This document is published in:

2010 5th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC 2010). Netherlands, 8-10 December 2010. IEEE, pp. 579-586. DOI: http://dx.doi.org/10.1109/NAVITEC.2010.5708060

Analysis of a sensor fusion hybrid solution for indoor/outdoor robot navigation

E. Martí, J. García, J.M. Molina
Group of Applied Artificial Intelligence
Univ. Carlos III of Madrid
Colmenarejo, Spain
{emarti, jgherrer}@inf.uc3m.es, molina@ia.uc3m.es

***Abstract*—Autonomous mobile robots need robust, flexible and accurate navigation algorithms. One approach consists in fusing as many information sources as possible, integrating measures from internal sensors with data obtained from external sensing entities. This work presents a solution for combined indoor/outdoor robot navigation, and analyzes some preliminary results in an outdoor environment using a Particle Filter for GPS/INS sensor fusion. Experiments are based in predesigned trajectories which have been simulated in first place and then reproduced using a robotic platform. As a concluding remark, some considerations about the use of Particle Filters and the differences between simulated and real data are presented.**

Keywords-component; particle filters, robot navigation, indoor/outdoor navigation

I. INTRODUCTION

This work aims to introduce a simple and robust architecture for combined indoor/outdoor navigation through sensor fusion technology, where the information provided by on-board sensors is aligned with external references [1]. Several positioning technologies (including GPS, Ultra Wide Band sensors and external video-based trackers) will help to avoid the cumulative drifts caused by locally referenced information, so that the navigation is coherent with the rest of entities in the environment –fixed obstacles, other moving objects, etc.

Due to the non-linear relation of the inertial sensors with absolute references, a Sampling Importance Resampling (SIR) Particle Filter (PF) [2] explored in previous simulated experiments is applied to estimate the navigation solution, The state to be estimated includes the location of the robot with respect to absolute references, its attitude and kinematics.

After describing the proposal at high level, part of the system depicted to outdoor navigation will be implemented and tested using an Inertial Measure Unit (IMU) and a GPS device. The system capabilities will be tested through software simulation and validated using a real platform. This solution will be complemented with additional mechanisms to increase robustness, as detecting particle population degeneracy which can cause the filter to diverge.

The available sensors are a MicroStrain® 3DM-GX2™ IMU and a Novatel® OEMV GPS card. The IMU integrates tri-axial accelerometer, gyroscope and magnetometer, as well as a temperature sensor. Novatel product provides differential GPS positioning compatible with Satellite Based Augmentation Systems (SBAS) as EGNOS.

After detailing sensor specifications, its measurement model will be mathematically described. The obtained models will be implemented in a software simulation system based on MATLAB® Aerosim Aeronautical Simulation Block Set [3]. This will allow us to test the solution and evaluate its performance systematically with a set of representative scenarios, and propose modifications before integrating it in the hardware platform. The simulation analysis includes also the application of different metrics to evaluate the expected quality of the real navigation algorithm. After this simulated experimental section, real sensor data will be used to validate the solution and the applicability of the statistical models assumed. The robot, equipped with the sensors mentioned above, will be manually controlled in a number of controlled experiments to obtain sensor data in representative situations. The results of this part can also be used to approximate the optimal value for some configuration parameters. Finally, the extension of developed sensor fusion system with other available positioning tools such as indoor localization services can be considered as a future work to increase the navigation capabilities.

I. PROPOSED SYSTEM

The platform target is a single autonomous robot, so a centralized fusion algorithm is the best option for integrating all the available knowledge. In order to increase its robustness and boost the obtainable accuracy as far as possible, it has to be capable of incorporating different types of information from heterogeneous sources, even external entities.

A. Particle Filter

The most common filtering algorithms use Bayesian inference to estimate the state of a partially observed system. The uncertainty about the real state makes necessary to store the belief as a probability distribution, so that at each time the filter can estimate which is the most probable state according to the available information.

This probability distribution changes with time. It can be adapted using a prediction model that describes system dynamics, and incorporating some occasional measurements providing information about the real state.

Some techniques, as the Kalman Filter (KF) or the Extended Kalman Filter (EKF) [4] assume that all uncertainties have Gaussian distribution, and store the state probability distribution as another Gaussian. Thanks to that, all the mentioned elements can be compactly described as a "mean value" vector and a covariance matrix. Thanks to that simplification, they obtain a matrix-based analytical solution that can be calculated fast (and is optimal if the assumptions are true).

