An Object-Oriented Asynchronous Kalman Filter with Outlier Rejection for Autonomous Tractor Navigation
Yiannis Ampatzidis\textsuperscript{1)}, Stavros Vougioukas\textsuperscript{1)}, Simon Blackmore\textsuperscript{2)}, Dionisis Bochtis\textsuperscript{1)}

\textsuperscript{1)} Aristotle University of Thessaloniki, Department of Agricultural Engineering, 54124 Thessaloniki, Greece. E-mail: iampatzi@agro.auth.gr
\textsuperscript{2)} The Royal Veterinary and Agricultural University (KVL), Department of AgroTechnology, Denmark. E-mail: Simon@unibots.com

Introduction
In the near future, precision farming operations will increasingly rely on the automatic steering and navigation capabilities of agricultural vehicles. Auto-steering reliability and accuracy depend on the continuous availability of valid and accurate state data, provided at a frequency at least equal to the sampling rate of the tracking controller. The data provided by any single sensor (e.g., GPS) is inadequate for long term autonomous steering. The reasons are that occasional signal outage and exceedingly large noise content; may last for time periods which are very long in comparison to the controller’s sampling rate. The Kalman filter is a well-established technique to combine data from different sensors, in order to improve the availability and precision of the overall localization system. The actual trend is to use sensors of different nature and to properly weigh the relative data according to their reliability. For this purpose Kalman filtering techniques are used for the estimation of robots localization (Jetto et al., 1999a; Jetto et al., 1999b; Roumeliotis et al., 2000; Sasiadek et al., 1999; Pathirana et al., 2005). Last years this filter has been used in modern precision farming applications to achieve an accurate and unfailing estimation of vehicle positioning (Hague et al., 2000; Han et al., 2002; Noguchi et al., 1998a).

Typical filter implementations are platform specific, and do not address in a systematic way the issues of different sensor sampling rates and data validity. In this paper the implementation of an object-oriented, asynchronous, outlier-rejecting, extended Kalman filter is presented, along with simulation and experimental results, which validate the correctness and performance.

The Extended Kalman Filter Equations
The Kalman filter is a set of mathematical equations that provides an efficient computational solution for state estimation of dynamic systems, based on a recursive least-squares method (Kalman, 1960). The use of Kalman filtering requires the derivation of a stochastic state-space representation of the system model and of the state measuring process. For a slow moving tractor, this can be readily performed by using the vehicle’s kinematical model and
appropriate measuring models for its sensors. The extended Kalman filter (EKF) proposed in
the literature (Marins et al., 2001; Welch and Bishop, 1995) is an extension of the standard
filter for nonlinear system and sensor models and it is also based on the same input and
measurement noise covariance matrices. In this paper an Object-Oriented Asynchronous
EKF is implemented. The complete set of EKF equations are shown below (Kelly, 1994;
Ampatzidis, 2005).

\[
\begin{align*}
\text{system model} & \quad \overline{X} = f(\overline{x},t) + g(t) \cdot \overline{w}(t) \quad (1) \\
\text{measurement model} & \quad \overline{Z} = h(\overline{x},t) + \overline{v}(t) \quad (2) \\
\text{measurement Jacobian} & \quad H_k = \frac{\partial h}{\partial X}(\overline{X}_k) \quad (3) \\
\text{system Jacobian} & \quad F_k = \frac{\partial f}{\partial X}(\overline{X}_k) \quad (4) \\
\text{Kalman gain} & \quad K_k = P_k^{-} \cdot H_k^T \cdot (H_k \cdot P_k^{-} \cdot H_k^T + R_k)^{-1} \quad (5) \\
\text{state-estimate update with measurement} & \quad \overline{X}_k = \overline{X}_k + K_k \cdot [\overline{Z}_k - h(\overline{X}_k)] \quad (6) \\
\text{error covariance } P_k \text{ update} & \quad P_k^{-} = (I - K_k \cdot H_k) \cdot P_k^{-} \quad (7) \\
\text{state and covariance projection} & \quad \overline{X}_{k+1} = \Phi_k \cdot \overline{X}_k \quad (8) \\
& \quad P_{k+1}^{-} = F_k \cdot P_k^{-} \cdot F_k^T + \Gamma_k \cdot Q_k \cdot \Gamma_k^T \quad (9)
\end{align*}
\]

The Object-Oriented EKF Implementation
The main building block of the object-oriented Kalman filter is the Sensor class, which
abstracts and encapsulates all the functionality of a “smart” sensor via virtual functions. The
class data contains information about a sensor’s refresh rate, data availability, position in the
vehicle frame, and sensor model (noise covariance matrix and Jacobian sensor state
measurement). The class’s virtual functions prescribe the operations of connecting to the
sensor hardware, getting a reading, predicting a reading, updating the measurement
Jacobian, updating the sensor noise matrix, the Kalman gain matrix, the state and the state
uncertainty, i.e., its covariance matrix (equations 3-7). This class is a parent class for
different sensor types, e.g., GPS, compass, gyro, odometers, Doppler radar, etc. In order to
incorporate a sensor in the filter, only the virtual functions need to be overloaded. So, each
robot’s sensor individually is described by a class (e.g. GPS_sensor class) which is based in
the parent Sensor class and inherits all the Sensor’s virtual functions (figure 1). Hence, the
filter is extensible and can use sensors from different companies and with different
communication protocols. The *KalmanFilter* class contains as data the vehicle’s kinematics homogeneous matrix, its state transition matrix, its Jacobian matrix, the total state uncertainty, control uncertainty matrices, and the controller’s sampling rate. The class also contains functions which implement all the extended Kalman filter equations.

