Event-Based Localization in Ackermann Steering Limited Resource Mobile Robots

Leonardo Marín, *Member, IEEE*, Marina Vallés, Ángel Soriano, Ángel Valera, *Member, IEEE*, and Pedro Albertos, *Senior Member, IEEE*

***Abstract*—This paper presents a local sensor fusion technique with an event-based global position correction to improve the localization of a mobile robot with limited computational resources. The proposed algorithms use a modified Kalman filter and a new local dynamic model of an Ackermann steering mobile robot. It has a similar performance but faster execution when compared to more complex fusion schemes, allowing its implementation inside the robot. As a global sensor, an event-based position correction is implemented using the Kalman filter error covariance and the position measurement obtained from a zenithal camera. The solution is tested during a long walk with different trajectories using a LEGO Mindstorm NXT robot.**

***Index Terms*—Dynamic model, embedded systems, event-based systems, global positioning systems (GPSs), inertial sensors, Kalman filtering, mobile robots, pose estimation, position measurement, robot sensing systems, sensor fusion.**

I. Introduction

LOCALIZATION is a fundamental issue in autonomous mobile robots. Without the knowledge of its exact position and heading (*pose*), the robot would not be able to navigate in the environment, follow a path, go to a goal point, or return to its starting point. This is a challenging issue since the robot position information is obtained from sensors subject to noise and nonlinearities. If this fact is not taken into account, it could lead to a big uncertainty in the positions measure. This problem has been widely studied in the literature [1], [2] in multiple ways. First, assuming that the initial position prior to the movement is known, the relative robot *pose* can be estimated by using the local information of the robot movement obtained from several sensors, this is known as *Dead Reckoning*, *Odometry* [1], [3], or *Local pose estimation*. This procedure has a fast response time but with it, the estimation error grows unbounded over time. Second, if the initial position of the robot is unknown, but it possesses a sensor that obtains the relative position between the landmarks in the environment and itself, then, the robot can build a map of the environment and obtain its absolute position simultaneously using the fusion of the movement and the environment sensors. These methods are known as simultaneous localization and mapping (SLAM) [4]–[6]. These strategies usually require a high consumption of resources due to the computational complexity of the solution and in some cases also because of the processing needed for the landmark sensor (mainly when it is vision based). This and other issues such as data association (loop-closure problem) and nonlinearities, make the real time implementation a challenging issue [7]. Third, using a complex sensor like a global positioning system (GPS, zenithal camera, or a radio-based sensor as seen in [8]), the absolute robot *pose* in the environment can be determined without using any of the local information. This procedure is known as *Global pose estimation* and has the disadvantage of low response time, the limitation of working only indoors (camera) or outdoors (GPS) depending on the sensor used, and, in the GPS case, the appearance of several problems like multipath propagation depending on the environment navigated [2].

As the robot position information is needed by the navigation algorithm, the localization method response time is important. If the robot location information is missed or delayed, the updated data would not be available for the control algorithm and it will produce a miss timed control action that can lead the system to an unstable behavior. Due to that, the local pose estimation is commonly used as the main source of information for the navigation algorithm and the GPS is commonly used to correct the local estimation when the global information is available [2], [9]. This is usually implemented using a fusion scheme based on a version of the Kalman filter (KF) like the linear KF, extended (EKF), or the unscented (UKF) (algorithms can be found in [10]–[13]). This takes into account the accuracy of each sensor and the robot model (Ackermann steering is used in this paper) to join the information optimally, increasing the localization accuracy. Although the estimation of the robot pose is essential, other tasks such as communications (supervision and coordination), sensor management (reading and calibration), and control algorithms (actuators drive and navigation) are also important and require processor time. This establishes a design compromise between the complexity of the localization technique and its precision when the robot is resource limited. Complex fusion schemes based on nonlinear models and filters (UKF and other fusion techniques like particle filters [14], [15]) will provide a precise estimation but will require longer execution times as they perform complex calculations (large matrix

Manuscript received December 14, 2012; revised May 27, 2013; accepted July 26, 2013. Date of publication August 23, 2013; date of current version April 25, 2014. Recommended by Technical Editor B. Shirinzadeh. This work was supported by FEDER-CICYT projects with references DPI2011-28507-C02-01 and DPI2010-20814-C02-02, financed by the Ministerio de Ciencia e Innovación (Spain). This work was also supported by the University of Costa Rica.

The authors are with the Department of Systems Engineering and Control, Instituto Universitario de Automática e Informática Industrial, Universidad Politécnica de Valencia, 46022 Valencia, Spain (e-mail: leomarpa@upv.es; mvalles@aii.upv.es; ansovi@ai2.upv.es; giuprog@isa.upv.es; pedro@aii.upv.es).

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/TMECH.2013.2277271

inverse and square root) and also require a larger number of parameter identification to obtain the robot model. On the other hand, linearized fusion schemes using the EKF take less time to calculate, but if the model is highly nonlinear, it will diverge quickly.

Considering these factors, a good localization algorithm should be resource efficient (bandwidth, processor time, and energy consumption [16]), it must take into account local and global information using all available sensors with a fusion algorithm, and, ideally, it should work very close in performance to more complex algorithms in order to provide a good estimation of the robot position. There are several works in this area, as stated in the next section.

II. Related Work

Mobile robot localization using sensor fusion is a well-researched topic and there are several examples of localization algorithms developed for different types of mobile robots and sensors. Many of them use local estimation only, mostly with EKF-based sensor fusion, and present only simulated results [17], [18] without the real robot constraints; or algorithms implemented on a computer external to the robot ([19] and [20] using Pioneer robots), with the disadvantage of communication delays and the added weight to the robot. Onboard implementation of the fusion algorithm is less common, requiring a robot with a powerful processing unit. This is the case presented in [21], where a multirate EKF fusion algorithm is used with the inertial measurements of a three-axis gyroscope and accelerometer and with an optical navigation sensor (laser mice type) with the advantage of nonslip measurement. The filter is tested using a custom build Ackerman type mobile robot showing good performance in the mapping of a 3-D pipeline; but without autonomous navigation. There are also few implemented examples on global estimation in indoor environments, using the EKF with an inertial measurement unit (IMU) and different global sensors, such as an ultrasonic satellite (U-SAT [22]), a zenithal camera [23], or radio frequency identification tags (see [24]), all showing an improvement in the localization over traditional encoder-only odometry.

The Ackermann type robot or car is used by many examples in outdoor environments; most of them use an IMU, a GPS, and a laptop to perform the fusion algorithm, once again it is EKF based in most cases. For example, in [25], an EKF is used to fuse the observations from a DGPS, a linear variable differential transformer for the steer angle, and the velocity encoder to obtain the pose estimate of a vehicle, and then use laser scan data as landmarks to improve the robot localization. A car is used in [26] with an IMU with two magnetometers and a single GPS antenna installed. The EKF fusion is performed in a notebook and the GPS data are used as soon as it is available (constant rate). A similar setup is used in [27] but aiding the EKF with a neural network trained using wavelet multiresolution analysis to aid the localization during GPS outages and performing the experimental test on road. In [28], a car equipped with a high-grade IMU and a GPS (with one and two antennas) is used with a full 3-D bicycle model for the Ackermann vehicle to estimate the car pose, a KF is used in the estimation of the model parameters. Also in [29], a two real time kinematic GPS units with three antennas and an IMU with a gyroscope is used in an adaptive EKF to estimate the robot pose. The sensor covariance is adapted to reflect the GPS outages reducing the estimation errors. In [30], a cascaded configuration is used: a first EKF is used to estimate the robot pose, and this is used as input for the second EKF to estimate the vehicle dynamics required for control. The system is tested in an Ackermann farm tractor using a GPS, a fiber-optic gyroscope (FOG), a doppler for longitudinal velocity measurement, and a potentiometer for measuring steer angle. Another car like experiment is performed in [31] but using a numerical algorithm that adapts the model structure online during GPS availability and works in cascade with a KF. This case is extensively tested in long distance showing good performance and low errors during GPS outages. Finally, a complete general review is presented in [2] for positioning and navigation, covering the general aspects of the systems commonly used for sensor fusion and localization improvement.

The examples presented are in most cases computationally expensive, and not implemented onboard the robot processor (except in [23] and [21]). Also, most of them use the GPS in a regular sample time basis, even if the local estimation error (using the IMU and the encoders) is small. A better approach, to save computational resources, would be to use the global information only when the odometry error is big enough (when it passes a predefined limit). This event-based update can save the process time and extend the battery life. To deal with these limitations in the current algorithms, the main objective of this paper is to develop a new form of the multisensor KF fusion algorithm, with similar performance when compared to more complex fusion schemes, but with lower computational cost and easy implementation on a limited resource mobile robot. This is achieved by using a new and simpler robot dynamic model with a cascade KF–EKF technique to perform the local sensor fusion first, and then use odometry from this fusion to estimate the robot pose, while using an event based correction from a zenithal camera when the local error passes a predefined threshold. As the platform cost is an important factor due to the current challenge of developing high-performance systems using low-cost technology [2], for this paper, the LEGO ®NXT will be used as it provides low-cost sensors and robot.