Nonetheless, if system dynamics obey a highly non-linear model or uncertainties are far from being Gaussian, then these techniques deliver poor performance. A PF is a Monte Carlo algorithm capable of dealing with such non-linear non-gaussian scenarios (see section III, Particle filter definition). Implementation simplicity is another great value of Particle Filters: it is sufficient to provide mathematical formulations for (a) system dynamics, (b) sensor models, and (c) process and measure noises. The specifications of (c) must support the generation of random samples identically distributed to them.

B. Loosely-coupled, centralized Sensor Fusion

There are many important concepts in the search of robustness. Amongst them, we can enumerate redundancy, failure tolerance and adaptability. For these and other reasons, a central loosely-coupled Particle Filter has been selected as fusion system.

Loose coupling enables the possibility of not having hard dependencies on individual sensors, or on combinations of such devices. This is of the uttermost importance when sensors have limited availability, such are the cases of GPS in mixed indoor/outdoor environments and fixed external sensors when the robot is in constant movement.

The centralized approach is convenient to the initial system purposes. Its implementation is simpler than in the case of distributed systems, and its accuracy either similar or superior [5]. Pure centralized fusion systems have the disadvantage of a communication bottleneck around central node. This is not our case, since the scale of the target system maintains the number of involved elements inside reasonable bounds.

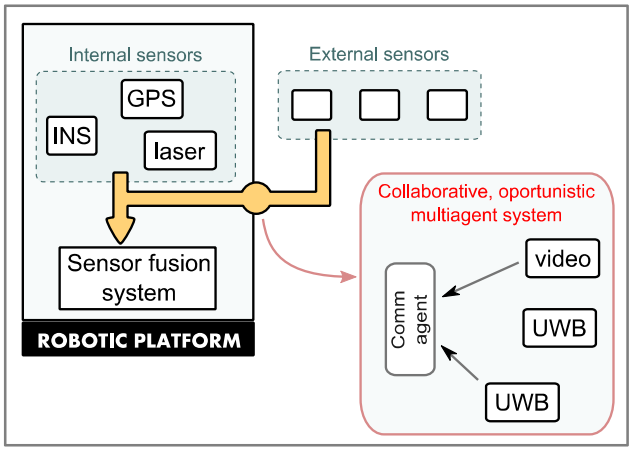

Figure 1. Fusion system is event driven; it acts when new sensor readings arrive. An independent software agent is in charge of discovering and managing external sensors, making the process transparent to the central fusion system.

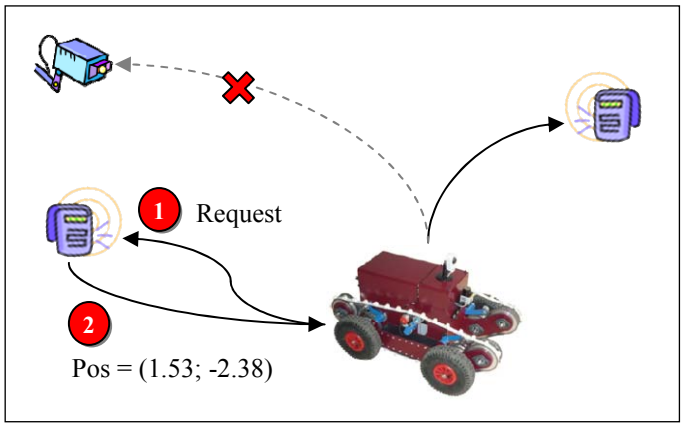

Figure 2. A multi-agent approach can be used to create a collaborative, intelligent environment. A software agent in the robotic platform manage all communication tasks. The final target is to all the issues not directly related with sensor measures transparent to central fusion system.

Combining the reviewed elements, we have what is shown in Figure 1. The sensor fusion system (i.e., the PF) integrates the information coming from different devices as it is received. To accomplish this task, the fusion system must see all sensors as sources of observations about the state vector.

C. Interaction with the environment

The proposed navigation system can be extended to include external sensors. We define external sensors as entities not belonging to the robotic platform that can provide some services useful for navigation, such as absolute positioning. This project aims to integrate at least UltraWide Band (UWB) and video-based tracking.

The availability of external devices is not guaranteed: for instance, the robot can modify its position leaving some sensors out of reach. Or the external sensors, as independent entities, can halt their activity without notice.

At this point we have two inconveniences: the availability, and the requirement of a homogeneous view of all sensors. To deal with this problem the environment will be modeled as a Multi-Agent System. A software agent in the robotic platform will work managing external sensors (discovery, communication, out-of-service deletion), and adapting the obtained data to the needs of the sensor fusion node. External sensors will also be software agents that respond to information requests.