The Kalman filter object executes in the following way: at every sampling interval all state-dependent matrices are computed, and the next vehicle state and its uncertainty are predicted. Next, each sensor is polled and if new data is available, it is used to update the state estimate and its uncertainty, along with all relevant filter matrices, according to the filter equations. Hence, the filter is asynchronous and accommodates sensors with different sampling rates (figure 2). The filter was implemented in ANSI C++, so that the code can be used in any embedded navigation microcontroller.

**Outlier Rejection**
In the Kalman filter each sensor measurement contributes to the vehicle’s estimated state with a weight-factor that depends on a sensor uncertainty model, which provides the
variance of the noise content of the measurement itself. Sensor models are typically known a priori; some sensors may provide updates of their model dynamically. In practice, sensor data may deviate significantly from the assumed sensor model. Such data are called outliers and cause large errors in the state estimation. In the proposed filter implementation, if one or more state components resulting from a sensor reading deviate more than a preset multiple of the corresponding deviation of the state components predicted by the filter, then the reading is discarded, i.e. the sensor’s availability flag is lowered. It is known that the difference \[ \overline{Z_k} - h(\overline{X_k}) \] in (6), which is called the measurement innovation, or the residual, reflects the discrepancy between the actual measurement \( \overline{Z_k} \) and the predicted measurement \( h(\overline{X_k}) \). So, in the filter implementation and before equation (6) is used, the deviation between the actual sensor measurements \( \overline{Z_k} \) and the predicted measurements \( h(\overline{X_k}) \) is computed. If a sensor’s measurement contains large error, the residual will be very large and the filter will discard this measurement. Hence, at each sample interval only valid sensor readings are used by the filter.

**Experimental Results**
The filter was used in a number of simulations. The sensor measurements were the vehicle’s global position (UTM x, y, z), the roll, pitch and yaw angles and the linear and rotational velocities. A PID tracking controller kept the vehicle’s motion on the pre-planned path. Figure 1 shows a simulation of autonomous vehicle motion in an orchard. The simulated vehicle is equipped with a very accurate RTK-GPS (2cm precision). For each one of the other sensors the variance is set to 10%. In parts of the trajectory high-amplitude non Gaussian random noise was added in the GPS measurements to simulate signal outliers due to the trees and nearby building (figure 3). The scenario was executed twice. First, only the GPS was used to calculate the vehicle position (blue dotted line), and next the Kalman filter was used (red line). The sensor’s noise was the same in both cases. The black dotted line shows the desired path. When only the GPS was used, the vehicle deviated significantly from the desired path. Close to the building, between points 1-2 (figure 3, right), the precision of the GPS measurements was reduced, non Gaussian random noise was added, and the vehicle did not follow the predetermined path, but on the contrary it moved near the trees.
When the Kalman filter was used (red line) to achieve an accurate estimation of vehicle position, the vehicle continued to follow the trajectory, even when it passed near the building. This happened because when the GPS’s accuracy was lowered, the filter automatically discarded the reading. Hence, at each sample interval only valid sensor readings are used by the filter. The standard deviation and maximum absolute value of the errors between the actual and planned trajectories were estimated twice, first using only GPS ($\sigma_{\text{GPS}}=1.01$, $e_{\text{max}}=3.02$ m) and then with the Kalman filter ($\sigma_{\text{KF}}=0.02$, $e_{\text{max}}=0.19$ m).

The filter was also used on the iRobot™ research platform at KVL, using an RTK-GPS, an INS, a digital compass and an odometer. Figure 4 shows a simple experiment, where the iRobot was commanded to move with constant linear and angular velocities ($V=0.6$ m/s and $V_{\text{rot}}=0.2$ m/s), so that it moved on a circle of diameter $\rho=6$ meters. The Kalman filter and sensor measurements were recorded. The blue dotted line shows the GPS measurements and the red line shows the Kalman filter measurements.

Conclusions
This paper presented an object-oriented, asynchronous, outlier-rejecting, EKF suitable for any embedded navigation microcontroller. The EKF implementation was modified so that if a sensor’s measurement contains large error, the filter will discard this measurement. Hence, at each sample interval only valid sensor readings are used by the filter. Simulations and experiments showed that both the positioning precision and reliability improved significantly.

References