but impossible to adjust convergence mechanisms freely).

This is the main reason why the vector filter was developed. First stage was development of an IMU filter to provide orientation estimation for a loosely coupled INS filter (described in [1]). This article describes enhancement to a tightly coupled and mixed solution INS. The filter has these main features:

a) It is based on vectors rather than single variables. It means that $\delta \boldsymbol{q}_{\boldsymbol{RPY}\rightarrow\boldsymbol{ENU}}$, $\delta \boldsymbol{b}_{\boldsymbol{RPY}}$, $\delta \boldsymbol{v}_{\boldsymbol{ENU}}$, $\delta \boldsymbol{p}_{\boldsymbol{ECEF}}$ are taken directly as 4 variables (not 12 as in the EKF).

b) Equation searching for solution of $\delta \boldsymbol{q}_{\boldsymbol{RPY}\rightarrow\boldsymbol{ENU}}$, $\delta \boldsymbol{b}_{\boldsymbol{RPY}}$, $\delta \boldsymbol{v}_{\boldsymbol{ENU}}$, $\delta \boldsymbol{p}_{\boldsymbol{ECEF}}$ to approach measured values is solved directly. It does not matter if it is solved analytically, iteratively, using gradient or any other way. The goal is to find a proximate solution to reach measured values and distance between estimation and measurement.

c) The criteria to reach measured values can be/are defined for any vector independently and can by customized any way and also changed during run.

d) Almost all criteria of the filter are accessible and can be changed on run.

1) Equation solution

The filter is based on the same principles as presented in previous work in [1]. Firstly the bias drift $\delta \boldsymbol{b}_{\boldsymbol{RPY}}$ is excluded from equation 2 because it can be estimated from $\delta \boldsymbol{q}_{\boldsymbol{RPY}\rightarrow\boldsymbol{ENU}}$ directly:

$$\delta \boldsymbol{b}_{\boldsymbol{RPY}}(k+1)=\delta \boldsymbol{b}_{\boldsymbol{RPY}}(k)-\alpha_b(k)\delta \boldsymbol{q}_{\boldsymbol{RPY}\rightarrow\boldsymbol{ENU}}(k) \quad (8)$$

(α_b = correction size, see section *Correction Size Calculations*) and angular rate is corrected outside the error equation

$$\boldsymbol{\omega}_{\boldsymbol{RPY},\boldsymbol{compensated}}=\boldsymbol{\omega}_{\boldsymbol{RPY}}-\delta \boldsymbol{b}_{\boldsymbol{RPY}} \quad (9)$$

So the modified equation 2 now consists of $\delta \boldsymbol{q}_{\boldsymbol{RPY}\rightarrow\boldsymbol{ENU}}$, $\delta \boldsymbol{v}_{\boldsymbol{ENU}}$ and $\delta \boldsymbol{p}_{\boldsymbol{ECEF}}$ only. There are more ways how to find proximate solution of this modified equation. The solution presented here solves $\delta \boldsymbol{q}_{\boldsymbol{RPY}\rightarrow\boldsymbol{ENU}}$ and $\delta \boldsymbol{v}_{\boldsymbol{ENU}}$, $\delta \boldsymbol{p}_{\boldsymbol{ECEF}}$ separately (because it involves minimum of arithmetic operations).

a) $\delta \boldsymbol{q}_{\boldsymbol{RPY}\rightarrow\boldsymbol{ENU}}$ by quaternion math