After this review, the paper is organized as follows. In Section III, the mathematical models needed by the KF equations are obtained for an Ackermann-type robot. The main contributions are exposed in Section IV, where the event-based localization algorithms are presented, and in Section V that shows the implementation aspects in the proposed platform. In Section VI, the performance and run time tests comparing the performance of the proposed methods are presented. Finally, some conclusions are drafted in the last section.

III. Models of an Ackermann Steering Robot

An Ackermann steering robot consists of a rigid body with center of mass and gravity denoted by P_0, turning radius R_G, mass M_G, and moment of inertia I_G, with two nonorientable

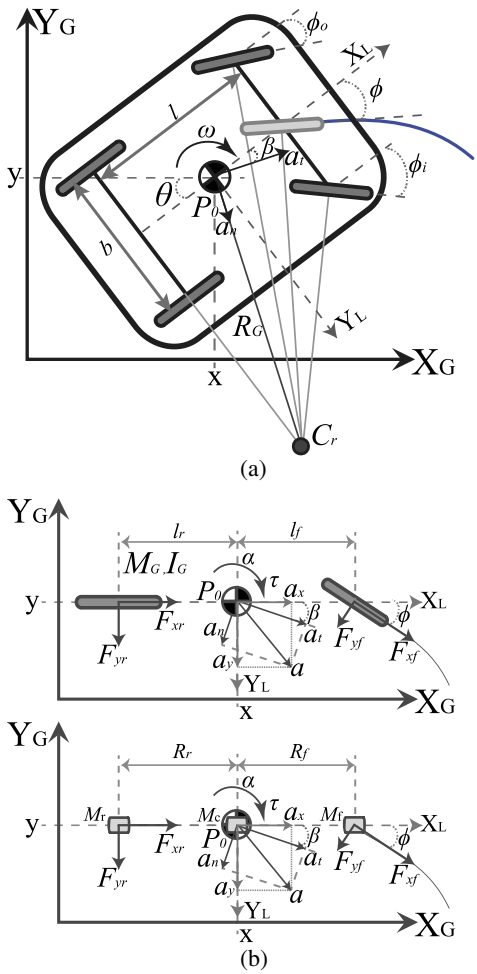

Fig. 1. Ackermann steering mobile robot. (a) Kinematics. (b) Dynamics.

(fixed) rear wheels separated by a distance b between them and a distance l from two orientable front wheels (also with a distance b between them) as shown in Fig. 1(a). The Ackermann steering system modifies the heading of the front wheels in a way that, at low speeds, all the tires are in pure rolling without lateral sliding [32]. This is accomplished when each wheel follows a curved path with different radius but with one common turn center C_r, as shown in Fig. 1(a). This system is analyzed using the bicycle model [32] and [33] shown in Fig. 1(b) assuming planar motion. With this, the mean of the inside and outside front wheels steer angles (ϕ_i and ϕ_o) is used (ϕ) and the back wheels are considered as one single wheel where the motor force is applied. The kinematics and dynamics of this robot are analyzed to obtain the model that will be used in the KF.

A. Kinematic Model

This model represents the robot velocities evolution in a fixed inertial frame. The robot pose is defined by its position $P_0 = (x, y)$ with the heading angle θ in the *Global* reference frame (X_G,Y_G) in Fig. 1(a). By knowing the linear and angular velocities (v and ω) in the *Local* frame (X_L,Y_L), the *global* velocities are defined as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}. \tag{1}$$

By discretizing and recursively integrating (1) with sample time T_s, the following robot *global* pose $\boldsymbol{L}$ is obtained:

$$\boldsymbol{L}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k-1} + T_s \begin{bmatrix} \cos\theta_{k-1} & -\sin\theta_{k-1} & 0 \\ \sin\theta_{k-1} & \cos\theta_{k-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}_{k-1}. \tag{2}$$

The absolute linear acceleration of P_0 in the global frame a, with components (a_x, a_y), is expressed in terms of the accelerations ($\dot{v}_x$, $\dot{v}_y$) and velocities in the local frame using its normal and tangential components (a_n, a_t) [32]

$$\begin{aligned} a_x &= \dot{v}_x - v_y\omega \\ a_y &= \dot{v}_y + v_x\omega. \end{aligned} \tag{3}$$

The kinematic relation between ϕ and ω is shown in

$$\omega = (v_x \tan\phi)/l. \tag{4}$$

With this, the robot dynamics are analyzed next.

B. Dynamic Model

This model represents the robot linear and angular accelerations (a,α) evolution in terms of the forces applied to it in the front (F_f) and rear (F_r) wheels and the angular moment (τ). The models in the literature represent the dynamics in the *global* reference frame (for example [34]–[38]) but these yield too complex nonlinear models that cannot be implemented in a fusion scheme using a limited resources mobile robot, or a model with many unknown parameters that have to be identified for each used robot and motor (they are not provided by most manufacturers). As the limitation in computational resources is quite common when working with mobile robots, a simpler model is needed to implement the fusion scheme. Also, as no direct measurements of the motor forces or input currents are available in most robots, obtaining the dynamic model in terms of the linear accelerations is convenient. These accelerations can be measured by two 3-D accelerometers placed above the center of each wheel axis, using a similar configuration of the one proposed in [39], where two one-axis accelerometers are used to obtain ω, but using the accelerometers to obtain v_x,v_y, and ω with the new dynamic model. Instead of using directly the second Newton's Law to obtain a and α in the center of mass, the robot rigid body can be studied as a dynamically equivalent particle system, formed by three mass particles M_r, M_c, and M_f joined by two massless connectors of constant length R_r and R_f (with $R_f = l_f = R_r = l_r = 0.5l$) as shown in Fig. 1(b). Using this equivalence and the conditions stated by [40] (full

development in Appendix), the accelerations are obtained as

$$\begin{aligned} a_x &= \lambda_a \left(a_{f,x} + a_{r,x}\right) \\ a_y &= \lambda_a \left(a_{f,y} + a_{r,y}\right) \\ \alpha &= \lambda_\alpha \gamma \left(a_{f,y} - a_{r,y}\right). \end{aligned} \tag{5}$$

The parameters of the particle system depend on M_G, I_G, and the robot shape. In the case of a robot with rectangular form, with length c and width b, they are obtained as

$$\begin{aligned} \gamma &= \frac{1}{6}(1 + (b/c)^2), \quad \lambda_a = 0.5 \\ \lambda_\alpha &= \frac{6}{c(1 + (b/c)^2)}, \quad \lambda_\alpha \gamma = \frac{1}{c}. \end{aligned} \tag{6}$$

Taking into account the normal acceleration components by substituting (5) into (3)

$$\begin{aligned} \dot{v}_x &= v_y\omega + \lambda_a \left(a_{f,x} + a_{r,x}\right) = v_y\omega + \lambda_a u_1 \\ \dot{v}_y &= -v_x\omega + \lambda_a \left(a_{f,y} + a_{r,y}\right) = -v_x\omega + \lambda_a u_2 \\ \dot{\omega} &= \lambda_\alpha \gamma \left(a_{f,y} - a_{r,y}\right) = \lambda_\alpha \gamma u_3. \end{aligned} \tag{7}$$

By discretizing and recursively integrating (7), the robot *local* dynamical model is obtained as

$$\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}_k = \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}_{k-1} + T_s \begin{bmatrix} v_y\omega + \lambda_a u_1 \\ -v_x\omega + \lambda_a u_2 \\ \lambda_\alpha \gamma u_3 \end{bmatrix}_{k-1}. \tag{8}$$

By substituting (8) as inputs of (2), the robot *global* dynamical model is written as

$$\begin{bmatrix} x \\ y \\ \theta \\ v_x \\ v_y \\ \omega \end{bmatrix}_k = \begin{bmatrix} x \\ y \\ \theta \\ v_x \\ v_y \\ \omega \end{bmatrix}_{k-1} + T_s \begin{bmatrix} v_x \cos\theta - v_y \sin\theta \\ v_x \sin\theta + v_y \cos\theta \\ \omega \\ V_y\omega + \lambda_a u_1 \\ -V_x\omega + \lambda_a u_2 \\ \lambda_\alpha \gamma u_3 \end{bmatrix}_{k-1}. \tag{9}$$

Finally, in case the robot travels at low speed, the nonslip condition [32], [33] can be assumed ($v_y \approx 0$), leading to the simplified local and global models

$$\begin{bmatrix} v_x \\ \omega \end{bmatrix}_k = \begin{bmatrix} v_x \\ \omega \end{bmatrix}_{k-1} + T_s \begin{bmatrix} \lambda_a u_1 \\ \lambda_\alpha \gamma u_3 \end{bmatrix}_{k-1} \tag{10}$$

$$\begin{bmatrix} x \\ y \\ \theta \\ v_x \\ \omega \end{bmatrix}_k = \begin{bmatrix} x \\ y \\ \theta \\ v_x \\ \omega \end{bmatrix}_{k-1} + T_s \begin{bmatrix} v_x \cos\theta \\ v_x \sin\theta \\ \omega \\ \lambda_a u_1 \\ \lambda_\alpha \gamma u_3 \end{bmatrix}_{k-1}. \tag{11}$$

The models (2), (5), and (8) to (11) are used in conjunction with the KF to estimate the global pose of the robot as shown in the next section.