All together configures a collaborative multi agent system where the information sources are passive, and the sinks (i.e. the robotic platform) act following an opportunistic strategy. Apart from satisfying the criteria of the proposed system, the multi-agent architecture does not restrict external sensors to help in robot navigation system; they can perform other tasks simultaneously.

II. ROBOTIC PLATFORM

Our test platform is a GUARDIAN rover from Robotnik corporation. It features a wide range of sensors, including but not limited to odometry, laser ranging, inertial navigation and a video camera. The unit is equipped with an embedded

computer for high-demanding computing tasks and integrating its hardware control through the Player/Stage architecture [6].

A. Sensors used in this work

The experiments of this work are based in an Inertial Measure Unit and a GPS device. The first one is a InertiaLink 3DM-GX2 unit containing triaxial accelerometer, gyroscope and magnetometer (see TABLE I.). Its hardware implements a Complementary Filter [7] for stabilizing the noisy sensor readings and obtaining a better estimate of real magnitudes.

The unit was mounted near electronic components. This introduces a sustained magnetic interference affecting the magnetometer, and so the Complementary Filter stabilized estimation for acceleration and angular rate.

TABLE I. IMU TECHNICAL DETAILS

	Range	***Bias stability***	***Nonlinearity***
Accelerometer	10 g	0.01 g	0.2 %
Gyroscope	300 deg/sec	0.2 deg/sec	0.2 %
Magnetometer	1.2 Gauss	0.01 Gauss	0.4 %

For the global positioning, a Novatel® OEMV-1G differential GPS will be used. It is compatible with Satellite Based Augmentation Systems as EGNOS.

According to the technical specification, horizontal position can be measured with an accuracy of 1.5 m (RMS) when operating on single point L1 mode.

III. PARTICLE FILTER DEFINITION

The selected filtering algorithm is a Sampling Importance Resampling Particle Filter. Given a set of measures up to present time (1), it tries to estimate the probability distribution of the state x (2), also called "posterior probability distribution" or just "posterior".

$$Y_t = \{y_1, y_2, \ldots, y_t\} \quad (1)$$

$$P(x_t|Y_t) \quad (2)$$

Particle filters describe the above probability distribution using a population of N individual samples also called particles, where each one is assigned an importance or "weight" (w) measuring how likely it represents the true state (3), so that an estimate of this state can be obtained by an weighted average of particle population (4).

$$P(x_t|Y_t) \sim \{x_t^i, w_t^i\}_{i=1:N} \quad (3)$$

$$E(x_t) = \sum_i x_t^i \cdot w_t^i \quad (4)$$

Filtering is performed in two different steps: prediction and update. Prediction applies a mathematical model about state dynamics to the particles, to represent the evolution of the system in time. The prediction step can include some information about how the dynamic model has to be applied; this data is often known as control input (5). An example of control inputs are the readings of an IMU, which provide information about dynamics that cannot be inferred from previous state or the prediction model.

$$x_{t+1} = f(x_t, u_t) \quad (5)$$

The uncertainty about real state before predicting is augmented proportionally to the uncertainty associated to prediction model and control inputs. This means that prediction phase increase the variance of the particles in the state space.

The second step is known as update, and is applied when a new observation of system state is available. During the update phase, a measure y_t and its uncertainty model is used to estimate the quality of each particle, and modify their weights:

$$w_{t+1} = w_t \cdot p(y_t \mid h(x_t)) \quad (6)$$

Where $h(x_t)$ is the measure model: a mathematical function that express how to calculate the measure of the selected sensor from a given state.

Update reduces uncertainty thanks to the newly acquired information.

Using and infinite number of particles we obtain a perfect representation of the state probability distribution, but we only have a finite number of them. The problem with this approach is, thus, that almost all particles will eventually be far from real state and their weights will be infinitesimal. The estimation of the filter is in these cases based on a few samples, reducing its overall quality.

To avoid this situation, each several update steps the filter goes through a resampling stage. This process consists in deleting particles with too low weights and multiplying the more promising ones in a number proportional to their weight. As a result, the filter represents real state probability distribution more compactly, and with more particles –i.e. more detail– in zones with a high probability.

Using a PF, the navigation problem is reduced to defining a state vector, as many prediction models as types of control inputs available, and an update model for each kind of measure. Next, the specification for the INS/GPS combination is described.