Unlike in the [1], now the main vector is the measured magnetic field vector $\boldsymbol{m}_{\boldsymbol{RPY}}$ and measured velocity $\boldsymbol{v}_{\boldsymbol{ENU}}$ is the second vector. $\delta \boldsymbol{q}_{\boldsymbol{RPY}\rightarrow\boldsymbol{ENU}}$ is calculated (using superposition in each axis (Roll, Pitch and Yaw)) as

for $k=1$ to 3

$c_{rot,\pm}=\left(\boldsymbol{C}_{\boldsymbol{rot}}(\boldsymbol{\omega}_{\boldsymbol{search}}(k))|\tilde{\boldsymbol{m}}_{\boldsymbol{RPY}}|\right)\cdot|\boldsymbol{m}_{\boldsymbol{RPY}}|$

if $c_{rot,+}\geq c_{rot,-}$ then $s=1$

else $s=-1$

$\boldsymbol{\omega}_{\boldsymbol{corr}}[k]=s\left[acos(|\tilde{\boldsymbol{m}}_{\boldsymbol{RPY}}|\cdot|\boldsymbol{m}_{\boldsymbol{RPY}}|)-acos\left(max(c_{rot,+},c_{rot,-})\right)\right]$

end

$$\delta \boldsymbol{q}_{\boldsymbol{RPY}\rightarrow\boldsymbol{ENU}}=-\frac{1}{2}\alpha_{\delta q,m}|\boldsymbol{\omega}_{\boldsymbol{corr}}|$$

$$\boldsymbol{C}_{rot}\left(\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix}\right)=\begin{bmatrix}\cos(\beta)\cos(\gamma) & -\cos(\beta)\sin(\gamma) & \sin(\beta)\\ \sin(\alpha)\sin(\beta)\cos(\gamma)+\cos(\alpha)\sin(\gamma) & \cos(\alpha)\cos(\gamma)-\sin(\alpha)\sin(\beta)\sin(\gamma) & -\sin(\alpha)\cos(\beta)\\ \sin(\alpha)\sin(\gamma)-\cos(\alpha)\sin(\beta)\cos(\gamma) & \cos(\alpha)\sin(\beta)\sin(\gamma)+\sin(\alpha)\cos(\gamma) & \cos(\alpha)\cos(\beta)\end{bmatrix} \quad (10)$$

Where $\boldsymbol{C}_{\boldsymbol{rot}}$ is a rotation matrix [10]

$\boldsymbol{\omega}_{\boldsymbol{search}}$ is a search rotation vector – for separate rotations in R, P and Y axis respectively (for k =1, 2, 3)

$\boldsymbol{\omega}_{\boldsymbol{corr}}$ is a correction vector

$\alpha_{\delta q,m}$ is a magnetic vector correction size (see section *Correction Size Calculations*)

In case a new GPS measurement is available, $\delta \boldsymbol{q}_{\boldsymbol{RPY}\rightarrow\boldsymbol{ENU}}$ is updated using second pair of vectors (the velocity vector $\boldsymbol{v}_{\boldsymbol{ENU}}$ and the model velocity estimation $\tilde{\boldsymbol{v}}_{\boldsymbol{ENU}}$). These two vectors are not compared directly because there is a singularity when the vehicle is not moving (→ zero velocity vector). So the vectors are taken with added gravity and subtracted previous estimation

$$\begin{aligned}\tilde{\boldsymbol{v}}_{\boldsymbol{ENU},\boldsymbol{mod}}&=\tilde{\boldsymbol{v}}_{\boldsymbol{ENU}}(t)+\Delta t_{GPS}\cdot \boldsymbol{g}_{\boldsymbol{ENU}}-\tilde{\boldsymbol{v}}_{\boldsymbol{ENU}}(t_{previousGPS})\\ \boldsymbol{v}_{\boldsymbol{ENU},\boldsymbol{mod}}&=\boldsymbol{v}_{\boldsymbol{ENU}}(t)+\Delta t_{GPS}\cdot \boldsymbol{g}_{\boldsymbol{ENU}}-\tilde{\boldsymbol{v}}_{\boldsymbol{ENU}}(t_{previousGPS})\end{aligned} \quad (11)$$