Algorithm 1: Recursive EKF algorithm

Input : u_k,z_k,x_{k-1},P_{k-1}
Output: $\hat{x}_k$,P_k
Data: f and h form model (9),Q_k,R_k
Initialization: x_0,P_0
for *current time* k **do**
  Prediction Step:
  $\hat{x}_k = f\left(\hat{x}_{k-1}, u_{k-1}, 0\right)$
  $\mathbf{A}_k = \frac{\partial f}{\partial x} |\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, 0$
  $\mathbf{H}_k = \frac{\partial h}{\partial x} |\hat{\mathbf{x}}_k, 0$
  $\mathbf{W}_k = \frac{\partial f}{\partial w} |\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, 0$
  $\mathbf{V}_k = \frac{\partial h}{\partial v} |\hat{\mathbf{x}}_k, 0$
  $P_k = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$
  Correction Step:
  $K_k = P_k H_k^T \left(H_k P_k H_k^T + V_k R_k V_k^T\right)^{-1}$
  $\hat{x}_k = \hat{x}_k + K_k \left[z_k - h\left(\hat{x}_k, 0\right)\right]$
  $P_k = \left(I - K_k H_k\right) P_k$
end

IV. Event-Based Localization

In this section, the proposed event-based localization schemes are described, but first a short review of the time-based method is presented. To simplify the notation, the models used in the algorithms are referred as *linear* when it can be written as (12a) with input $\boldsymbol{u}_k \in \Re^u$, measurement $\boldsymbol{z}_k \in \Re^m$, and state $\boldsymbol{x}_k \in \Re^n$, or *nonlinear* when a nonlinear function $\boldsymbol{f}$ is needed to relate the state at time k with the previous time step $k-1$ and $\boldsymbol{h}$ to relate the x_k to z_k as shown in (12b). Also, the terms $\boldsymbol{w}_k$, $\boldsymbol{v}_k$ represent the process and measurement noises at time k that have an independent, white probability distribution with zero mean.

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1}, \quad \boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k \tag{12a}$$

$$\boldsymbol{x}_k = f\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}\right), \quad \boldsymbol{z}_k = h\left(\boldsymbol{x}_k, \boldsymbol{v}_k\right). \tag{12b}$$

A. Global EKF Localization With Time-Based Update

This method is the "traditional" one, which uses the model (9) [or (11)] and all the available measurements to estimate the robot pose at every time k. If the available measurements are $v_{x,enc}$, $v_{y,enc}$, and ω_{enc} from the encoders, ω_{gyr} from a gyroscope, u_1, u_2, u_3 from two 3-D accelerometers [placed as shown in Fig. 1(a)] and the global position information $(x, y, \theta)_{GM}$, obtained from a zenithal camera or other global sensor, then $\boldsymbol{z}_k = [x_{GM} \;\; y_{GM} \;\; \theta_{GM} \;\; v_{x,enc} \;\; v_{y,enc} \;\; \omega_{enc} \;\; \omega_{gyr}]^T$ is used with $\boldsymbol{u}_k = [u_1 \;\; u_2 \;\; u_3]^T$ in (9). This model is used in the EKF algorithm (Algorithm. 1) which performs the sensor fusion at every time k (time based) using the process and measurement noises covariance matrices $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$ respectively, and obtains the state vector estimate $\hat{\boldsymbol{x}}_k$, i.e., the robot pose in (9) and the covariance of the estimation error $\boldsymbol{P}_k$. This method has all the available information at any time k (all available sensors are

used in z_k and u_k) but requires large matrix inversion to obtain the filter gain K_k; thus, large memory and resources are needed.

B. Local EKF Localization With Global KF Event-Based Update

A first approach to reduce the communication, processor and memory requirements is to reduce z_k dividing it into the local and global measurements. With this, the robot pose is determined every time k using (9) in the EKF and the local $\boldsymbol{z}_k = [v_{x,enc} \;\; v_{y,enc} \;\; \omega_{enc} \;\; \omega_{gyr}]^T$ with $\boldsymbol{u}_k = [u_1 \;\; u_2 \;\; u_3]^T$, and using the global position $L_{GM} = [x_{GM} \;\; y_{GM} \;\; \theta_{GM}]^T$ to correct the local estimation on an event-based scheme. The basic idea is to use L_{GM} only when the error in the estimation is big enough, indicated by an event that uses the pose section of P_k in the EKF. This $\boldsymbol{P}_{k,xy}$ is used to obtain the *area* of the $3-\sigma$ error interval ellipsoids using (13), where the ellipsoids axis are a_σ and b_σ and $l_e = 3$ for the $3-\sigma$ error. With this, the event is generated using the ratio $\boldsymbol{R}_A$ of the ellipsoid area A_{ellip} and the robot area A_{NXT}. This is a normalized indicator of the moment in which the error in the robot position is as large as the size of the robot. For example, when $\boldsymbol{R}_A$ exceeds a certain level $R_{A,lim}$, e.g., 1.6 (indicating that the error area is 1.6 times the robot area), then L_{GM} is used to correct the robot pose. This limit is chosen as a compromise between the number of calls (energy and processor time consumption) and the desired precision of the estimated position, being a reasonable range $0.6 \leq \boldsymbol{R}_A \leq 2$ for the LEGO NXT

$$\boldsymbol{P}_{k,xy} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 \\ \sigma_{xy}^2 & \sigma_y^2 \end{bmatrix} \quad a_\sigma = \sqrt{\frac{2l_e^2\,|P_{xy}|}{\sigma_x^2+\sigma_y^2+T_\sigma}}$$

$$T_\sigma = \sqrt{\sigma_x^4+\sigma_y^4-2\sigma_x^2\sigma_y^2+4\sigma_{xy}^4}, \quad b_\sigma = \sqrt{\frac{2l_e^2\,|P_{xy}|}{\sigma_x^2+\sigma_y^2-T_\sigma}}$$

$$A_{NXT} = b \cdot c \quad A_{ellip} = \pi a_\sigma b_\sigma. \tag{13}$$

To correct the pose with L_{GM}, the estimated pose $x_{k,p}$ of the EKF output can be replaced by L_{GM} and $P_{k,p} = P_{k,x,y,\theta}$ by the corresponding values of P_0. But a better approach is to fuse the EKF output with L_{GM} taking into account the global sensor accuracy. This is done by using the EKF output (pose states and error covariances) as the prediction step of a linear KF. The model used in the fusion is (12a) with A and H_p both being the identity matrix $\mathbb{I}_{3,3}$ and no input. The $R_{k,GM}$ values are obtained from the global sensor accuracy. With this, the KF will fuse the local estimate with the global measurement. The resulting $P_{k,p}$ decreases when the pose is corrected, resetting the event. The event-based EKF is shown in Algorithm 2. A cascaded model approach can also be used, as described next.

C. Local Cascaded EKF Localization With Global KF Event-Based Update

This approach takes a further step in reducing computational requirements. Instead of using (9) with the full state, the velocities in (8) are used to perform the local fusion, and then, (2) is used to obtain the robot pose. With this $x_k = [v_x, v_y, \omega]$, and again $\boldsymbol{u}_k = [u_1 \;\; u_2 \;\; u_3]^T$ and $\boldsymbol{z}_k = [v_{x,enc} \;\; v_{y,enc} \;\; \omega_{enc} \;\; \omega_{gyr}]^T$. Also L_{GM} is used to correct the local estimation in the event-based scheme. This reduction in both x_k and z_k (comparing to the time based scheme) allows a fast calculation in the matrix inversion required for K_k, reducing the memory use and leaving resources for other tasks. As the uncertainty in the velocity measurement is not propagated to the pose estimation as does the previous methods, a linear recursive approximation is used to propagate the covariance from the velocity $\boldsymbol{P}_{k,v_x,v_y,\omega}$ to the pose $\boldsymbol{P}_{k,p}$ and also propagate it in time [13], [41] as shows (14). In this, $\nabla \boldsymbol{F}_u$ is the gradient operator applied to (2) with respect to the inputs (v_x,v_y,ω) and $\boldsymbol{Q}_x$ are set equal to the $Q_{k,pose}$ terms in the EKF. The cascaded event-based EKF is shown in Algorithm 3. A final algorithm can be proposed with a smaller x_k, as described next

$$\boldsymbol{P}_{k,p} = \boldsymbol{P}_{k-1,p} + \nabla \boldsymbol{F}_u \boldsymbol{P}_{k,v_x,v_y,\omega} \nabla \boldsymbol{F}_u^T + \boldsymbol{Q}_x. \tag{14}$$