A. State vector

The state vector for this problem has to represent at least bidimensional position, speed and orientation. The final configuration has 5 dimensions:

$$x = [px; py; vx; vy; \alpha]^T \quad (7)$$

Variables px, py represent position in Cartesian coordinates. The model uses the flat world assumption because it makes easier to enable the mixed indoor/outdoor navigation, and is enough in outdoor environments given the reduced reach of the robot.

Although the robot is supposed to move only along its longitudinal axis we include in the model two degrees of

freedom for speed. This is translated in the global speed variables vx, vy.

Finally, the attitude of the robot is expressed as a single yaw angle α. The tests take place in flat environments, so the possibility of including pitch and roll angles in the state is left for future work.

B. Prediction model

Our model performs Euler-like time integration using inertial measures as control input. The position, speed and body orientation are calculated using the inertial prediction model :

$$x_{t+\Delta t} = [p_x, p_y, v_x, v_y, \alpha]^T = f_{IMU}(x_t, [a_x, a_y, \omega]) \quad (8)$$

$$p(t+ \Delta t) = p(t) + v(t)\cdot\Delta t + ½\cdot a(t)\cdot \Delta t^2 \quad (9)$$

$$v(t+ \Delta t) = v(t) + a(t) \cdot C_{b2n} \cdot \Delta t \quad (10)$$

$$\alpha(t+ \Delta t) = \alpha(t) + \omega (t) \quad (11)$$

Where p is position, v is speed, α is the orientation angle, a/ω are the accelerometer/gyro inputs, and C_{b2n} in (10) is the body-to-navigation rotation matrix created from α(t) value.

In spite of its simplicity, this model has proved to be accurate enough for the small time steps to integrate and the smooth dynamics of selected trajectories.

C. Update model

The last step in the PF creation is to define the error model of the positioning sensor. For these experiments the GPS is assumed to give 3D measures in Cartesian coordinates, with an independent error component in each axis following a Gaussian distribution (standard deviation 0.7 meters). Only two coordinates will be used for updating the weights.

When receiving a GPS measure $y_t = [p_x; p_y]$, the so called measure model has to be applied to each particle (12). Then, the likelihood of the actual measure can be checked against every sample to update their weight.

$$y_t^i = h(x_t^i) = [p_x^i; p_y^i]^T \quad (12)$$

$$w_{t+1}^i = w_t^i \cdot p(y_t|y_t^i) \quad (13)$$

D. Improvements

In spite of being just a proof of concept, the implemented PF includes some additional techniques to improve its behavior and making it usable in real conditions. This section is depicted to justify and describe them.

1) Divergence detection + reinitialization

Every robust navigation system should avoid scenarios of uncontrolled performance degradation. Part of this preventive behavior consists in monitoring its (estimated) performance, so that it can act accordingly to the situation.

Reference [8] suggests reinitializing particle filters when the population is too far the real posterior. Their reasoning is that recovering normal function by heavily increasing plant noise can lead to "oscillations" of the prediction around the real state, or may even not have any effect and result in a total divergence of the filter.

If the likelihood of the received GPS measures compared with PF estate estimation is consistently low for a few consecutive cycles, then the filter is considered to be diverged. Current particle population is discarded, and a new one is created using the two last direct observations of the state.

2) Adaptive noise

The noise applied during resampling stage is modified by a factor M_t which depends on estimation quality. The concept has been taken and adapted from [9], where it is applied to video surveillance.

A similarity measure φ_t has to be calculated between predicted state and the last received measure at each update step. This similarity is defined as the likelihood of the measure with respect to filter prediction.

In the concrete case of using a GPS sensor, we have that the measure likelihood follows a multivariate Gaussian distribution with covariance matrix Σ_{gps}. The similarity of state x and measure z involves computing a Mahalanobis distance using the aforementioned covariance:

$$z' = h(x) \quad (14)$$

$$\varphi_t = \exp(0.5 \cdot [(z'-z)^T \cdot \Sigma_{gps}^{-1} \cdot (z'-z)]) \quad (15)$$

The result is a factor that will be used to scale the original noise. This factor is upper-bounded to avoid the side effects of introducing an excessive perturbation to particles:

$$M_t = \min(\sqrt{(1/\varphi_t)}, M_{max}) \quad (16)$$

The final value multiplies the original covariance matrix for plant noise (w) used in our system:

$$\Sigma_w(t) = M_t . \Sigma_w(0) \quad (17)$$

3) Particle repropagation on resampling

Resampling stage involves multiplying the most well fitted particles and dropping the worst from the population. After that, a random sample drawn from plant noise is added to each new particle with the purpose of covering better the posterior state space.