The estimated (modified) velocity $\tilde{\boldsymbol{v}}_{ENU,mod}$ vector is rotated around measured magnetic field vector $\boldsymbol{m}_{RPY}$ by angle $\alpha_{\delta q,v}$ towards the measured (modified) velocity vector $\boldsymbol{v}_{ENU,mod}$. The sequence of calculations is:

$$
\begin{aligned}
&\delta \boldsymbol{q}_{ENU\to RPY} = \boldsymbol{C}\left(\tilde{\boldsymbol{q}}_{RPY\to ENU}\right)\delta \boldsymbol{q}_{RPY\to ENU}\\
&\boldsymbol{q}_{corr,+} = \left[\cos\left(\frac{\alpha_{\delta q,v}}{2}\right) \quad \sin\left(\frac{\alpha_{\delta q,v}}{2}\right)\left(\boldsymbol{C}\left(\boldsymbol{q}_{RPY\to ENU} * \delta \boldsymbol{q}_{RPY\to ENU}\right)|\boldsymbol{m}_{RPY}|\right)^T\right]^T\\
&\boldsymbol{q}_{corr,-} = \left[\cos\left(\frac{\alpha_{\delta q,v}}{2}\right) \quad -\sin\left(\frac{\alpha_{\delta q,v}}{2}\right)\left(\boldsymbol{C}\left(\boldsymbol{q}_{RPY\to ENU} * \delta \boldsymbol{q}_{RPY\to ENU}\right)|\boldsymbol{m}_{RPY}|\right)^T\right]^T\\
&\boldsymbol{v}_{rot,+} = \boldsymbol{C}\left(\boldsymbol{q}_{corr,+} * \delta \boldsymbol{q}_{ENU\to RPY}\right)|\tilde{\boldsymbol{v}}_{ENU,mod}|\\
&\boldsymbol{v}_{rot,-} = \boldsymbol{C}\left(\boldsymbol{q}_{corr,-} * \delta \boldsymbol{q}_{ENU\to RPY}\right)|\tilde{\boldsymbol{v}}_{ENU,mod}|\\
&\text{if } |\boldsymbol{v}_{ENU,mod}|\cdot\boldsymbol{v}_{rot,+} \geq |\boldsymbol{v}_{ENU,mod}|\cdot\boldsymbol{v}_{rot,-}\\
&\quad \text{then } \delta \boldsymbol{q}_{ENU} = \boldsymbol{q}_{corr,+} * \delta \boldsymbol{q}_{ENU\to RPY}\\
&\quad \text{else } \delta \boldsymbol{q}_{ENU} = \boldsymbol{q}_{corr,-} * \delta \boldsymbol{q}_{ENU\to RPY}\\
&\delta \boldsymbol{q}_{RPY\to ENU} = \left[\delta \boldsymbol{q}_{ENU}[1] \quad \boldsymbol{C}\left(\boldsymbol{q}_{RPY\to ENU}\right)^T \delta \boldsymbol{q}_{ENU}[2:4]\right]^T[2:4]
\end{aligned}
\tag{12}
$$

As mentioned before $\delta \boldsymbol{q}_{RPY\to ENU}$ is a vector part of quaternion and is used for small correction angels ($\rightarrow$ scalar part of quaternion ≈ 1). If the angles are huge (usually than ten degrees per step) transform $\delta \boldsymbol{q}_{RPY\to ENU}$ to quaternion and calculate with pure quaternions.

This method is used because it is very fast (less than 500 arithmetic operations per step) and is easy to implement.

b) $\delta \boldsymbol{q}_{RPY\to ENU}$ by gradient

Usage of a minimum-squared-error (MSE) criterion and gradient (or Hessian) calculations is another way to find $\boldsymbol{q}$ or $\delta \boldsymbol{q}$ for two sets of vectors. The approach how estimate quaternion from two sets of vectors using Gauss-Newton or Newton method was presented by Marins *et al* in [3] (article available on web). But this method is not as efficient as the previous because it involves several matrix calculations (including inversion).

c) $\delta \boldsymbol{v}_{ENU}$ and $\delta \boldsymbol{p}_{ECEF}$ calculations

Velocity and position correction direction and size calculation is much easier. It involves subtraction and distance calculation of two vectors only (no transformations are necessary).

$$
\begin{aligned}
\delta \boldsymbol{v}_{ENU} &= \alpha_v\left(\boldsymbol{v}_{ENU} - \tilde{\boldsymbol{v}}_{ENU}\right)\\
\delta \boldsymbol{p}_{ECEF} &= \alpha_p\left|\boldsymbol{p}_{ECEF} - \tilde{\boldsymbol{p}}_{ECEF}\right|
\end{aligned}
\tag{13}
$$

The formula shows that $\delta \boldsymbol{v}_{ENU}$ is calculated regarding distance between measurement and estimation while $\delta \boldsymbol{p}_{ECEF}$ not. This is because different correction calculations were used for each case.