Algorithm 2: Recursive EKF algorithm with an event-based global KF update

Input : u_k,z_k,x_{k-1},P_{k-1},L_{GM}
Output: $\hat{x}_k$,P_k
Data: f and h from model (9),Q_k,R_k,A_{NXT}
Initialization: x_0,P_0
for *current time* k **do**
  Prediction Step EKF (same as Alg. 1)
  Correction Step EKF:
  $K_k = P_k H_k^T \left(H_k P_k H_k^T + V_k R_k V_k^T\right)^{-1}$
  $\hat{x}_k = \hat{x}_k + K_k \left[z_k - h\left(\hat{x}_k, 0\right)\right]$
  $P_k = \left(I - K_k H_k\right) P_k$
  $3-\sigma$ ellipsoid area A_{ellip} using (13)
  $\boldsymbol{R}_A = A_{ellip}/A_{NXT}$
  if $\boldsymbol{R}_A > R_{A,lim}$ **then**
    Global Correction Step, pose KF
    $K_{k,KF} = P_{k,p} H_p^T \left(H_p P_{k,p} H_p^T + R_{k,GM}\right)^{-1}$
    $\hat{x}_{k,p} = \hat{x}_{k-1,p} + K_{k,KF}\left(L_{GM} - H_p \hat{x}_{k-1,p}\right)$
    $P_{k,p} = \left(I - K_{k,KF} H_p\right) P_{k,p}$
  end
end

D. Local Cascaded KF Localization With Global KF Event-Based Update

This approach takes advantage of the nonslip condition ([33], [32]) in case the robot travels at low speed, using the velocities in (10) to perform the local fusion, and then uses (2) to obtain the robot pose. As (10) is linear, the fusion is performed using the KF instead, saving more computational resources (compared to Algorithms 1 to 3) as the linearization step of the EKF is not needed. With this $x_k = [v_x, \omega]$ and $\boldsymbol{u}_k = [u_1 \;\; u_3]^T$. In replacement of the v_y measurement, ω_{comp} from a compass is used in $\boldsymbol{z}_k = [v_{x,enc} \;\; \omega_{enc} \;\; \omega_{gyr} \;\; \omega_{comp}]^T$. Also L_{GM} is used to correct the local estimation in the event-based scheme by using the covariance propagation of (14). Without

Algorithm 3: Recursive-cascaded EKF algorithm with an event-based global KF update

Input : u_k,z_k,x_{k-1},P_{k-1},L_{GM}
Output: $\hat{x}_k$,P_k
Data: f and h from model (8),Q_k,R_k,A_{NXT},Q_x
Initialization: x_0,P_0
for *current time* k **do**
  Prediction Step EKF (same as Alg. 1)
  Correction Step EKF (same as Alg. 1)
  Pose estimation using (2)
  Covariance propagation using (14)
  $3-\sigma$ ellipsoid area A_{ellip} using (13)
  $\boldsymbol{R}_A = A_{ellip}/A_{NXT}$
  if $\boldsymbol{R}_A > R_{A,lim}$ **then**
    Global Correction Step, pose KF (same as Alg. 2)
  end
end

Algorithm 4: Recursive-cascaded KF algorithm with an event-based global KF update

Input : u_k,z_k,x_{k-1},P_{k-1},L_{GM}
Output: $\hat{x}_k$,P_k
Data: A, B and H from model (10),Q_k,R_k,A_{NXT},Q_x
Initialization: x_0,P_0
for *current time* k **do**
  Prediction Step KF:
  $\hat{x}_k = A\hat{x}_{k-1} + Bu_{k-1}$
  $P_k = AP_{k-1}A^T + Q_k$
  Correction Step KF:
  $K_k = P_k H_k^T \left(H_k P_k H_k^T + R_k\right)^{-1}$
  $\hat{x}_k = \hat{x}_k + K_k\left[z_k - H\hat{x}_k\right]$
  $P_k = \left(I - K_k H_k\right) P_k$
  Pose estimation using (2)
  Covariance propagation using (14)
  $3-\sigma$ ellipsoid area A_{ellip} using (13)
  $\boldsymbol{R}_A = A_{ellip}/A_{NXT}$
  if $\boldsymbol{R}_A > R_{A,lim}$ **then**
    Global Correction Step, pose KF (same as Alg. 2)
  end
end

the estimation of the slip, this method is less accurate in the case of high-speed movements or in certain surfaces, but this can be compensated (within certain limit) by using a low $R_{A,\lim}$ value. The cascaded event-based KF is shown in Algorithm 4. With this, the selected platform, the LEGO NXT is described below along with the experimental results.

V. Implementation in the LEGO NXT

The tests are performed in an Ackermann drive LEGO NXT. In this section, the platform is described along with the local sensor calibration and preprocessing, the navigation algorithm, and the global sensor scheme. Finally, the platform and filter parameters are exposed.

A. Test Platform Description

The LEGO Mindstorms ®NXT is a low-cost mobile robot platform. Its control unit, the NXT, is based on an ARM7 32-bits microcontroller with 256-kB FLASH and 64 kB of RAM. For programming and communications, NXT has an USB 2.0 port and a wireless Bluetooth class II, V2.0 device, and four inputs and three analog outputs. The basic pack offers four electronic sensors: touch, light, sound, and distance. But in addition of these default sensors, nowadays a great variety of different sensor is available, as for example vision cameras, magnetic compass, accelerometers, gyroscopes, infrareds searchers, etc. (www.mindsensors.com, www.hitechnic.com). The actuators consist of dc motors that have integrated the encoder sensors with a 1° resolution. A more detailed description about LEGO NXT motors can be found in www.philohome.com/motors/motorcomp.htm. The LEGO NXT robot used in the tests is shown in Fig. 2. The sensors used in the fusion scheme are two accelerometers placed above the center of each wheel axis [for the dynamic model, Fig. 1(b)], one gyroscope, one magnetic compass, and the wheels encoders. The programing platform used is the Java-based LeJOS, due to the programming advantages in the communications between a robot and a supervising PC. Although the microcontroller can perform many tasks given enough time, the main limitation of this robot is the memory. It has a limit of 255 local variables, 1024 constants, 1024 static fields, and a maximum code length of 64 kb when programing with LeJOS. Because of this, it is an excellent option to implement the proposed filters as it does not have enough memory to implement a large KF with many states and measurements.

B. Local Sensors Calibration and Preprocessing

The used local sensors must be calibrated to remove any bias, this can be accomplished by doing a preliminary test with the robot, with the sensors measuring the same value for some time. Also, their values must be in the same units of measurement (S.I. in this paper). The preprocessing is done to obtain v and ω from the sensor data. Integrating the encoders reading with a sample time of $T_s = 50$ ms, v_x is obtained from the rear encoder, and ϕ from the front. With this, ω_{enc} is obtained from (4) and $v_{y,enc} = v_x \tan\phi$. The gyroscope directly measures ω_{gyr}. and θ_{comp} is measured from the compass in a range $[0, 2\pi]$, so it must be accumulated to obtain a continuous measure of θ and then derivate it to obtain ω_{comp}. As both, the calibration and the preprocessing steps, depend on the sensors, these processes should be done for every robot that is used in the experiments and also when a sensor or a motor is replaced. The robot sensors are marked in Fig. 2. Also, two gears are placed at the motors

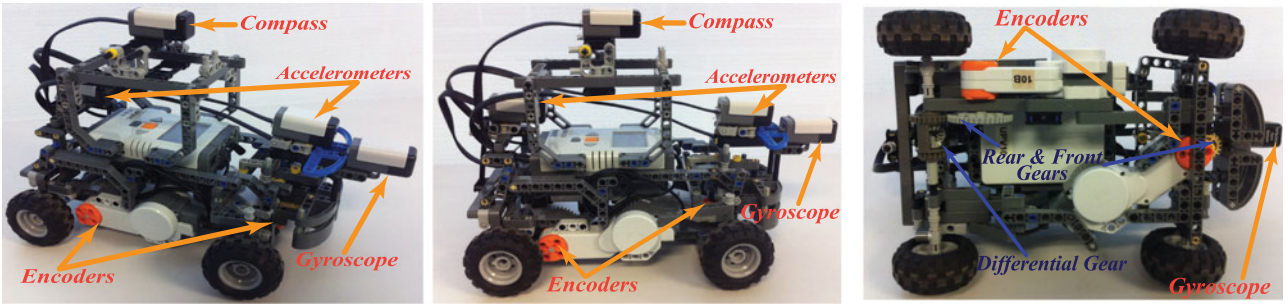

Fig. 2. LEGO NXT mobile robot used in this paper.

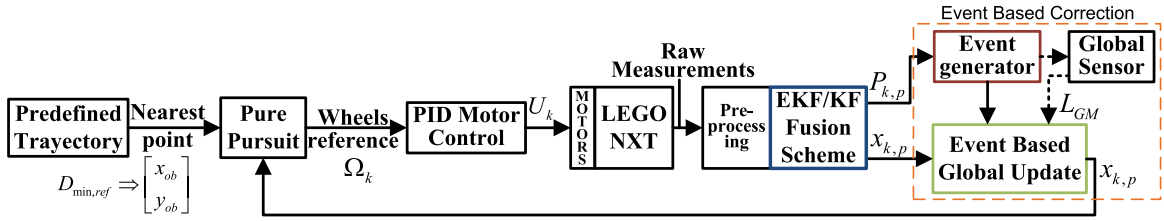

Fig. 3. Navigation and localization algorithm.

output to increase the movement range allowing a smoother control in the steering. Finally, a differential is added to transmit the motor movement to the wheels, while allowing them to rotate at different speeds (being v_x the average of them), allowing the correct turn of the vehicle.

