A Double-Stage Kalman Filter for Orientation Tracking With an Integrated Processor in 9-D IMU

Simone Sabatelli, Marco Galgani, Luca Fanucci, and Alessandro Rocchi

***Abstract*—This paper presents an application-specific integrated processor for an angular estimation system that works with 9-D inertial measurement units. The application-specific instruction-set processor (ASIP) was implemented on field-programmable gate array and interfaced with a gyro-plus-accelerometer 6-D sensor and with a magnetic compass. Output data were recorded on a personal computer and also used to perform a live demo. During system modeling and design, it was chosen to represent angular position data with a quaternion and to use an extended Kalman filter as sensor fusion algorithm. For this purpose, a novel two-stage filter was designed: The first stage uses accelerometer data, and the second one uses magnetic compass data for angular position correction. This allows flexibility, less computational requirements, and robustness to magnetic field anomalies. The final goal of this work is to realize an upgraded application-specified integrated circuit that controls the microelectromechanical systems (MEMS) sensor and integrates the ASIP. This will allow the MEMS sensor gyro plus accelerometer and the angular estimation system to be contained in a single package; this system might optionally work with an external magnetic compass.**

***Index Terms*—Angular position, application-specific instruction-set processor (ASIP), inertial measurement unit (IMU), Kalman filter, orientation tracking, quaternions, sensor fusion.**

I. Introduction

CURRENTLY, microelectromechanical systems (MEMS) sensors are widely used in a variety of consumer applications due to their low cost, small size, and low power consumption. In industrial research, there is widespread interest in additional features and improved precision from MEMS systems. In scientific literature, a lot of attention is paid to sensor fusion algorithms, because they allow raw data from multiple sensors to be processed to obtain additional information or improved precision.

A great amount of research is done for improving gyro accuracy for the use of inertial measurement units (from now on, IMUs) in high-precision applications, such as dead reckoning and pedestrian indoor navigation. These systems are based on IMU units that estimate the angular position. The expected gravity vector is subtracted from the measured accelerations, and the position of the IMU is calculated with a double integration. The main problem is that errors grow rapidly with the double integration, so that the system is not reliable for more than a few seconds without the aid of GPS signal or without speed or movement constraints. For example, in a navigation application, the movements of the sensor could be calculated within the constraints of a building map; in an odometer and foot counter, if the sensor is placed on a shoe, the accelerometer will measure a spike when the foot is landed and it can be assumed that, at this instant, the speed is zero, so that integration errors can be periodically corrected.

In a work dealing with an existing dead reckoning unit, [1] the analysis showed that the main error source in such systems is not the accelerometer noise but the incorrect gravity vector (because of inaccurate angle estimation) that is subtracted from measured accelerations. This work is thus focused on an improved angle estimation system. The main problem of this system is that the use of an integrator for gyro data leads to the sum of gyro errors during time, so that a correction factor must be introduced. In particular, it is known from theory that gyro angle random walk, the Gaussian white noise, introduces an error in angular estimation that grows with the square root of time, while bias instability and constant biases on gyro due to calibration errors or to thermal or soldering effects give an error that grows linearly with time.

A number of angular estimation systems have been previously published. Most of the papers use quaternions to represent the angular position, because they are more flexible than Euler angles to represent the angular position; moreover, with the use of quaternions, there is no problem of angular singularities, and the system can be better linearized. If needed, quaternions can be easily converted to other rotation representation methods, like a rotational matrix or a sequence of Euler angles. To eliminate the effects of drift in angular estimation due to gyro errors, all studies used some kind of sensor fusion algorithm. The Kalman filter is the most commonly used. In [2], an extended Kalman filter is used with three rate gyros and three accelerometers, and the state equation is composed of the quaternion and also of the angular velocity and the gyro drift. The proposed system has a good estimation of roll and pitch angles but only a low correction on yaw angle. In order to have a complete correction for the yaw angle, a magnetic compass is needed. Indeed, in [3] and [4], a complete magnetic, angular gate, and gravity sensor is used for angle estimation. To simplify the design of the Kalman filter, in these two works, an additional algorithm is proposed to estimate the quaternion that

Manuscript received March 27, 2012; revised May 4, 2012; accepted June 18, 2012. Date of publication September 28, 2012; date of current version February 5, 2013. The Associate Editor coordinating the review process for this paper was Dr. Deniz Gurkan.

S. Sabatelli and L. Fanucci are with the Department of Information Engineering, University of Pisa, 56126 Pisa, Italy (e-mail: simone.sabatelli@for.unipi.it).

M. Galgani is with the On-Board Payload Data Processing Section, European Space Research and Technology Centre, European Space Agency, 2200 AG Noordwijk, The Netherlands.

A. Rocchi is with the SensorDynamics AG, 56123 Pisa, Italy.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIM.2012.2218692

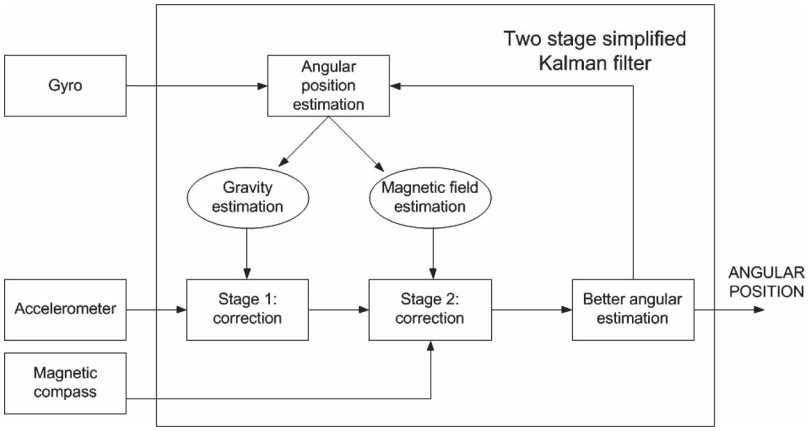

Fig. 1. Sensor fusion algorithm principle.

represents the angular position from accelerometer and magnetic compass data, so that a linear Kalman filter could be used. In particular, a Gauss–Newton method is proposed in [3], and a QUaternion ESTimate (QUEST) algorithm is proposed in [4]. An adaptive gain for the Kalman filter is used in [5] to adjust the angular correction when the sensor is in high acceleration mode or in a quiet state. Instead of the extended Kalman filter, [6] uses an unscented Kalman filter because this filter is deemed to be more accurate and less costly to implement. However, in [7], a comparison between the extended and the unscented filters is done, and the resulting precision is found comparable, although the unscented filter requires much more computational time.