2) Solution "distance" calculations

"Distance" calculations (estimation to measurement) serve to find a distance between estimated and measured values. They are much simpler than direction (or gradient) calculations and involve vector angle and vector distance calculations. General equations are as follows

$$
\begin{aligned}
d_{vec1} &= acos\left(\frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{\|\boldsymbol{v}_1\|\|\boldsymbol{v}_2\|}\right)\\
d_{vec2} &= \|\boldsymbol{v}_1 - \boldsymbol{v}_2\|
\end{aligned}
\tag{14}
$$

d_{vec1} represents vector angle calculations for the magnetic vector and modified velocity vector. d_{vec2} is used for velocity and position error distance calculations.

3) Correction Size Calculations

The first section of the filter gives only information which direction correct to (to the measurements) and about distance between estimation and measurement. So the aim of the second section is to find a correction size to keep estimation on INS sensors (accelerometer and gyrometer) in short term but also lock them to GPS and magnetometer measurements in long term. The function can be defined any way according to user specification. This part is under development but even in an early stage it provides good results (see *Results* section). In this case it was defined as

1. Bias drift $\delta \boldsymbol{b}_{RPY}$ accumulation. The $\alpha_b = 0.1$ except initialization time ($\alpha_b = 0$). This setting says that the bias accumulation is directly dependent on the attitude correction. But this simple accumulation algorithm works as proven in the [1].

2. $\delta \boldsymbol{q}_{RPY\to ENU}$. In this case situation is a bit complicated. $\delta \boldsymbol{q}_{RPY\to ENU}$ consists of two parts and it is the magnetic vector $\boldsymbol{m}$ and velocity vector $\boldsymbol{v}$. The are treated separately to calculate $\alpha_{\delta q,m}$ and $\alpha_{\delta q,v}$ to form final $\delta \boldsymbol{q}$. These gain coefficients are calculated according to the following general formulas

$$\alpha_{vec}(k+1) = \alpha_{vec}(k) - T_S\, d_{vec}(k)\,\beta_{fall} \tag{15}$$

or

$$\alpha_{vec}(k+1) = \alpha_{vec}(k) + T_S\, d_{vec}(k)\,\beta_{rise} \tag{16}$$

The equations show that actual gain depends on previous gain α_{vec} and actual measurement to estimation error d_{vec}. Decision whether to accumulate (equation 12) or de-accumulate (equation 13) gain depends on sensor noise, actual gain size, error between measurement and estimation and time. β_{rise} and β_{fall} are constants which determine speed of accumulation and de-accumulation (set respecting sensor drifts/offsets). The gain α_{vec} is limited by an upper border to prevent filter from divergence/oscillations (which may occur in EKF/UKF designs when coefficient are set improperly).

3. $\delta \boldsymbol{v}_{ENU}$. For most situations $\alpha_{v_{ENU}} = 0.5$ is sufficient enough. This converges to measurement for all cases.

4. $\delta \boldsymbol{p}$. The calculations are based on the same principles (equations 15-16) as $\delta \boldsymbol{q}_{RPY\to ENU}$ size calculations.

As can be seen from equations above it is very easy and simple to calculate correction sizes but the algorithm still is very efficient (as proven by simulations shown in chapter IV).