The problem with this approach is that, in the true posterior, some dimensions of the state are influenced by others and are thus correlated (in our case, position is affected by speed, and speed is influenced by previous attitude because it affects how acceleration is applied). Adding samples of pure random noise have the problem of creating particles that are bad candidates for representing the true state.

Particle resampling quality can be improved by using a simple algorithm: particle repropagation. With this scheme, particles are selected for resampling as usual, but instead of reproducing the actual particles we will used a snapshot of them in a past time step.

The "particles in the past" are perturbed with noise affecting just first-order dimensions, this is, those variables that are not affected by others in the state vector. The resulting population is carried to the present using the in-between control inputs and prediction model.

Repropagation reduces the number of particles required to filter a trajectory, and can boost slightly the maximum accuracy obtainable by the filter.

IV. SELECTED TRAJECTORY DATASET

Three trajectories were designed for the first test of the platform. They have been created to be simple, yet allowing to test all the features of the system.

As the robot was operated on manual control and proximity sensors were not included in the specification, the trajectories do not involve obstacle avoidance.

A. *Straight line*

A 24 meter long straight line. The robot starts at one end, and after a few seconds stopped it travels the whole path at non-constant speed.

At the end of the path, the platform is stopped for approximately 3 seconds, performs and stationary turn and goes back to the starting point. Then it turns again and goes to the ending mark. The whole process takes 80-90 seconds.

It is intended to be an easy scenario, though the stops and stationary turns can cause problems to those filters that have not obtained accurate speed estimations.

B. *Stadium*

The second trajectory, shown in Figure 3. , is travelled at constant speed in near 40 seconds. It includes two turns with the shape of a semicircle, so that the angular rate is constant through them.

This trajectory is used to test the average performance of our solution in normal conditions, as it has a number of features in common with usual trajectories both indoor and outdoor.

C. *Circumference*

The last trajectory has circular shape, and is traveled several times. The robot takes 10-15 seconds for covering a lap, and just a single GPS reading is provided for each complete round (period of about 10 seconds). This experiment is intended to test the quality of pure inertial navigation under different conditions, as using regular or bias-corrected measures, or applying enlarged/reduced inertial noises (in simulations).

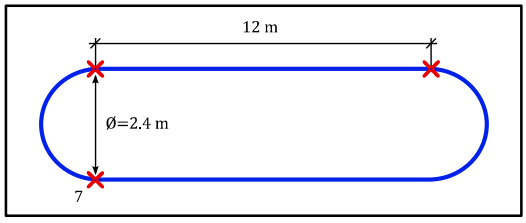

Figure 3. Stadium trajectory

V. SIMULATION

The robotic platform is simulated as a rigid body with three degrees of freedom (3DoF in advance): two degrees for translation (horizontal movement), and one for rotation.

We have developed a basic simulation process based on MATLAB™. This framework provides a complete set of tools for rapid development of detailed 6DoF nonlinear models, though it can also generate 3DoF trajectories by properly defining the forces and angular moments in the body frame of the vehicle.

The simulator generates the calculated values for position, speed, orientation, acceleration and angular rates. These reference values are used to generate simulated measures for the selected sensors. The next paragraphs present an overview of the measure simulation process for GPS and inertial sensors.

A. *GPS sensor model*

Our simulation uses a flat-earth model. Taking as input the reference 2D position, each coordinate is independently altered by adding a random sample drawn from a zero-mean Gaussian distribution.

B. *IMU sensors model*

In the selected IMU, both accelerometer and gyroscope are triaxial: each measure is a 3D vector with the value of the measured magnitude in the orthogonal axes of the device local coordinate system (uses North-East-Down convention).

Our simulation of IMU raw measurements considers two types of errors: random noise with Gaussian distribution and a constant bias for each axis.

Typical biases are short-term stable but can drift under temperature changes; 3DM-GX2 model is temperature compensated. The only noticeable effect is a sudden bias change between runs, in spite that those values are stable along each single simulation. Figure 4. illustrates the generated accelerometer magnitudes corresponding to a simulated stadium trajectory.

The model was initially prepared to introduce a cross-coupling effect caused by sensor axes misalignment, but the real device lacks this problem thanks to vendor initial calibration.

Comparing Figure 4. and Figure 5. , we can see how the noise in accelerometer measures is larger than expected in the real experiment. This is caused by a poor mounting that do not absorb motion vibrations, and will be direct cause of most of the problems found in experimentation.