All the cited previous works use discrete sensor systems, with separate sensors for gyro, accelerometer, and magnetic compass, and the filter algorithm is processed on a microcontroller or a personal computer (PC). A commercial example of this kind of angle estimation system can be found in [8], where the angular estimation is carried out with the use of three separate sensors and a microcontroller. In some models, it is also possible to interface this system with a GPS for a dead reckoning system.

In this paper, we focused on the realization of a complete system in a single package. For this reason, starting from the existing prototype of SensorDynamics SD746 6-D IMU, a system was developed that can use only gyro and accelerometer data to run the filter and may be optionally connected with an external magnetic compass. The two-stage filter offers greater operational flexibility, as it is possible to switch on or off the second stage to include or exclude the magnetic compass correction. Using accelerometer data, it is possible to correctly estimate roll and pitch angles only, while with the magnetic compass, the yaw angle is correctly estimated as well. This approach also offers simplification: The division of the system in two stages allows operating with smaller matrices, with less computing power required. This two-stage filter also allows the separation of the effects of a magnetic anomaly on roll and pitch angles' estimation precision, because, in this case, only the yaw angle estimation is affected [11]. Our final goal is to develop and integrate this angular estimation system within the application-specified integrated circuit (ASIC) that controls the gyro and accelerometer sensors; the external magnetic compass may be connected by an Inter-Integrated Circuit or two-wire interface (I^2C) bus. To realize this, we have an area constraint of 0.5 mm^2.

This paper is organized as follows. After this Introduction, in Section II, the mathematical theory used for the angular estimation system model is explained, Section III describes the design and the architecture of the application-specific instruction-set processor (ASIP), and finally, in Section IV, the field-programmable gate array (FPGA) prototype and test results are presented.

II. Theory of System Modeling

The system model was developed using MathWorks Simulink. This allows fast system development and testing both with artificial data sets and real acquired data.

The system has three different types of input data, as shown in Fig. 1: the gyro data measuring the angular rate in degrees per second on each of the three axes, the accelerometer data measuring acceleration in gram units, and the magnetic compass data measuring the magnetic field in gauss. It will be subsequently explained how the gyro data are used as input to update the angular position; meanwhile, the accelerometer and magnetic compass data are used as fixed references to correct the errors in the angular position estimation.

According to the Kalman filter theory (see [14]), it is necessary to define a discrete-time state equation that describes the evolution of the system state x_k, starting from the system state at the previous step x_{k-1} with the evolution law described by the matrix A_k and responding to some system inputs u_k with the evolution law B_k

$$\hat{x}_k^- = A_k \hat{x}_{k-1} + B_k u_k. \tag{1}$$

Equation (1) will give the so-called "*a priori*" system state estimation. For this reason, the system state x_k has the superscript "minus," while the hat represents the fact that the real system state is unknown and the Kalman filter is providing an estimation.

Only the quaternion representing the angular position as system state was used in the state equation. The addition of

other variables does not significantly increase the precision of the angular estimation, although it requires significantly greater processing power. The inclusion of extra variables in the system state leads to bigger matrix operations. Using them could possibly account for the gyro bias, but according to our simulations, the resulting increase of precision is less than 1°.

The angular position is represented using a quaternion q

$$q = [q_0, q_1, q_2, q_3]^{\mathrm{T}}. \tag{2}$$

A quaternion is composed of a real number and a vector: q_0 represents the real number, and $v = [q_1, q_2, q_3]^{\mathrm{T}}$ represents the vector.

A rotation is defined by a rotation axis and by the angle of rotation α around this axis. The vector part v represents the axis of rotation, while the real part represents the angle of rotation α according to

$$q = \cos\left(\frac{\alpha}{2}\right) + \sin\left(\frac{\alpha}{2}\right) * v. \tag{3}$$

In order to correctly represent a rotation, the quaternion must have unit norm.

The quaternion norm is calculated with

$$|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}. \tag{4}$$

Standard formulas were used to convert the angular rotation representation from the quaternion to the Euler angle form. The XYZ sequence of Euler angles was chosen, and the roll, pitch, and yaw angles are given by the following formulas:

$$\phi = \operatorname{atan2}\left(2(q_2q_3 + q_0q_1), \left(q_0^2 - q_1^2 - q_2^2 + q_3^2\right)\right) \tag{5}$$

$$\theta = \operatorname{asin}\left(\frac{2(q_0q_2 - q_1q_3)}{(q_0^2 + q_1^2 + q_2^2 + q_3^2)}\right) \tag{6}$$

$$\psi = \operatorname{atan2}\left(2(q_1q_2 + q_0q_3), \left(q_0^2 + q_1^2 - q_2^2 - q_3^2\right)\right). \tag{7}$$

The state equation in the time-continuous form is given by

$$\dot{q}_n^b = \frac{1}{2}\Omega_{nb}^n q_n^b \tag{8}$$

where q_n^b is the quaternion representing the rotation of the body frame, united to the IMU sensor, with respect to the inertial n-frame, and Ω_{nb}^n is the rotational matrix, derived from the quaternion's properties (see [12] and [13])

$$\Omega_{nb}^n = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}. \tag{9}$$

The angular velocities ω_x, ω_y, and ω_z measured by the gyro are used to fill the matrix (9). In this particular system, there is no explicit external input u_k with the evolution law B [see (1)]. The gyro data are implicit inputs because they are used to determine the matrix deriving the evolution of the system from the previous state. A time-varying $A_{\mathrm{TC}} = (1/2)\Omega_{nb}^n$ matrix was thus obtained. For the digital Kalman filter implementation, it is necessary to transform (8) from this time-continuous version to a time-discrete one. By making explicit the derivative operation, the continuous-time system can be written as

$$\dot{q}_n^b = \lim_{T\to 0} \frac{q_n^b(t+T) - q_n^b(t)}{T} = A_{\mathrm{TC}} q_n^b(t). \tag{10}$$

In order to transform this equation to a discrete-time one, the time step T between each execution of the algorithm in the digital system is used. In this manner, we obtain

$$q_n^b(t+T) = q_n^b(t) + A_{\mathrm{TC}} q_n^b(t)T = (I + A_{\mathrm{TC}}T) q_n^b(t). \tag{11}$$

The discrete-time matrix A_k has the form shown in

$$A_k = (I + A_{\mathrm{TC}}T) = \left(I + \frac{1}{2}\Omega_{nb}^n T\right). \tag{12}$$

The quaternion norm is not preserved in the state equation. As the angle position can be correctly represented only with unit quaternions, a normalization unit was introduced to ensure data integrity.