4) Higher Level of Control

Because the vector filter calculates correction direction and correction size there can be some superior control which can bypass lower levels of the filter and set corrections , this is sufficient for example during initialization when the filter settles its estimation within one or several consecutive steps (figure 5., *Higher Level of Control* sub-section). Or when the GPS looses fix for several seconds - in this case all the correction may by zeroed for that time or the INS can be switched to a loosely coupled version to keep at least attitude estimation valid.

D. Mixed Solution

Another possible filter design is to leave some parameters for the smaller EKF and the rest calculate by the vector filter. In this case $\delta \boldsymbol{q}_{RPY \rightarrow ENU}$ and $\delta \boldsymbol{v}_{ENU}$ are estimated by the 6 state MEKF and $\delta \boldsymbol{b}_{RPY}$ and $\delta \boldsymbol{p}_{ECEF}$ are estimated by the vector filter (using rules from above). This reduces filter complexity dramatically but still the MEKF is used for crucial parts of the INS.

IV. Results

In this chapter results and filter comparisons are presented. The comparisons are based on simulated data which are set to represent sensor errors during UAV flights as close as possible or with worse parameters for stress tests. Simulated data were chosen because in this case true values are known and many variations can be tested. The sensor errors were set to exceed measured errors of real sensors during flight conditions to test filter limits.

Three INS solutions are compared. The first one is a "full" MEKF – it means 12 state EKF, the second one is a combined solution with 6 state EKF estimating attitude and velocity error with the rest being estimated by a vector filter and as the last one is the full vector filter (without higher level of control).

A. Simulated Data

The flight itself was simulated by these parameters: UAV accelerations were simulated by a quarters of sinusoids (simulating acceleration of an vehicle with some momentum) with random amplitudes which did not exceed ±4 $m \cdot s^{-2}$ and random periods which did not exceed 10 seconds. Resulting speed and position were calculated by integrations. Vehicle rotation was simulated similar way with maximum angular rates of ±250 $deg \cdot s^{-1}$ and random periods with less then 10 seconds. In fact these conditions cannot be reached during a flight because the UAV has limits in freedoms of motion (cannot rotate upside down etc.) and length ans size of acceleration but the values are suitable to test filter limits.

B. Sensor Errors

Every sensor included additive white noise, bias drift, offset, mis-scaling and quantization error (simulating ADCs). Besides these errors, magnetometer included uncompensated hard and soft iron distortion and GPS included measurement delay. Here is a short list of (proximate) parameters.

Table I.
Sensor parameters (errors)

White noise for each axis (std. deviation)	offset/bias/other for each axis	Mis-scaling for each axis	quantization
accelerometer [$m \cdot s^{-2}$]			
< 0.3	< 0.25 , constant	< 1% , constant	0.01
Gyrometer [$deg \cdot s^{-1}$]			
2.7	< 1.5 , changing	< 2% , constant	0.15
Magnetometer (46μT vector size) [μT]			
0.2	< 0.5 , constant	< 10% , constant	0.2
GPS ENU velocity [$m \cdot s^{-1}$]			
0.3	0.25 sec measurement delay	-	0.01
GPS ECEF position [m]			
2.5	< 3 changing	-	0.01

As can be seen from previous data simulated sensors contain significant errors. For example overall error between real and measured magnetometer data may reach up to 4 degrees. Such a big error was set to simulate partially uncompensated iron distortions. With GPS velocity, 2D velocity error does exceed 0.7 $m \cdot s^{-1}$.

C. Simulation Results for Extreme Conditions

The sub-chapter presents results of simulations constructed with parameters mentioned in table I. In table there are presented the most important estimation INS outputs: gyro bias drift estimation, attitude estimation and velocity estimation. Position estimation is not considered (as being estimated from velocity). Estimation errors during filter initialization are excluded. The simulation is considered to be a "worst condition" scenario. It means that in a real flight INS should perform considerably better (the maximum error should be less than maximum error presented in table II). So the table has to be treated that way.

Table II. shows that the vector filter performs a little bit better in this case but it is caused by more proper selection of parameters compared to the EKF. The 12 state EKF does not perform better compared to the mixed 6 state EKF solution. So the table shows that the vector filter can replace the EKF without any problem. Large attitude estimation error is caused by uncompensated hard and soft iron distortions which are in this case equal to 4 deg (compare with table III. where the distortion error is roughly 1.5 deg).