As an additional note, the dynamic margin of biases in the performed experiments is superior to that stated in the technical datasheet, reaching 0.25 m/s^2 in the case of the accelerometer – that is, 0.025 g– and 0.5 deg/sec in the gyroscope.

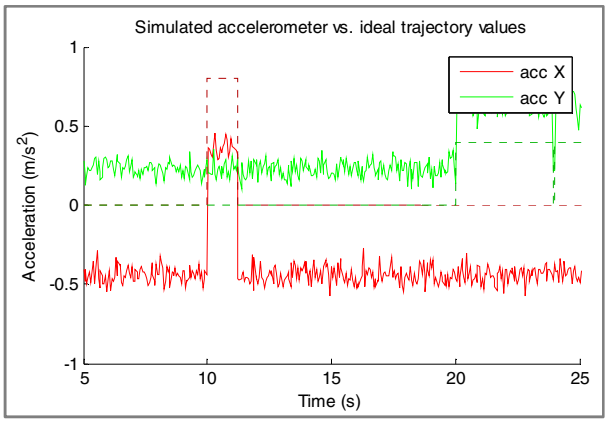

Figure 4. Fragment of simulated accelerometer reading for the stadium trajectory. Ideal values appear as dotted lines, to show bias magnitude

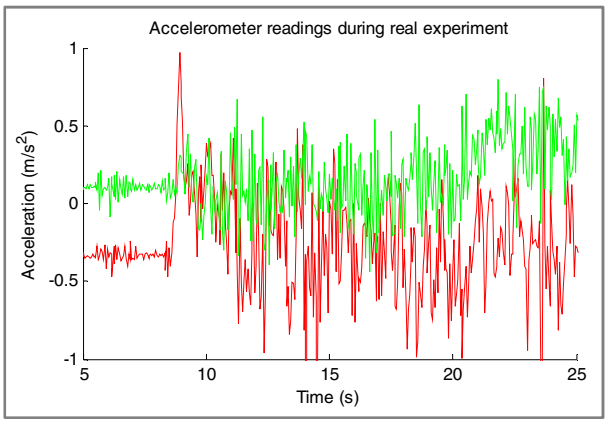

Figure 5. Fragment of real accelerometer readings for the stadium trajectory. Noise during robot motion is an order of magnitude above that shown when stopped

VI. RESULTS

This section presents the results of executing the proposed experiments with simulated and real trajectories. Special attention will be put in unexpected behaviors and other unusual effects, to help refining the system for future developments.

A. *Straight line*

This scenario is very simple, so it has been tested using a lower frequency for GPS measures (period of approximately 5 seconds) to make it harder. A PF can obtain good estimates for positions and speeds during the first run, but the total stop followed by a static turn is difficult to filter properly. Figure 6. shows a run over a real-data trajectory where the plant noise has been set to the IMU specifications. Real noise is several times superior during movement (see Figure 7.), but this does not affect the PF until the static turn at the end of the path.

When the plant noise is set to mimic the real running conditions, the filtering becomes less accurate because the state variance is greater, but its behavior is more robust. We can expect a mean positioning error of about 60 cm for 1 m precision GPS measures. More elaborate schemes and better techniques as Rao-Blackwellized Kalman Filter [10] or Unscented Particle Filter [11] are known to be more effective than plain PF, but the obtained performance is enough for these first tests.

B. *Stadium*

The stadium provides a general view of the overall filter performance, thanks to its combination of turns, accelerations and straight fragments. Receiving GPS measures at 1 Hz, the PF can easily obtain an average 40% improvement over bare observations, as shown in Figure 8.

Further tests with no bias correction (Figure 9.) show that an artificially increased plant noise makes possible to obtain similar position accuracy thanks to the high GPS update rate.

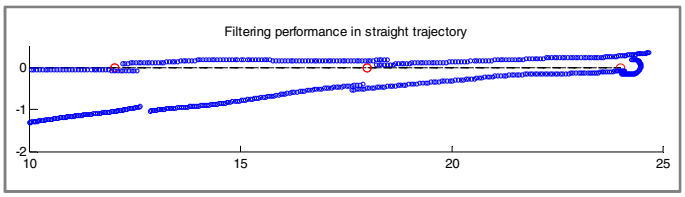

Figure 6. Using a reduced plant noise it is possible to obtiain a great prediction accuracy, but some maneuvers can lead to a population divergence (the filter has to be restarted)