This first part, described earlier in (1), is the predictive, or "*a priori*," equation that updates the angular position using gyro data only. It is now necessary to define the correction equation of the Kalman filter. In our system, there are two correction equations, one for each stage: the first operating with accelerometer data and the second operating with magnetic compass data. Using the Kalman filter theory, the correction equation results as in

$$\hat{x}_k = \hat{x}_k^- + K_k\left(z_k - H\hat{x}_k^-\right) \tag{13}$$

where z_k represents the actual measurement, which, in the proposed system, is the accelerometer or the magnetic compass data. $H\hat{x}_k^-$ is the expected measurement that is calculated from the "*a priori*" system state. The difference between the real and the expected measurements is called the residual. The residual must be weighted with the K_k gain to obtain the correction factor and to calculate the "*a posteriori*" estimation of the system state $\hat{x}_k$, which is the least square solution to this probabilistic problem. In our system, it is not possible to use the linear relationship $H\hat{x}_k^-$ to calculate the estimated gravity or the magnetic field, but it is necessary to use a nonlinear relationship $h(\hat{x}_k^-)$ and the extended Kalman filter. This is because we have second-degree polynomials [see (17) and (21)].

The calculation of the Kalman gain K_k is a quite complex task, involving many matrix operations. First, it is necessary to calculate the "*a priori*" error covariance P_k^- that represents the error in the state estimation using the "*a priori*" state equation only

$$P_k^- = A_k P_{k-1} A_k^{\mathrm{T}} + Q_{k-1}. \tag{14}$$

In (14), the P_{k-1} matrix is the "*a posteriori*" error covariance matrix at the previous filter iteration [see (16)], while Q_{k-1} is the process noise covariance matrix. In this system, the Q_{k-1} covariance matrix directly depends on gyro noise and other gyro error sources, because the system evolution is described by the A_k matrix that contains the gyro data terms. However, the Q_{k-1} value does not represent the gyro noise but is related to the noise during the system evolution.

In the extended Kalman filter, the Kalman gain is calculated with

$$K_k = P_k^- H_k^{\mathrm{T}} \left(H_k P_k^- H_k^{\mathrm{T}} + V_k R_k V_k^{\mathrm{T}} \right)^{-1} \tag{15}$$

where R_k is the measurement noise covariance matrix, directly depending from the noise of the accelerometer and of the magnetic sensor, plus other error sources that are considered as noise. H_k and V_k are the Jacobian matrices of the partial derivatives with respect to the quaternion and to the noise of the nonlinear equations h_1 and h_2 [(17) and (21)]. These equations relate the quaternion to the estimated gravity and the estimated magnetic field. Because of the nonlinearity of these equations, the use of an extended Kalman filter is needed.

At the end, it is necessary to calculate the "*a posteriori*" error covariance matrix P_k to be used in the subsequent filter iteration

$$P_k = (I - K_k H_k) P_k^- \tag{16}$$

Now that the Kalman filter theory is explained, it is possible to see how it is applied to the system described in this paper. The first correction stage uses data from the accelerometers to correct the system state: The expected gravity vector is calculated from the estimated angular position and is subtracted from the value measured from the accelerometer, obtaining the so-called residual. The estimated gravity vector $\hat{g}$ is calculated using the direction cosine matrix R_n^b, assuming the g-force acceleration $|g|$ constant

$$h_1(q_k) = \hat{g} = R_n^b \begin{bmatrix} 0 \\ 0 \\ |g| \end{bmatrix} = |g| \begin{bmatrix} 2q_1q_3 - 2q_0q_2 \\ 2q_0q_1 + 2q_2q_3 \\ q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{17}$$

$$R_n^b = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2q_1q_2+2q_0q_3 & 2q_1q_3-2q_0q_2 \\ 2q_1q_2-2q_0q_3 & q_0^2-q_1^2+q_2^2-q_3^2 & 2q_2q_3+2q_0q_1 \\ 2q_1q_3+2q_0q_2 & 2q_2q_3-2q_0q_1 & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}. \tag{18}$$

The measured gravity suffers from significant errors because the accelerometers measure not only gravity but also the external accelerations if the system is in a dynamic state. For this reason, the residual must be weighted with the Kalman gain K_{k1}, a coefficient that is calculated from the statistics of the noise covariance matrices of the system. To calculate the Kalman gain, it is necessary to calculate the Jacobian matrix H_{k1}

$$H_{k1} = \frac{\partial h_{1[i]}}{\partial q_{[j]}} = \begin{bmatrix} -2q_2 & 2q_3 & -2q_0 & 2q_1 \\ 2q_1 & 2q_0 & 2q_3 & 2q_2 \\ 2q_0 & -2q_1 & -2q_2 & 2q_3 \end{bmatrix}. \tag{19}$$

The noise of the accelerometer (and also of the magnetic compass) is considered noncorrelated with the current angular position. Thus, the Jacobian matrix V_k is an identity matrix.