For a short comparison, table III shows attitude estimation error for the 6 state EKF (mixed filter) and the vector filter for more common conditions. Maximum magnetometer error is set to 1.5 deg and the rest of errors are lowered to a half (except quantization errors and GPS velocity and position errors which were kept the same).

TABLE II.
INS ESTIMATION ERRORS FOR EXTREME CONDITIONS

Filter	average	standard deviation	maximum
Attitude estimation error [deg]			
12 state EKF	3.59	1.64	10.66
Mixed Filter	3.49	1.56	10.23
Vector Filter	3.21	1.33	9.81
Bias drift vector estimation error [deg/s]			
12 state EKF	0.49	0.2	1.4
Mixed Filter	0.41	0.16	1
Vector Filter	0.41	0.17	0.95
velocity vector estimation error [m/s]			
12 state EKF	0.38	0.16	1.16
Mixed Filter	0.37	0.16	1.15
Vector Filter	0.36	0.16	1.16

TABLE III.
INS ESTIMATION ERRORS FOR MORE COMMON CONDITIONS

Filter	average	standard deviation	maximum
Attitude estimation error [deg]			
Mixed Filter	1.53	0.98	5.86
Vector Filter	1.35	0.81	5.25

D. Simulation Results - Figures

Here are several examples of INS filter estimations. These figures were taken for common conditions simulation.

Figure shows that both two filters tend to overact a bit. This is caused by $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices set for extreme conditions for the EKF and rise β_{rise} and fall β_{fall} coefficients for the vector filter.

E. Simulation Results – Filter Complexity Calculations

As shown before both the MEKF and the vector filter can reach about the same estimation results and the MEKF can be replaced by the vector filter. The table below tries to show the main merit of the vector filter and this is its complexity (when it comes to number of arithmetic operations per filter step). The data are calculated for for filter frequency of 100 Hz and floating point operations are taken in single precision (one double precision FP operation was approximated as two single precision operations). Model calculations and ECEF to LLA transformations were omitted. Results were rounded.

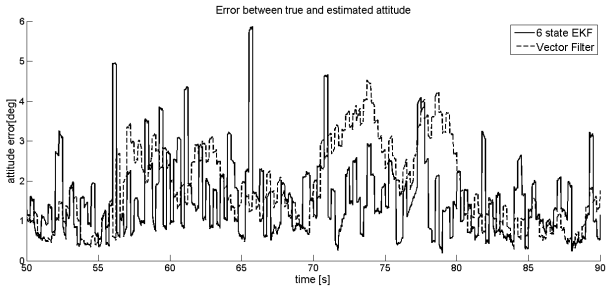

Fig 3. Example plot of attitude error for the mixed filter (6 state EKF) and the vector filter

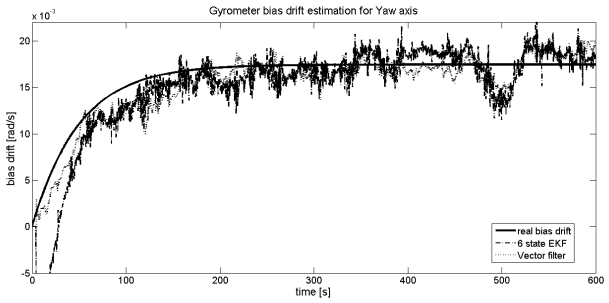

Fig 4. Example plot of a bias drift estimation

Matrix optimizations were calculated for the 12 state EKF only to show the huge difference between unoptimized and optimized calculations with sparse matrices. The table shows that the 12 state EKF needs really huge processing power and without optimizations and lowering filter frequency it is not suitable for miniature INS systems. The mixed solution and the vector filter are much better suited. The vector filter needs less than 4% of processing power compared to the (unoptimized) 12 state EKF and 20% of the processing power compared to the mixed solution.