C. Navigation Algorithm

The tests are performed using the navigation algorithm shown in Fig. 3, where $\boldsymbol{\Omega}_k = [\,\omega_r \quad \omega_f\,]^T_{\text{ref}}$ and $\boldsymbol{U}_k = [\,u_r \quad u_f\,]^T$ are the velocity reference vector and the motor control action vector for the rear and front motors, respectively. In this, a predefined trajectory is stored in the robot memory (for example a square, a circle, etc.) as a set of global points. Knowing the initial position of the robot, the algorithm calculates the distance $D_{\min,\text{ref}}$ to the trajectory points to determine the nearest one, then, a *look ahead* distance is added to obtain the objective point $(x, y)_{ob}$. This point is used by the navigation algorithm, a pure pursuit controller (see examples in [1] and [3]), or a decentralized point controller, to determine Ω_k. This reference is followed by a PID controller that generates the control action for the motors $\boldsymbol{U}_k$ to follow the reference velocities, and so the desired path. One sample time later, the sensors in the robot measure all the inputs for the KF fusion scheme which estimates the actual position $\boldsymbol{x}_{k,p}$. This is used again by the algorithm to obtain a new objective point and keep the robot moving in the desired trajectory.

D. Global Sensor Scheme

This is the system shown in Fig. 4 being composed by the camera (640 × 480, 30 frames/s), the server that processes the image (camera server—C.S.) and the server that communicates with the robot (supervision server—S.S.). The C.S. executes a Java-based program that constantly gets the image from the webcam, and makes the image processing to obtain the globally measured pose along with the time taken by the server to obtain

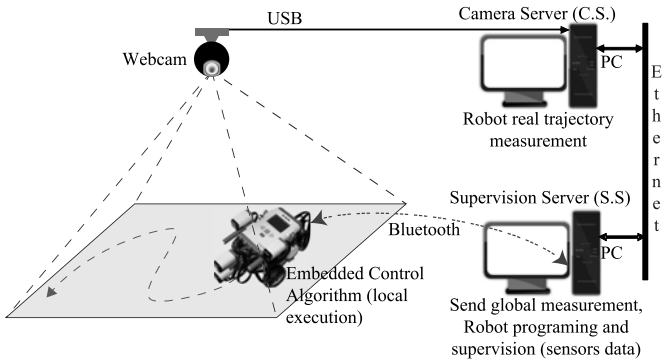

Fig. 4. Hardware setup, global position measurement.

this value T_{cam} (50 ms in average but the sent value is the actual measured time). This information forms the message that is communicated to the S.S. This server also stores every measure in a file and a video for the later analysis of the robot movement. When the event is generated, the robot asks the S.S. via Bluetooth for a global measurement. As the S.S. executes a Java program that is continuously listening for the robot calls, the request is processed immediately, so it sends the last message (pose and time) received from the C.S. to the robot.

To deal with the communication delay, several actions are taken when an event occurs. First, the output provided is the EKF/KF estimation until the message from the S.S. arrives, while a timer counts the time elapsed between the camera call and the reception of the S.S. message T_{ss}. Also the velocities in the global axis Vx_e, Vy_e of the robot at the event time are stored in the robot memory. When the message arrives with the global position, (15) is used to obtain the displacement caused by the delay $\Delta\,(x, y)$ adding it to the received (x_r, y_r) from the camera to obtain the L_{GM} used by the event-based filter, assuming that the camera measure is performed as soon as the message is received by the S.S. The received heading θ_r is assumed to be

TABLE I
FILTER PARAMETERS FOR THE LEGO NXT

	State	Alg 1	Alg 2	Alg 3	Alg 4
Q	x	0.0001		0.0001 (Q_x)	
	y				
	θ				
	v_x		0.01		0.01
	v_y				-
	ω		0.03		0.9
	Measurement	Alg 1	Alg 2	Alg 3	Alg 4
R	x_{GM}	0.015	0.015 (event based)		
	y_{GM}				
	θ_{GM}	0.00001	0.00001 (event based)		
	$v_{x,enc}$		0.01		
	$v_{y,enc}$		0.098		-
	ω_{enc}		0.038		
	ω_{gyr}		0.013		
	ω_{comp}		-		0.97

constant during T_{ss}

$$\Delta T = T_{\text{cam}} + 0.5\,(T_{ss} - T_{\text{cam}}) \qquad L_{GM,x} = x_r + \Delta X$$
$$\Delta X = (V x_e \Delta T) \qquad L_{GM,y} = y_r + \Delta Y$$
$$\Delta Y = (V y_e \Delta T) \qquad L_{GM,\theta} = \theta_r. \qquad (15)$$

Finally, to preserve the linear displacement assumed by (15), a final condition is checked by the algorithm. There must be no global updates performed when the robot is in a hard movement (i.e., when applying a big control action to the steer motor to negotiate a curve or to avoid an obstacle). This is done by checking the PID control. If the error (difference between the desired and real value) in the steer motor velocity exceeds a certain value ($|e_f| > 2$), then the robot is assumed to be in a hard curve. In this case, the algorithm waits 1 s and then checks again this condition. If it is false, then it makes again the global update procedure and outputs the corrected position.

E. Platform and Filter Parameters

The required parameters for the fusion scheme are obtained for the Lego NXT. For the dynamic model, the parameters correspond to the solid box robot with $b = 154$ mm, $c = 167$ mm, and $\lambda_a = 0.5$ in (6). The measurement noise covariance matrix $\boldsymbol{R}$ and the system error covariance matrix $\boldsymbol{Q}$ are obtained from the experimental tests, doing a trial and error tuning using the sensor data obtained from the robot when a square and a circular trajectory are followed. The criteria used to make the adjustment were that the most suitable parameters are the ones that make the KF trajectory estimated very similar to the one measured by a zenithal camera. The adjustment was done knowing that a small value in a term of these matrices means that the KF will consider this value more relevant for the estimation than the others. For example, a low value of the ω_{gyr} means that the filter will perform the estimation based mainly in this measurement, so the others ω sensors and model will be less relevant. The matrices Q and R are both diagonal, with the corresponding values for the states and measurements showed in the Table I. After discussing the implementation issues, the preformed tests are presented next.

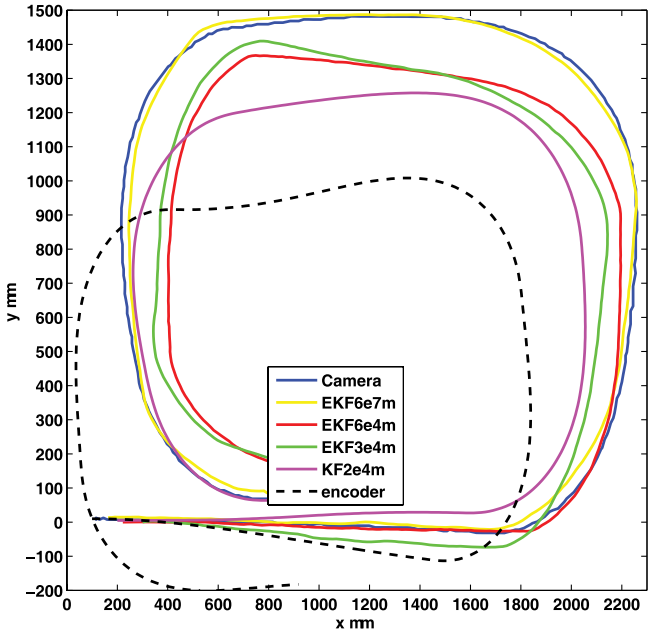

Fig. 5. Algorithms performance, simulation test.

VI. EXPERIMENTAL TESTS WITH THE LEGO NXT

To test the performance of the proposed algorithms with the event-based global update (*EBGU*), several tests are performed in an Ackermann drive LEGO NXT. In this section, the localization performance and execution time test are presented.

A. Performance Tests

A first test is performed without using the event-based correction. The robot is set to follow a square reference trajectory, while recording the local measurements and the information of the C.S. to use in an offline MATLAB simulation. This is shown in Fig. 5, where all the algorithms show good performance. The closest estimation to the camera measure is the one for Algorithm 1, the EKF with six states and seven measurements (EKF6e7m), as it has access to the global measurement. The other algorithms, EKF6e4m in Algorithm 2, EKF3e4m in Algorithm 3, and KF2e4m in Algorithm 4, show a close estimation to the camera measurement, but it is clear that the localization accuracy is degraded when using only a less resource demanding estimation scheme (simpler model or less measurements) without the global information. Also, using only the encoders is insufficient in this platform.

The proposed event-based localization scheme will increase the accuracy of these algorithms to be close to EKF6e7m, as the second test in Fig. 6 shows. In this, the proposed EKF/KF fusion schemes are implemented (onboard) on the LEGO NXT (Algorithms 2 to 4, using the EBGU) to follow a square trajectory. Algorithm 1 is not implemented as it exceeds the T_s and the memory limit. Again, good performance is observed as the localization solution is closer to the camera measurement than when the event-based update is not used.