With the gravity vector measurement, it is possible to correct roll and pitch angles only. In order to ensure that the yaw angle is not affected by in applicable corrections, the third vectorial part of the correction quaternion $q_{\epsilon 1}$ is set equal to zero

$$q_{\epsilon 1} = q_{\epsilon 1,0} + q_{\epsilon 1,1} + q_{\epsilon 1,2} + 0 \cdot q_{\epsilon 1,3}. \tag{20}$$

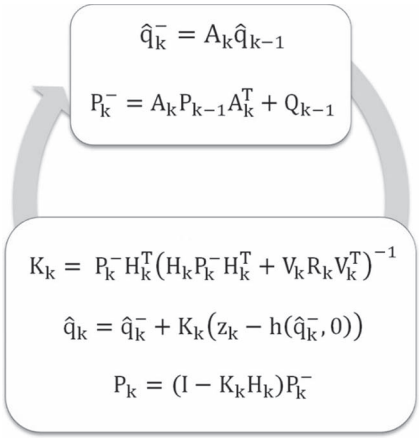

Fig. 2. Kalman filter algorithm.

The second stage of the filter is similar to the first one: A second Kalman gain K_{k2} is calculated with the same algorithm but using different noise covariance matrices. The magnetic field, normalized to one, is considered directed along the y-axis only, and the vertical component is neglected, as the Earth's magnetic field has a variable intensity from 0.25 to 0.65 G and the vertical component is geographically dependent. In the filter, the magnetic field is estimated with the following equation:

$$h_2(q_k) = \widehat{m} = R_n^b \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2q_1q_2 + 2q_0q_3 \\ q_0^2 - q_1^2 - q_2^2 - q_3^2 \\ 2q_2q_3 - 2q_0q_1 \end{bmatrix} \tag{21}$$

The Jacobian matrix H_{k2} is the following:

$$H_{k2} = \frac{\partial h_{2[i]}}{\partial q_{[j]}} = \begin{bmatrix} 2q_3 & 2q_2 & 2q_1 & 2q_0 \\ 2q_0 & -2q_1 & -2q_2 & -2q_3 \\ -2q_1 & -2q_0 & 2q_3 & 2q_2 \end{bmatrix}. \tag{22}$$

In this case, to ensure that magnetic anomalies do not influence roll and pitch estimation, but yaw angle only, the first two vectorial parts of the correction quaternion $q_{\epsilon 2}$ are set equal to zero

$$q_{\epsilon 2} = q_{\epsilon 2,0} + 0 \cdot q_{\epsilon 2,1} + 0 \cdot q_{\epsilon 2,2} + q_{\epsilon 2,3}. \tag{23}$$

The complete algorithm is summarized as follows (see also Fig. 2 for a schematic outline).

Start of "*a priori*" system estimation:

1) reading of the gyro sensor and obtaining angular velocities $\omega_x, \omega_y,$ and ω_z;
2) calculation of the discrete-time state transition matrix $A_k = I + (1/2)\Omega_{nb}^n T$;
3) calculation of the "*a priori*" system state estimation $\hat{q}_k^- = A_k \hat{q}_{k-1}$;
4) calculation of the "*a priori*" noise covariance matrix $P_k^- = A_k P_{k-1} A_k^{\mathrm{T}} + Q_k$.

Start of the correction stage 1:

1) calculation of the Jacobian matrix $H_{k1} = \begin{bmatrix} -2q_2 & 2q_3 & -2q_0 & 2q_1 \\ 2q_1 & 2q_0 & 2q_3 & 2q_2 \\ 2q_0 & -2q_1 & -2q_2 & 2q_3 \end{bmatrix}$;

2) calculation of the Kalman gain $K_{k1} = P_k^- H_{k1}^{\mathrm{T}}(H_{k1}P_k^- H_{k1}^{\mathrm{T}} + V_{k1}R_{k1}V_{k1}^{\mathrm{T}})^{-1}$;
3) reading of the accelerometer data $z_{k1} = a_x a_y, a_z$;
4) calculation of $h_1(\hat{q}_k^-) = \begin{bmatrix} 2q_1q_3 - 2q_0q_2 \\ 2q_0q_1 + 2q_2q_3 \\ q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$;
5) calculation of the correction factor $q_{\epsilon 1} = K_{k1}(z_{k1} - h_1(\hat{q}_k^-))$, $q_{\epsilon 1,3}$ is put equal to zero;
6) calculation of the "*a posteriori*" state estimation $\hat{q}_{k1} = \hat{q}_k^- + q_{\epsilon 1}$;
7) calculation of the "*a posteriori*" error covariance matrix $P_{k1} = (I - K_{k1}H_{k1})P_k^-$.

Start of the correction stage 2:

1) calculation of the Jacobian matrix $H_{k2} = \begin{bmatrix} 2q_3 & 2q_2 & 2q_1 & 2q_0 \\ 2q_0 & -2q_1 & -2q_2 & -2q_3 \\ -2q_1 & -2q_0 & 2q_3 & 2q_2 \end{bmatrix}$;
2) calculation of the Kalman gain $K_{k2} = P_k^- H_{k2}^{\mathrm{T}}(H_{k2}P_k^- H_{k2}^{\mathrm{T}} + V_{k2}R_{k2}V_{k2}^{\mathrm{T}})^{-1}$;
3) reading of the magnetic compass data $z_{k2} = m_x m_y, m_z$;
4) calculation of $h_2(\hat{q}_k^-) = \begin{bmatrix} 2q_1q_2 + 2q_0q_3 \\ q_0^2 - q_1^2 - q_2^2 - q_3^2 \\ 2q_2q_3 - 2q_0q_1 \end{bmatrix}$;
5) calculation of the correction factor $q_{\epsilon 2} = K_{k2}(z_{k2} - h_2(\hat{q}_k^-))$, $q_{\epsilon 2,1}$ and $q_{\epsilon 2,2}$ are put equal to zero;
6) calculation of the "*a posteriori*" state estimation $\hat{q}_k = \hat{q}_{k1} + q_{\epsilon 2}$;
7) calculation of the "*a posteriori*" error covariance matrix $P_k = (I - K_{k2}H_{k2})P_{k1}$.

III. Algorithm Simplification Towards Hardware Implementation

To create an integrated system with low area occupation and low power consumption, the Kalman filter algorithm presented in Section II was simplified. The objective was to process the filter on a simplified arithmetic logic unit (ALU) capable of only multiplication and sum operations.