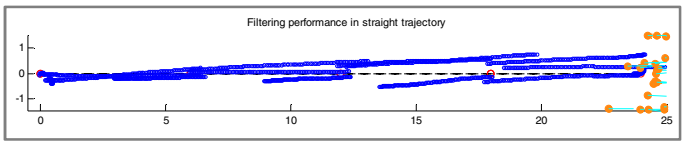

Figure 7. Increasing plant noise offers lower accuracy in positioning, but makes the filter more robust against unusual measures

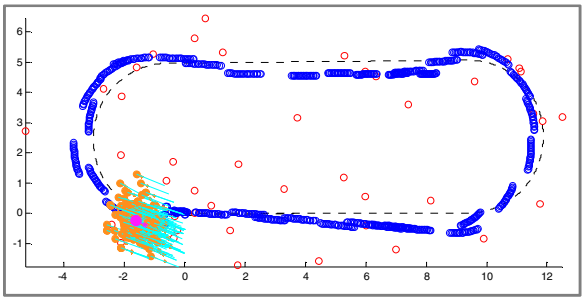

Figure 8. Using bias-corrected inputs, the PF improves GPS accuracy an average 40% (from meter-precision to 0.6 m).

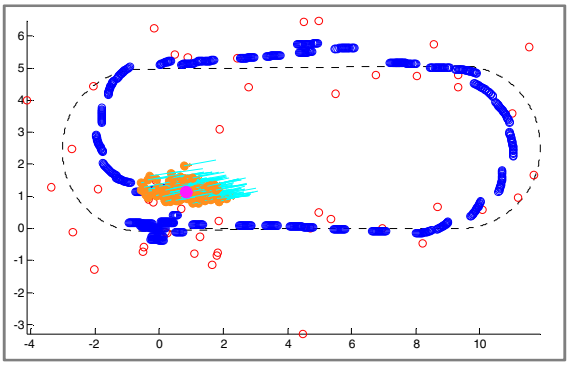

Figure 9. When the biases are not corrected speed estimation is severely affected, though positions have a reasonable accuracy –still better than bare GPS measures.

When using low-frequency GPS measures, biases inside the expected dynamic margin of the IMU will cause the filter to diverge.

C. Circumference

The last test has brought several interesting conclusions about using low-accuracy positioning in reduced spaces. Figure 10. describes the result of filtering the circumference trajectory (10 turns) using exclusively inertial information. It describes a smooth trajectory with variable accuracy –not always decreasing–. The uncertainty of the filter grows with time but the mean of the particles remains stable.

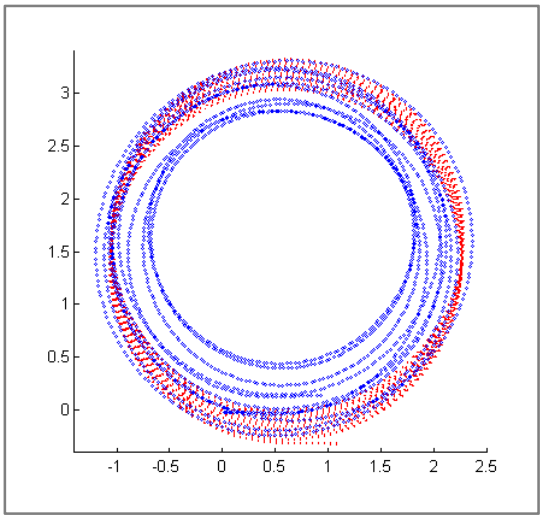

Figure 10. Inertial based filtering. The predicted trajectory describes quasi-circles of changing size. This effect is caused by the large noise of inertial measures incorporated to the particles

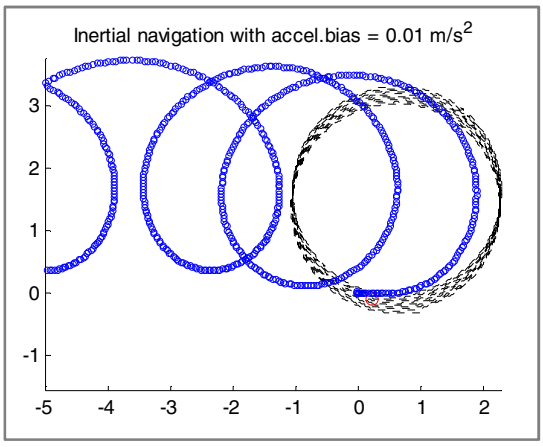

Figure 11. A residual bias in measured accelerations can be catastrophic, even with values as low as a tenth of the expected dynamic stability. The problem has the same magnitude under gyroscope bias.