The final performance test, a long-distance run, is done by using the less resource demanding algorithm, KF2e4m, setting the robot to follow a double square and a square trajectory for

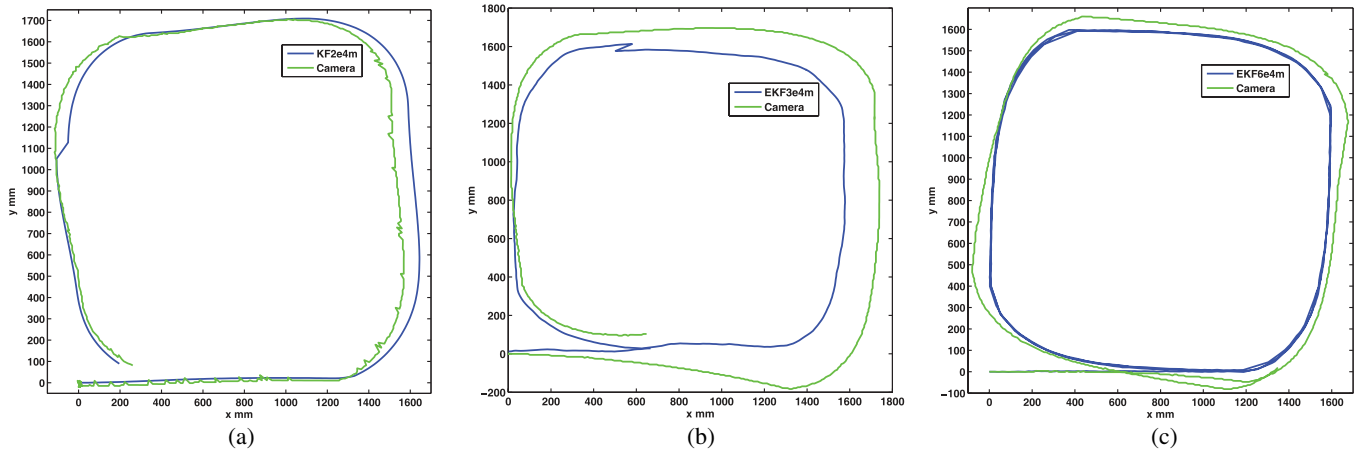

Fig. 6. Algorithms performance, EBGU, onboard the LEGO NXT. (a) EKF2e4m. (b) EKF3e4m. (c) EKF6e4m.

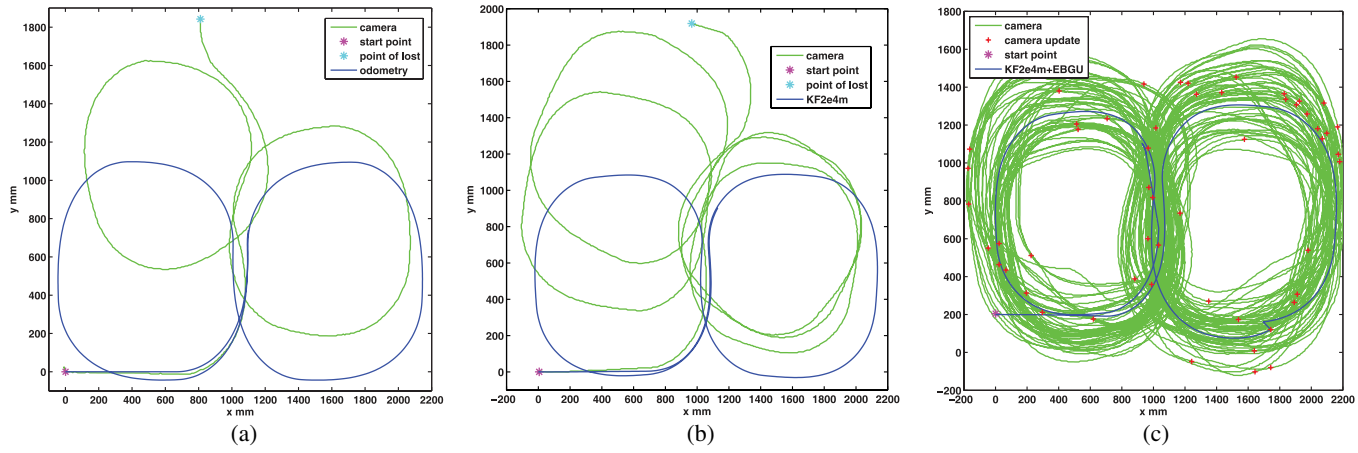

Fig. 7. 30-minutes run, double square trajectory. (a) Odometry. (b) KF2e4m. (c) KF2e4m+EBGU, $R_A = 1.2$.

30 min, as the experimental results of Figs. 7 and 8 show. In Fig. 7(a), the localization is done only using odometry from the encoders. Thence, the estimated position quickly diverges from the actual one. With the KF2e4m without the EBGU [see Fig. 7(b)], the pose estimation diverges slowly as the error covariance increases. But in the case of the KF2e4m with the EBGU [see Fig. 7(c)], the position is estimated properly as the trajectory remains very similar to the double square reference and the position covariance does not increase indefinitely. This is observed again in Fig. 8 for the square trajectory. Although the error bound for the KF2e4m may seem large [see Figs. 7(c) and 8], it can be decreased to fulfill the mission requirements by lowering the R_A value. This is done as a compromise between the method accuracy and the available resources and bandwidth in the robot and S.S. With these tests, the stability and convergence of the KF2e4m with EBGU are observed for large distance runs, and so the more complex schemes (KF6e4m and KF3e4m) will also perform well under similar circumstances. A video of the long-distance runs can be found in http://wks.gii.upv.es/cobami/webfm_send/6. The execution time of the different algorithms is analyzed next.

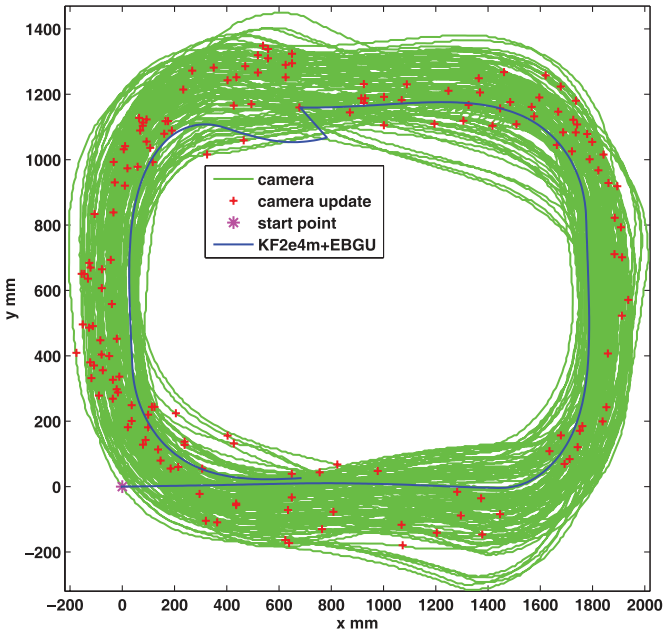

Fig. 8. 30-minutes run, square trajectory.

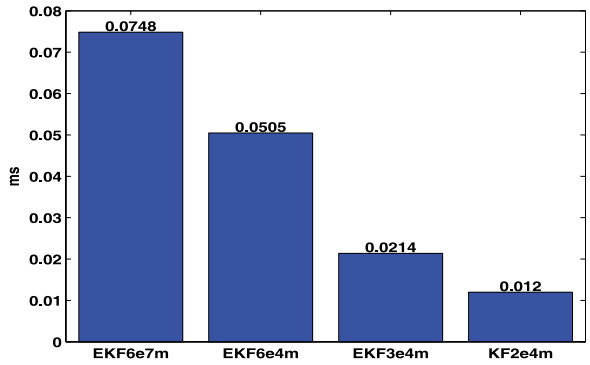

Fig. 9. Mean execution time test, PC simulation.

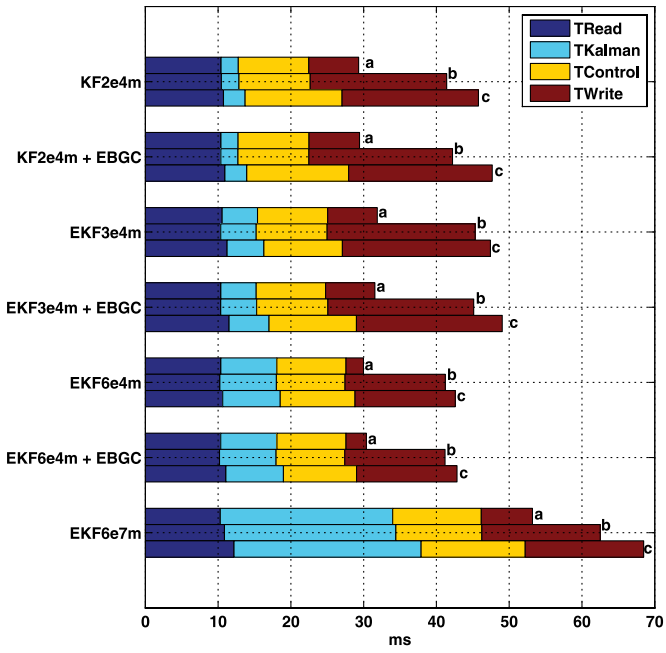

Fig. 10. Execution time test, experimental results onboard the LEGO NXT.