The first simplification consists of approximating the "*a priori*" error covariance P_k^- with the "*a posteriori*" error covariance P_k. This is a logical simplification if the angular position estimated before the correction can be estimated with the angular position after the correction. This assumption is valid if the filter is iterated at high frequency, with very low corrections at each iteration.

To reduce the number of matrix multiplications, a precomputed value of P_k is used. Instead of calculating this parameter online from the noise statistics of the sensor, its value is assigned offline as a tuning parameter of the filter, assuming that similar sensors will have similar noise statistics. During normal filter operation, the initial value of P_k is greater because of the uncertain initial angular position. It then converges to lower values. The final value of P was calculated experimentally with different simulations on Matlab Simulink system model. To simulate the fact that the initial value of P is higher and to allow a fast initial position correction even if the system starts upside down, it was chosen to initialize the filter with a higher value of the P_k matrix and to scale down this value with a step function to the nominal value after the first 128 filter iterations. The same value of P_k is used for the two filter stages.

The following simplifications are important, because they allow eliminating all the division operations in the filter algorithm. As the matrix P_k is now a fixed parameter, it is possible to use a fixed value as the matrix determinant during matrix inversion. Moreover, it is possible to eliminate the normalization operation on the quaternion when the Kalman filter is working, because it operates as a self-normalization algorithm on the quaternion. This property was experimentally verified with many simulations on the simplified Simulink model of the system, but it does not apply, in general, for the full Kalman algorithm. With these simplifications, further matrix algebra manipulation, and a fine tuning of the operations computed from the hardware, it was possible to reduce the number of multiplications and additions by up to 60% with negligible performance degradation compared to the initial filter algorithm [9].

In this simplified system, the error covariance matrices Q, R_1, and R_2 are considered constant and have the following values:

$$Q = 10^{-6} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

$$R_1 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \tag{25}$$

$$R_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{26}$$

The error covariance matrix P_k has the value $P0$ at the filter start-up and for the first 128 iterations, while its value is decreased to $P1$ with a step response for normal filter operation

$$P0 = \begin{bmatrix} 0.125 & 0.0003 & 0.0003 & 0.0003 \\ 0.0003 & 0.125 & 0.0003 & 0.0003 \\ 0.0003 & 0.0003 & 0.125 & 0.0003 \\ 0.0003 & 0.0003 & 0.0003 & 0.125 \end{bmatrix} \tag{27}$$

$$P1 = \begin{bmatrix} 0.001 & 0.0002 & 0.0002 & 0.0002 \\ 0.0002 & 0.001 & 0.0002 & 0.0002 \\ 0.0002 & 0.0002 & 0.001 & 0.0002 \\ 0.0002 & 0.0002 & 0.0002 & 0.001 \end{bmatrix}. \tag{28}$$

IV. ASIP Design and Synthesis

Before starting with the digital design, it was necessary to define the number of bits necessary to represent the data and to derive a bit-true model in fixed-point arithmetic from the simplified model in floating-point arithmetic. The objective was to use 16-b arithmetic to match the input data from gyro and accelerometer; however, the use of a tailored number of bits for specified operations is not a problem in our custom design.

Data were acquired with a repeatable test and were processed offline with a Matlab Simulink model of the system. The model

TABLE I
RMS ERROR WITH DIFFERENT BIT NUMBERS USED IN THE FIXED-POINT ARITHMETIC MODEL

n bit	RMS error (deg)
24	0,19
22	0,20
20	0,26
18	0,53
16	8,83

allows simulating the bit-true arithmetic that will be used in the integrated processor, to be implemented in FPGA or CMOS standard-cell technologies. The Simulink model allows a full simulation of the behavior of the system and fine tuning of key parameters of the processor architecture. In this case, it was used to simulate the effects of the fixed-point arithmetic using different bit numbers to represent the internal data. Using the data acquired during the test, the RMS error was calculated for the roll angle. The results are reported in Table I for the fixed-point arithmetic. For comparison, the floating-point arithmetic model leads to an RMS error of 0.17°.

As shown in Table I, the use of 16 b for the fixed-point arithmetic leads to poor precision in the angular estimation. While the best result is obtained with the use of the greater number of bits, it is useful to evaluate the best value in terms of the ratio precision/cost. The cost of implementing a processor with a processing unit is a function of n^2, where n is the number of bits needed. The best tradeoff between precision and cost is obtained using 20 b for the fixed-point arithmetic. To further optimize the system, a different number of bits was used to represent different kinds of data during the simulations. The data were divided into groups, each with a parameterized number of bits of precision: the quaternion, the state update matrix A, the K matrix, the estimated gravity and magnetic field, and the other internal variables of the simplified Kalman filter. Simulations showed that the critical data for the precision were the quaternion and the A matrix. Reducing the number of bits used in the fixed-point arithmetic to 16 did not cause a strong increase in RMS error. Using 20 b to represent the quaternion and state update matrix A data and 16 b to represent all other data resulted in an RMS error of 0.27 compared to 0.26 resulting from using all data as 20 b.

The second issue to be considered is the architecture to implement. A fully parallel implementation was considered but was deemed unsuitable because of its area occupation. It was preferred to have a flexible system, so that the current design could be easily updated for future system expansions or for algorithm improvements. An ASIP architecture with a general ALU and reduced instruction set computer (RISC) architecture was chosen. It was possible to implement an RISC architecture because the number of instructions to be executed is much lower than the available number of clock cycles: The estimated ASIP frequency is 1 MHz, and a suitable filter update rate is 1 kHz.

A mixed approach between the hardwired and the programmed solution was chosen for the control unit, defining a custom instruction set architecture and storing the instructions

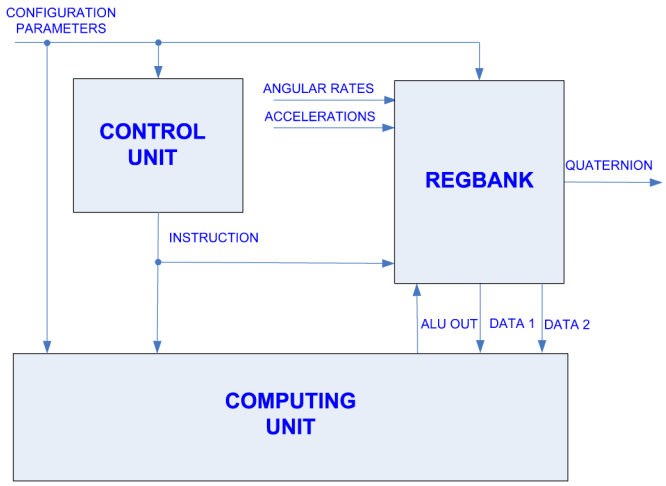

Fig. 3. Integrated processor architecture.

to be executed on a ROM. In the future, the system could be updated just by changing the ROM.