When GPS measures are provided with a frequency below 1Hz, the errors are always larger. The problem is related with the trajectory being contained in a reduced space: the mere positions are not useful by itself since they are very uncertain, and particles representing very different states can end in similar positions after a whole lap. It is interesting to see that the filtering results on this trajectory are equally bad for unbiased and biased inertial measures.

VII. Conclusions and Future Work

The obtained results validate the selected model for general navigation and the proposed architecture for outdoor use. The system can keep track of the robotic platform using as few as two different types of sensors (with only one for updating the Particle Filter).

The combination GPS/INS can be used to achieve reasonable accuracy in flat outdoors. In spite of the reduced dimensionality of state vector, low speeds and reduced spaces can be a problem if inertial biases are not corrected or positioning sensors are not accurate. Under these conditions, measures can be not well suited for rewarding particles adequately.

Using real data for the experiments have demonstrated that simulations are not suitable as a standalone benchmarking tool. Differences between technical specification and real device performance make the last phase mandatory. In order to guarantee stability of the PF under the harrowing conditions of real data, positioning measures must have a period close to one measure per second when biases have been not corrected, and inferior to 3-4 seconds under normal conditions. Nonetheless, the performance can be improved by using more types of sensor or by solving some hardware issues as the magnetic isolation and vibration suppression for IMU.

The general recommendation is to integrate other sensors to provide direct observations for different dimensions of the state vector. Magnetometer and odometers are perfect examples, as their measures complement the information provided by a GPS. They have been combined successfully in other works [12] to achieve exceptional accuracy levels.

Simulated and real data yield comparable results in most of the cases. The problem of increased noise can be mitigated by raising the number of particles, which is never required to be over a few hundred samples.

The plain Particle Filter used in this first working version is not capable of estimating IMU biases properly. We have found that they can be corrected using simpler techniques when the robot is stopped for brief time lapses, although moving to more advanced techniques such as [13] or [14] is recommendable.

Acknowledgment

This work was supported in part by Projects ATLANTIDA, CICYT TIN2008-06742-C02-02/TSI, CICYT TEC2008-06732-C02-02/TEC, SINPROB, CAM MADRINET S-0505/TIC/0255 DPS2008-07029-C02-02.

References

[1] M.E. Cannon, R. Nayak, G. Lachapelle, O.S. Salychev, and V.V. Voronov, “Low-Cost INS/GPS Integration: Concepts and Testing,” *The Journal of Navigation*, vol. 54, Jan. 2001, pp. 119-134.

[2] N.J. Gordon, D.J. Salmond, and A.F.M. Smith, “Novel approach to nonlinear/non-Gaussian Bayesian state estimation,” *Radar and Signal Processing, IEE Proceedings F*, vol. 140, 1993, pp. 107-113.

[3] “User’s Guide (Aerospace Blockset™).”

[4] G. Welch and G. Bishop, “An Introduction to the Kalman Filter,” 1995.

[5] R. Olfati-saber, “Distributed Kalman filtering and sensor fusion in sensor networks,” 2006, pp. 157-167.

[6] B.P. Gerkey, R.T. Vaughan, and A. Howard, "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems," 2003, pp. 317-323.

[7] M. Zimmerman and W. Sulzer, "High bandwidth orientation measurement and control ased on complementary filtering," *Proceedings of Symposium on Robotics and Control, SYROCO*, Viena: 1991, pp. 525-530.

[8] B. Turgut and R.P. Martin, "Restarting Particle Filters: An Approach to Improve the Performance of Dynamic Indoor Localization," *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, Ieee, 2009, pp. 1-7.

[9] X. Xu and B. Li, "Rao-Blackwellised Particle Filter with Adaptive System Noise and its Evaluation for Tracking in Surveillance."

[10] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, 2002, pp. 425-437.

[11] R.V.D. Merwe, N. de Freitas, A. Doucet, and E. Wan, "The Unscented Particle Filter."

[12] Y. Cheng and J.L. Crassidis, "Particle Filtering for Attitude Estimation Using a Minimal Local-Error Representation," *Journal of guidance, control, and dynamics*, vol. 33, pp. 1305-1310.

[13] X. Yang, K. Shi, and T. Huang, "Combined Parameter and State Estimation in Particle Filtering," *2007 IEEE International Conference on Control and Automation*, vol. 00, May. 2007, pp. 1036-1039.

[14] J.H. Gove and D.Y. Hollinger, "Application of a dual unscented Kalman filter for simultaneous state and parameter estimation in problems of surface-atmosphere exchange," *Journal of Geophysical Research*, vol. 111, Apr. 2006.