B. Run Time Test

To evaluate the run time efficiency of the proposed algorithms, the mean execution time of the simulation performed in Fig. 5 is measured, using MATLAB running on a computer (2.4 GHz with 4-GBRAM). This is shown in Fig. 9. From this test, it is clear that the proposed methods with the cascaded configuration (EKF3e4m and KF2e4m) have less computational effort (as they use less ms of processor time) to estimate the robot pose, when comparing with methods that use a full state estimation approach (EKF6s4m and EKF6s7m). Although EKF3e4m and KF2e4m use less time, they have similar performance to that of the more complex ones, as it was shown in the performance tests.

A second test, where the execution time is measured for the different tasks running inside the robot, is performed. In this, the proposed algorithms (with and without the EBGC) are compared with the EKF6s7m (which, due to the limit in the robot memory, was executed by sections to obtain the different task times). The measured tasks are the sensor reading (with calibration and preprocessing), the KF, the control algorithm (navigation and motor control), and the write task which stores the variables needed for supervision in a text file. This is shown in Fig. 10. Execution times are measured every cycle over a 1 min test, and, as they are not constant, three cases are presented. Case a shows the mean task time per cycle, case b shows the time of the worst execution, and case c shows the maximum task times measured during the test, although they do not occur at the same time at any cycle (or through the 30 min run), but these are an indicator of the worst case possible. This test shows that only the proposed event-based methods fulfill the sample time of 50ms for cases a, b, and c, being the KF2e4m the less resource consuming of all filters. The EKF6s7m exceeds the T_s in all the cases, and so is not suitable for being implemented in this robot because of the low available memory (when the calculation of K_k is performed in every cycle) and also, when this sample time is needed.

VII. Conclusion

Three efficient sensor fusion scheme using the KF and EKF with a new particle dynamical model of an Ackermann wheel mobile robot, along with an event-based global position update, have been presented as a solution for the localization problem. The results show that the performance of the proposed algorithms with the new model is similar to those obtained by using the more complex EKF with larger state and measurement vectors, but with a faster execution and less memory usage, allowing the implementation of the algorithm inside a limited resource robot, while leaving enough resources for other tasks that must be executed inside the robot. Also, as the event-based update uses less bandwidth for the localization task, more can be allocated to other data transmissions, such as robot coordination, formation control, system monitoring, diagnostics, etc.

The local sensor fusion can be adapted to add more or less sensors according to the robot capabilities and the available sensors, adjusting the KF matrices (dimensions and values of Table I) as needed. If the robot has very limited resources, this method can work using only the encoder measurements and one heading angle/rate sensor. Also, both accelerometers can be substituted for a model (identified from the experimental data) that relates the motors control action and the linear accelerations of the real wheels and the steering, to use them as inputs in the proposed dynamic model. On the other hand, if the robot has large resources availability, the proposed Algorithm 2 can be used to estimate the pose, while saving bandwidth when communicating to a zenithal camera (or a similar off-board global sensor). In this case, it can also be adapted to a more complex sensor fusion filter like the UKF or the Cubature KF, to improve the accuracy of the pose estimate.

In all the performance and execution time results, improved pose estimation over the traditional odometry and EKF6e7m fusion method was observed. Also, the proposed algorithms reduce the processor usage and battery consumption in the mobile robot, as it updates the position only when it is necessary.

As future work, the method can be extended into different mobile robots configurations such as the Omnidirectional, by adding particles to the proposed equivalent dynamical model. Also, different sources of global information can be used, for example, a GPS to develop the outdoor case, or a scanning laser rangefinder to extend the method into a SLAM algorithm.

Finally, the proposed algorithms can be modified to solve a multirobot localization scenario, using the relative robot pose information in the event-based scheme as mobile landmarks while saving bandwidth and resources.

APPENDIX

DYNAMICALLY EQUIVALENT PARTICLE SYSTEM

To study the robot rigid body as a dynamically equivalent particle system, formed by three mass particles M_r, M_c, and M_f joined by two massless connectors of constant length R_r and R_f [see Fig. 1(b)], the following conditions must be met [40]:

1) Mass Conservation: $M_G = M_f + M_c + M_r$
2) Mass Center Conservation: $M_f R_f = M_r R_r$
3) Moment of Inertia Conservation: $M_f R_f^2 + M_r R_r^2 = I_G$

By defining $R_f = R_r = R_N = 0.5l$, the masses are obtained in (A.1), where M_f and M_r are proportional to the total mass M_G expressed by the constant γ. This is true for M_c but with the constant δ. Also the constant λ_a is defined from γ and δ, and λ_α from M_G, R_N, and I_G

$$M_f = M_r = \frac{I_G}{2R_N^2} = \gamma M_G, \quad \lambda_a = \frac{\gamma}{1-\delta}$$
$$M_c = M_G - \left(\frac{I_G}{R_N^2}\right) = \delta M_G, \quad \lambda_\alpha = \frac{M_G R_N}{I_G}. \tag{A.1}$$

The parameters of the particle system are obtained in (A.2) for a robot of rectangular shape with length c and width b, by substituting I_G in (A.1). The simplified parameters are shown in (6). This procedure can be extended to other robot shapes substituting the corresponding I_G in (A.1)

$$I_G = \frac{1}{12} M_G \left(b^2 + c^2\right), \quad \gamma = \frac{1}{6}\left(1 + (b/c)^2\right)$$
$$\delta = \left(1 - \frac{1 + (b/c)^2}{3}\right), \quad \lambda_\alpha = \frac{6}{c\left(1 + (b/c)^2\right)}$$
$$\lambda_a = 0.5, \quad \lambda_\alpha \gamma = 1/c. \tag{A.2}$$

To obtain the dynamic model, the second Newton's law is applied to both the rigid body and the particle system [see Fig. 1(b)] to obtain (A.3) and (A.4)

$$\Sigma F_x = (F_{xf}\cos\phi - F_{yf}\sin\phi) + F_{xr} = M_G a_x$$
$$\Sigma F_y = (F_{yf}\cos\phi + F_{xf}\sin\phi) + F_{yr} = M_G a_y$$
$$\Sigma \tau = l_f\,(F_{yf}\cos\phi + F_{xf}\sin\phi) - l_r F_{yr} = I_G \alpha \tag{A.3}$$

$$(F_{xf}\cos\phi - F_{yf}\sin\phi) + F_{xr} = M_f a_{f,x} + M_c a_{c,x} + M_r a_{r,x}$$
$$(F_{yf}\cos\phi + F_{xf}\sin\phi) + F_{yr} = M_f a_{f,y} + M_c a_{c,y} + M_r a_{r,y}$$
$$R_f\,(F_{yf}\cos\phi + F_{xf}\sin\phi) - R_r F_{yr} = R_f M_f a_{f,y} - R_r M_r a_{r,y}. \tag{A.4}$$

D'Alembert's principle is used and (A.3) is set equal to (A.4). They are dynamically equivalent when using (A.2) and when $R_f = l_f, R_r = l_r = 0.5l$. This is the distance used to place the accelerometers in the robot structure. Using these conditions and solving for the accelerations (with $a_c = a$), the equation (5) is obtained.

REFERENCES

[1] R. Siegwart and I. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Cambridge, MA, USA: MIT Press, 2004.

[2] I. Skog and P. Handel, "In-car positioning and navigation technologies—A survey," *IEEE Trans. Intell. Transport. Syst.*, vol. 10, no. 1, pp. 4–21, Mar. 2009.

[3] O. Michel, F. Rohrer, N. Heiniger, and wikibooks contributors. (2009). *Cyberbotics' Robot Curriculum*. Cyberbotics Ltd. and Wikibooks contributors, [Online]. Available: http://en.wikibooks.org/wiki/Cyberbotics'_Robot_Curriculum

[4] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Trans. Robot. Autom.*, vol. 17, no. 3, pp. 229–241, Jun. 2001.

[5] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006.

[6] H. Chen, D. Sun, J. Yang, and J. Chen, "Localization for multirobot formations in indoor environment," *IEEE/ASME Trans. Mechatronics*, vol. 15, no. 4, pp. 561–574, Aug. 2010.

[7] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part II," *IEEE Robot. Autom. Mag.*, vol. 13, no. 3, pp. 108–117, Sep. 2006.

[8] C. Fuchs, N. Aschenbruck, P. Martini, and M. Wieneke, "Indoor tracking for mission critical scenarios: A survey," *Pervas. Mobile Comput.*, vol. 7, no. 1, pp. 1–15, 2011.

[9] N. Houshangi and F. Azizi, "Accurate mobile robot position determination using unscented Kalman filter," *Can. Conf. Electr. Comput. Eng.*, 2005, pp. 846–851.

[10] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice Using Matlab*. New York, NY, USA: Wiley, 2001.

[11] G. Welch and G. Bishop. (2007, Mar.). *An Introduction to the Kalman Filter*, Univ. North Carolina, Chapel Hill, NC, USA, [Online]. Available: http://www.cs.unc.edu/ welch/kalman

[12] S. Julier, J. Uhlmann, and H. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Trans. Autom. Control*, vol. 45, no. 3, pp. 477–482, Mar. 2000.

[13] D. Simon, *Optimal State Estimation: Kalman, H_∞, and Nonlinear Approaches*. New York, NY, USA: Wiley, 2006.