The system was divided into three main blocks: the control unit, the computation unit with the ALU, and the memory composed of the regbank (see Fig. 3). During normal operation, the input data from the IMU sensor, the gyro rate, and the accelerations are acquired in the regbank; the filter algorithm is then performed, and the output quaternion is stored in the regbank.

The computation unit is capable of sum, multiplication, and multiplier and accumulator operations on 20-b input data. Each instruction of the computation unit is performed in two steps: the fetch-and-decode operation and the execute-and-write operation. The regbank is made up of two register groups. The computation unit reads one register from each group. The output data of the computation unit are on 20 b. For the noncritical data, a truncation unit allows the data to be stored on a 16-b register with consequent area and power savings.

The control unit allows setting a specific filter frequency and sets a low-power state for the system when the elaboration is finished (i.e., before starting a new filter iteration). Most of the time, the integrated processor is in the low-power state, so that clock gating reduces both the switching activity and the power consumption. The ROM containing the instructions is read sequentially, but it is also possible to set some jump conditions for future upgrades.

The design was carried out using VHSIC hardware description language (VHDL) and Cadence NCSim tool for simulation and synthesis. The synthesis was done with BuildGates on Taiwan Semiconductor Manufacturing Company (TSMC) 0.18-μm CMOS standard-cell technology, which is currently used for mixed analog and digital ASIC design in SensorDynamics.

After the synthesis with TSMC 0.18-μm technology, the ASIP occupied an area of 0.36 mm^2, which is less than the initial constraint of 0.5 mm^2. The estimated current consumption is 75 μA in active mode and 2 μA of leakage current when inactive.

During the design, both Simulink and VHDL models were fed with the same random input vectors and real acquired data. The output data verified a perfect match.

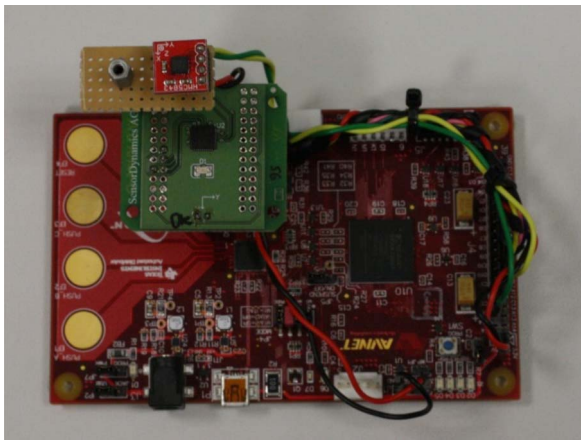

Fig. 4. FPGA board with SensorDynamics SD746 gyro plus accelerometer and Honeywell HMC5843 magnetic compass.

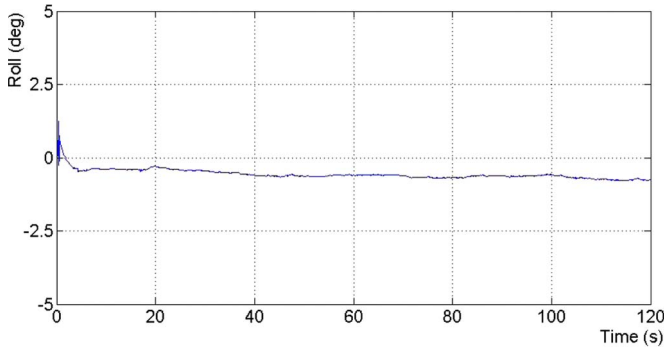

Fig. 5. Roll angle position estimation with still sensor.

V. Results: FPGA Prototyping and Testing

To verify the correct functionality in a practical scenario, the system was synthesized on an FPGA Xilinx Spartan 3A (see Fig. 4) using the Avnet design kit. A serial-to-parallel interface (SPI) was developed to connect directly the FPGA with SensorDynamics SD746 sensor, a three-axis gyro plus three-axis accelerometer integrated sensor. The Honeywell HMC5843 magnetic compass connected with I^2C bus. Gyro output data were available with an update rate of 10 kHz, but the filter was set to run at a frequency of 1 kHz. The FPGA was also connected via universal asynchronous receiver/transmitter (UART) to a PC, so that the output data formed by the quaternion could be acquired and processed. To allow a first system verification, a simple 3-D graphical animation was developed using the VPython library, so that the position of the sensor could be shown on a PC with a live demo.

The system was tested in static position to ensure that the filter was able to contrast the drift due to the gyro bias in angle estimation. As shown in Figs. 5–7, the static error results in less than 1° for roll and pitch angles, and less than 3° for the yaw angle.

The system was also tested to ensure that the initial position is correctly estimated even if the sensor starts upside down (see Figs. 8–10).

Finally, the system was tested with an RMS test machine that allows precise and repeatable rotations of the board. The magnetic compass was not used in this test.

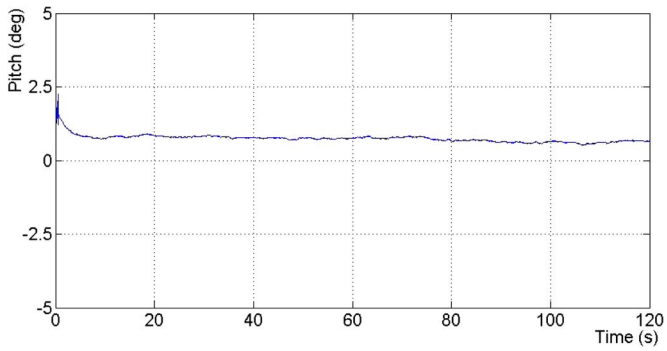

Fig. 6. Pitch angle position estimation with still sensor.