F. Vector Filter - Higher Level of Control

The above simulations were done using the vector filter without higher level of parameter control – in this case the filter behaves like the corresponding EKF. As the vector filter calculates direction and distance of convergence to the measured values the higher level of control can take these data and for example during filter initialization it can estimate correct filter outputs in one or several consecutive steps.

V. CONCLUSION

The article shows that a standard tightly coupled GPS aided inertial navigation can be easily replaced by much more efficient solutions keeping the estimation as accurate as for

TABLE IV.
FILTER COMPLEXITY

Filter	FLOPS	FLOPS, matrix optimizations	FLOPS, full optimizations + 25 Hz filter frequency
12 state EKF	1896000	640000	190000
Mixed Filter	348000	-	-
Vector Filter	70000	-	-

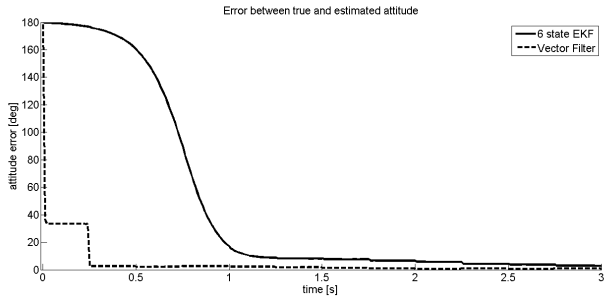

Fig 5. Example plot of a higher level control functionality (attitude estimation error). Compared vector filter and mixed solution.

the standard solutions. This may be done either by the hybrid solution where the KF is used to estimate some parameters while other are estimated by the custom filter. Or it may by done by the custom filter only. The main benefit of this approach is significant reduction in complexity (see table IV) and better system control-ability. Besides this benefit the nature of the vector filter implementation enables it to switch from tightly to loosely coupled system and back without any impact on the system which is vital in cases when GPS measurement is not present for longer period of time and INS attitude estimation would become corrupted due to present sensor errors.

These merits are vital especially in miniature embedded system where their price, size and power consumption is very limited (the system is being developed for the miniature UAV where the whole embedded system weights less than 100 grams and consumption is less than 1Watt (including wireless communication)).

References

[1] Floder J, Flying Object Control – Inertial Measurement Unit for an Embedded System, *PDES 2009*, pp. 108-113.

[2] Mohinder S. Grewal - Lawrence R. Weill - Angus P. Andrews, *Global Positioning Systems, Inertial Navigation and Integration*, ISBN 0-471-35023-X.

[3] Marins J. et al, An Extended Kalman Filter for Quaternion-Based Orientation Estimation Using MARG Sensors, *International Conference on Intelligent Robots and Systems,* 2001. *http://npsnet.org/~zyda/pubs/IROS2001.pdf*

[4] Bijker J., Steyn W., Kalman filter configurations for a low-cost loosely integrated inertial navigation system on an airship. *Elsevier Journal*, 2008. *http://linkinghub.elsevier.com/retrieve/pii/S0967066108000816*

[5] Markley F. L., Multiplicative vs. Additive Filtering for Spacecraft Attitude Determination, *NASA's Goddard Space Flight Center*.

[6] Xiaoying Kong., INS algorithm using quaternion model for low cost IMU, *Elsevier Journal,* 2004. *http:///linkinghub.elsevier.com/retrieve/pii/S0921889004000119*

[7] http://en.wikipedia.org/wiki/Geographic_coordinate_system

[8] Wendel J., Trommer G., Tightly coupled GPS/INS integration for missile applications, *Elsevier Journal*, 2004. *http://linkinghub.elsevier.com/retrieve/pii/S1270963804000793*

[9] Ko J., Klein D., Fox D., Haehnel D., GP-UKF: Unscented Kalman Filters with Gaussian Process Prediction and Observation Models *http://citeseerx.ist.psu.edu*

[10] Addison Wesley Publishing Company, The Official Guide to Learning OpenGL, Version 1.1, pp. 458-60