[14] J. H. Kotecha and P. M. Djuric, "Gaussian particle filtering," *IEEE Trans. Signal Process.*, vol. 51, no. 10, pp. 2592–2601, Oct. 2003.

[15] S.-B. Han, J.-H. Kim, and H. Myung, "Landmark-based particle localization algorithm for mobile robots with a fish-eye vision system," *IEEE/ASME Trans. Mechatronics*, to be published.

[16] S. Liu and D. Sun, "Minimizing energy consumption of wheeled mobile robots via optimal motion planning," *IEEE/ASME Trans. Mechatronics*, to be published.

[17] A. Martinelli and R. Siegwart, "Estimating the odometry error of a mobile robot during navigation," presented at the European Conf. Mobile Robots, Radziejowice, Poland, 2003.

[18] T. H. Cong, Y. J. Kim, and M.-T. Lim, "Hybrid extended Kalman filter-based localization with a highly accurate odometry model of a mobile robot," in *Proc. Int. Conf. Control, Autom. Syst.*, Oct. 2008, pp. 738–743.

[19] H. Chung, L. Ojeda, and J. Borenstein, "Accurate mobile robot dead-reckoning with a precision-calibrated fiber-optic gyroscope," *IEEE Trans. Robot. Autom.*, vol. 17, no. 1, pp. 80–84, Feb. 2001.

[20] J. Yi, H. Wang, J. Zhang, D. Song, S. Jayasuriya, and J. Liu, "Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation," *IEEE Trans. Robot.*, vol. 25, no. 5, pp. 1087–1097, Oct. 2009.

[21] D. Hyun, H. S. Yang, H.-S. Park, and H.-J. Kim, "Dead-reckoning sensor system and tracking algorithm for 3-D pipeline mapping," *Mechatronics*, vol. 20, no. 2, pp. 213–223, 2010.

[22] J. Kim, Y. Kim, and S. Kim, "An accurate localization for mobile robot using extended Kalman filter and sensor fusion," in *Proc. IEEE Int. Joint Conf. Neural Netw., (IEEE World Congr. Comput. Intell.)*, Jun. 2008, pp. 2928–2933.

[23] A. Valera, M. Weiss, M. Vallés, and J. L. Diez, "Bluetooth-networked trajectory control of autonomous vehicles," in *Proc. 8th IFAC Symp. Cost Orient. Autom.*, 2007, vol. 8, pp. 198–203.

[24] M. Boccadoro, F. Martinelli, and S. Pagnottelli, "Constrained and quantized Kalman filtering for an rfid robot localization problem," *Auton. Robots*, vol. 29, pp. 235–251, 2010.
[25] R. Madhavan and H. Durrant-Whyte, "Natural landmark-based autonomous vehicle navigation," *Robot. Auton. Syst.*, vol. 46, no. 2, pp. 79–95, 2004.
[26] Y. Yang and J. Farrell, "Magnetometer and differential carrier phase GPS-aided INS for advanced vehicle control," *IEEE Trans. Robot. Autom.*, vol. 19, no. 2, pp. 269–282, Apr. 2003.
[27] T. Zhang and X. Xu, "A new method of seamless land navigation for GPS/INS integrated system," *Measurement*, vol. 45, no. 4, pp. 691–701, 2012.
[28] D. Bevly, J. Ryu, and J. Gerdes, "Integrating ins sensors with GPS measurements for continuous estimation of vehicle sideslip, roll, and tire cornering stiffness," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 4, pp. 483–493, Dec. 2006.
[29] F. Aghili and A. Salerno, "Driftless 3-D attitude determination and positioning of mobile robots by integration of IMU with two RTK GPSS," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 1, pp. 21–31, Feb. 2013.
[30] D. M. Bevly and B. Parkinson, "Cascaded Kalman filters for accurate estimation of multiple biases, dead-reckoning navigation, and full state feedback control of ground vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 2, pp. 199–208, Mar. 2007.
[31] Z. Shen, J. Georgy, M. J. Korenberg, and A. Noureldin, "Low cost two dimension navigation using an augmented Kalman filter/fast orthogonal search module for the integration of reduced inertial sensor system and global positioning system," *Transp. Res., Emerg. Technol.*, vol. 19, no. 6, pp. 1111–1132, 2011.
[32] J. Wong, *Theory of Ground Vehicles*. New York, NY, USA: Wiley, 2008.
[33] R. Rajamani, *Vehicle Dynamics And Control* (Mechanical Engineering Series). New York, NY, USA: Springer, 2012. [Online]. Available: 10.1007/978-1-4614-1433-9.
[34] C. C. Ward and K. Iagnemma, "A dynamic-model-based wheel slip detector for mobile robots on outdoor terrain," *IEEE Trans. Robot.*, vol. 24, no. 4, pp. 821–831, Aug. 2008.
[35] I. Zohar, A. Ailon, and R. Rabinovici, "Mobile robot characterized by dynamic and kinematic equations and actuator dynamics: Trajectory tracking and related application," *Robot. Auton. Syst.*, vol. 59, no. 6, pp. 343–353, 2011.
[36] C. D. L. Cruz and R. Carelli, "Dynamic model based formation control and obstacle avoidance of multi-robot systems," *Robotica*, vol. 26, no. 3, pp. 345–356, 2008.
[37] A. Albagul, W. Martono, and R. Muhida, "Dynamic modelling and adaptive traction control for mobile robots," *Int. J. Adv. Robot. Syst.*, vol. 1, no. 3, pp. 149–154, 2005.
[38] S. Noga, "Kinematics and dynamics of some selected two-wheeled mobile robots," *Arch. Civil Mech. Eng.*, vol. 6, pp. 55–70, 2006.
[39] W. Chee, "Yaw rate estimation using two 1-axis accelerometers," in *Proc. Amer. Control Conf.*, 2005, pp. 423–428.
[40] H. A. Attia, "Dynamic model of multi-rigid-body systems based on particle dynamics with recursive approach," *J. Appl. Math.*, vol. 2005, pp. 365–382, 2005.
[41] K. O. Arras, "An introduction to error propagation: Derivation, meaning and examples of equation $C_y = F_x C_x F_x^T$," Auton. Syst. Lab, Inst. Robot. Syst., Swiss Federal Inst. Technol., Lausanne, Switzerland, Tech. Rep. EPFL-ASL-TR-98-01 R3, 1998.

Leonardo Marín (M'13) received the B.Eng. and Licentiate degrees in electrical engineering from the Universidad de Costa Rica, San José, Costa Rica, in 2006 and 2009, respectively, and the M.Sc. degree in automation and industrial computing from the Universitat Politècnica de València, Valencia, Spain, in 2011, where he is currently working toward the Ph.D. degree in the area of automation, robotics, and industrial computing.

His research interests include the localization, navigation, and control of resource-limited mobile robots using sensor fusion with Kalman filters, cooperative localization in heterogeneous groups of robots, and event-based estimation and communication systems.

Marina Vallés received the degree in computer science, the M.Sc. degree in CAD/CAM/CIM, and the Ph.D. degree in automatic control from the Universitat Politècnica de València (UPV), Valencia, Spain, in 1996, 1997, and 2004, respectively.

Since 1999, she has been a Lecturer in the Department of Systems Engineering and Control, UPV. She currently teaches courses on automatic control and computer simulation. Her research interests include real-time systems, automatic code generation, and control education.

Ángel Soriano received the B.Eng. and M.Sc. degrees in computer science from the Universitat Politècnica de València, Valencia, Spain, in 2010 and 2011, respectively, where he is currently working toward the Ph.D. degree in the area of mobile robot control, mechatronics, and multiagent systems.

His research interests include perception, control, coordination, and collaboration for mobile robots through multiagent systems.

Ángel Valera (M'10) received the B.Eng. and M.Sc. degrees in computer science, in 1988 and 1990, respectively, and the Ph.D. degree in control engineering in 1998, all from the Universitat Politècnica de València (UPV), Valencia, Spain.

He has been a Professor of automatic control with the UPV since 1989, where he is currently a Full Professor with the Department of Systems Engineering and Control. He has been involved in research and mobility projects funded by local industries, government, and the European community. He has authored or coauthored more than 150 technical papers in journals, technical conferences, and seminars. His current research interests include industrial and mobile robot control and mechatronics.

Pedro Albertos (M'81–SM'88) received the Doctoral (*Honoris-Causa*) degree from Oulu University, Oulu, Finland, and Bucharest Polytechnic, Bucharest, Rumania. He is currently a Full Professor in the Systems Engineering and Control Department, Universitat Politècnica de València, València, Spain. As an Invited Professor at more than 20 universities, he has delivered seminars at more than 30 universities and research centers. He has authored more than 300 papers, book chapters, and congress communications, and is a coeditor of seven books. He is a coauthor of *Multivariable Control Systems* (Springer, 2004) and *Feedback and Control for Everyone* (Springer, 2010). His research interests include multivariable control and nonconventional sampling control systems, with focus on time delays and multirate sampling patterns.

Dr. Albertos is a past President of the International Federation of Automatic Control (IFAC) (1999–2002), IFAC Fellow, and IFAC Advisor, is a world recognized expert in real-time control, leading several projects in the field. He is also an Associate Editor of *Control Engineering Practice* and Editor-in-Chief of the Spanish journal *RIAI*.