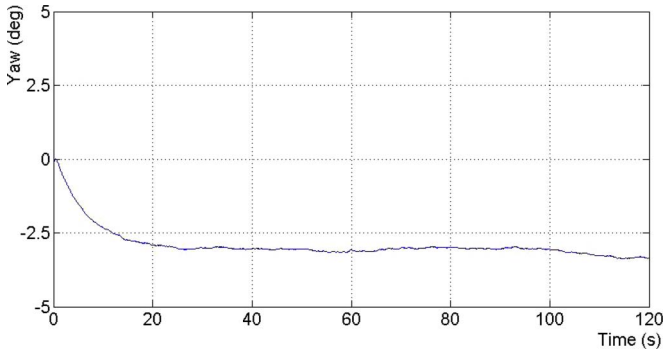

Fig. 7. Yaw angle position estimation with still sensor.

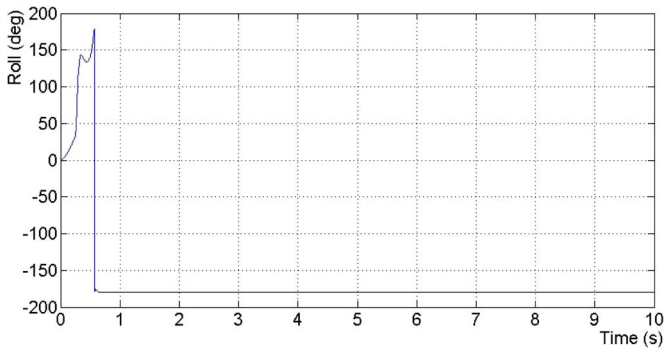

Fig. 8. Upside-down start-up for roll angle.

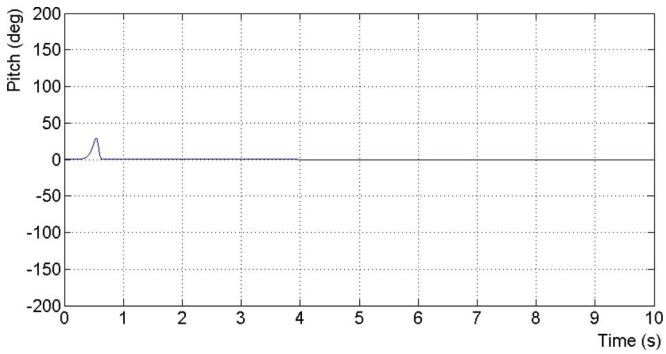

Fig. 9. Upside-down start-up for pitch angle.

For roll and pitch axes, the absolute error in static position is less than 1° across the entire acquisition time (see Figs. 11 and 12).

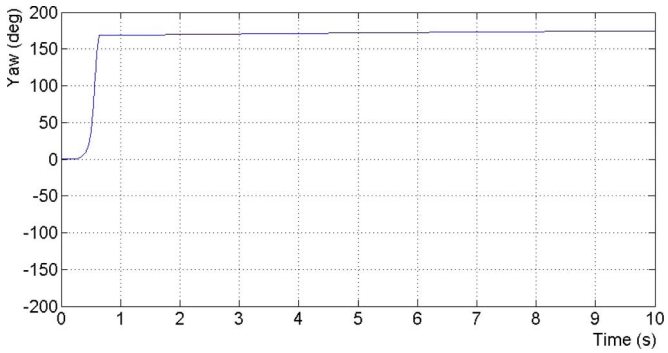

Fig. 10. Upside-down start-up for yaw angle.

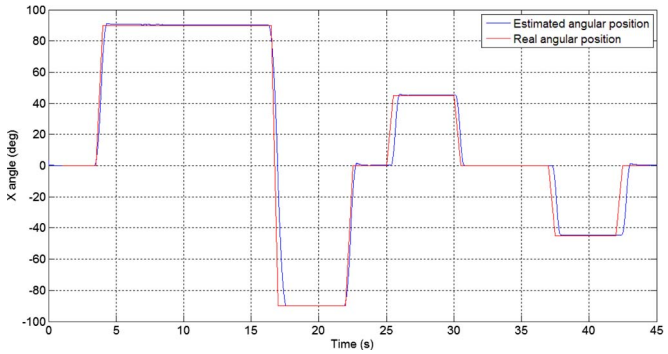

Fig. 11. Roll angle estimation.

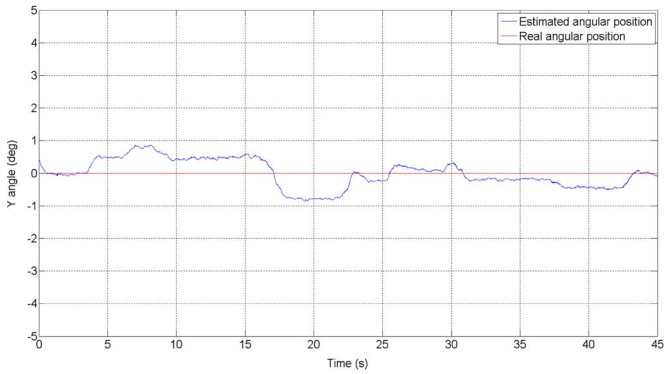

Fig. 12. Pitch angle estimation.

VI. Conclusion

During this work, a sensor fusion algorithm based on a novel two-stage Kalman filter has been developed. The two-stage filter has many advantages in terms of flexibility, computing requirements, system simplification, and also for robustness to magnetic anomalies. An ASIP to process the algorithm was specifically designed to be integrated within a 6-D sensor logic. The estimation system is flexible and can also process data from an external magnetic compass.

The filter was developed using MathWorks Simulink, several algorithm simplifications were tested and implemented, and a proof of concept was created on a FPGA for full testing. The ASIP was designed paying special attention to area and power consumption tradeoffs.

The FPGA was interfaced via SPI with SD746 gyro-plus-accelerometer sensor and with Honeywell magnetic compass and connected to a PC, so that it was possible to run a live demo and to record data. Tests showed errors of less than 1° for roll and pitch angles and less than 3° for yaw angle.

The future goal is to integrate the processing unit within the mixed analog/digital ASIC that controls the 6-D sensor, with the option to interface with an external magnetic compass.

References

[1] O. J. Woodman, "An introduction to inertial navigation," Univ. Cambridge, Cambridge, U.K., Tech. Rep., 2007.
[2] A. Kim and M. F. Golnaraghi, "A quaternion-based orientation estimation algorithm using an inertial measurement unit," in *Proc. Int. Conf. Intell. Robots Syst.*, May 2001, pp. 268–272.
[3] X. Yun, M. Lizarraga, E. R. Bachmann, and R. B. McGhee, "An improved quaternion-based Kalman filter for real-time tracking of rigid body orientation," in *Proc. Int. Conf. Intell. Robots Syst.*, Dec. 2003, pp. 1074–1079.
[4] X. Yun and E. R. Bachmann, "Design, implementation, and experimental results of a quaternion-based Kalman filter for human body motion tracking," *IEEE Trans. Robot.*, vol. 22, no. 6, pp. 1216–1227, Dec. 2006.
[5] M. Wang, Y. Yang, R. R. Hatch, and Y. Zhang, "Adaptive filter for a miniature MEMS based attitude and heading reference system," in *Proc. Position Location Navig. Symp.*, Jul. 2004, pp. 193–200.
[6] E. Kraft, "A quaternion-based unscented Kalman filter for orientation tracking," in *Proc. 6th Int. Conf. Inf. Fusion*, 2003, pp. 47–54.
[7] J. J. LaViola, Jr, "A comparison of unscented and extended Kalman filtering for estimating quaternion motion," in *Proc. Amer. Control Conf.*, Jun. 2003, pp. 2435–2440.
[8] XSense MTx Product Datasheet. [Online]. Available: http://www.xsens.com/en/general/mtx
[9] S. Sabatelli, F. Sechi, A. Rocchi, and L. Fanucci, "A sensor fusion algorithm for an integrated angular position estimation with inertial measurement units," in *Proc. DATE*, 2011, pp. 1–4.
[10] S. Sabatelli, M. Galgani, A. Rocchi, and L. Fanucci, "An application specific instruction set processor for angular position estimation with inertial measurement units," in *Proc. IEEE Sens. Conf.*, Oct. 2011, pp. 886–888.
[11] S. Sabatelli, M. Galgani, A. Rocchi, and L. Fanucci, "A double stage Kalman filter for sensor fusion and orientation tracking in 9D IMU," in *Proc. IEEE SAS Conf.*, Feb. 2012, pp. 1–5.
[12] L. Vicci, "Quaternions and rotations in 3-space: The algebra and its geometric interpretation," Microelectronic Syst. Lab., Dept. Compu. Sci., Univ. North Carolina, Chapel Hill, NC, 2001.
[13] J. B. Kuipers, *Quaternions and Rotations Sequences: A Primer With Applications to Orbits, Aerospace and Virtual Reality*. Princeton, NJ: Princeton Univ. Press, 1999.
[14] G. Welch and G. Bishop, "An introduction to the Kalman filter," Dept. Comput. Sci., Univ. North Carolina, Chapel Hill, NJ, 2001.
[15] M. Keating and P. Bricaud, *Reuse Methodoogy Manual for System on a Chip Design*, 3rd ed. New York: Springer-Verlag, 2002.
[16] R. R. Eckert, *Micro-Programmed Versus Hardwired Control Units: How a Computer Really Work*, Dept. Comput. Sci., Binghmanton, NY, 1988.
[17] N. S. Matlo, *Introduction of a Microcode Implementation of a CPU Architecure*, 1997.

Simone Sabatelli received the M.S. degree in electronic engineering from the University of Pisa, Pisa, Italy, in 2010, where he is currently working toward the Ph.D. degree in information engineering.

Since 2010, he has been collaborating with SensorDynamics AG, Graz, Austria, on the design, development, and testing of sensor fusion algorithms for 6-D inertial measurement units.

Marco Galgani received the M.S. degree in electronic engineering from the University of Pisa, Pisa, Italy, in 2011.

After the degree, he collaborated with SensorDynamics AG, Graz, Austria, as a Digital Designer for field-programmable gate array prototyping and testing to allow the integration of sensor fusion algorithms. He is currently with the On-Board Payload Data Processing Section, European Space Research and Technology Centre, European Space Agency, Noordwijk, The Netherlands, developing a new test system for avionics applications.

Luca Fanucci received the Laurea and Ph.D. degrees in electronic engineering from the University of Pisa, Pisa, Italy, in 1992 and 1996, respectively.

From 1992 to 1996, he was with the European Space Research and Technology Centre, European Space Agency, Noordwijk, The Netherlands, as a Research Fellow. From 1996 to 2004, he was a Senior Researcher with the Italian National Research Council, Pisa. He is a Professor of microelectronic systems with the Faculty of Engineering, University of Pisa. His research interests include several aspects of design technologies for integrated circuits and electronic systems, with particular emphasis on system-level design, hardware/software codesign, and sensor conditioning and data fusion. The main application areas are in the fields of wireless communications, low-power multimedia, automotive, healthcare, ambient assisted living, and technical aids for independent living. He is the coauthor of more than 300 journal and conference papers and coinventor of more than 30 patents. He is a Member of the Editorial Board of Elsevier *Microprocessors and Microsystems* Journal and IOS Press *Technology and Disability* Journal.

Dr. Fanucci served in several technical program committees of international conferences. He was the Program Chair of Euromicro Conference on Digital System Design (DSD) 2008, the Application Track Chair at Design, Automation & Test in Europe (DATE) from 2006 to 2010, and the Tutorial Chair at DATE in 2011 and 2012. He is currently serving as Vice Program Chair at DATE 2013.

Alessandro Rocchi received the B.E. degree in telecommunication engineering from the University of Pisa, Pisa, Italy, in 1995.

He was with the R&D, STMicroelectronics, and was a Senior Designer with AustriaMicroSystems. He is currently the R&D Director of Inertial and Micro Sensor Systems Business Unit (BU), SensorDynamics AG, Pisa.