INS/GPS fusion algorithm design in a fast prototyping environment: towards a real-time implementation

Richard Giroux, Ph.D.[1]
Former graduate student [2]
École de technologie supérieure
giroux_richard@videotron.ca

Richard Gourdeau, Ph.D.
Professor
École Polytechnique de Montréal
richard.gourdeau@polymtl.ca

René Jr. Landry, Ph.D.
Professor
École de technologie supérieure
rlandry@ele.etsmtl.ca

Abstract

In previous communications, the authors have highlighted the advantages of using a fast-prototyping approach in the development of low-cost inertial navigation integration algorithms. More specifically, they have addressed two fundamental aspects of inertial navigation integration algorithm design: the validation of a Simulink simulator and the performance evaluation of the integration algorithms provided within Simulink for inertial data integration.

This paper addresses the next step in the development sequence of INS/GPS integration algorithm development: the inclusion of an extended Kalman filter in the Simulink fast-prototyping environment, the evaluation of the computation load of the different parts of the algorithm, and the experimental assessment of the entire fusion algorithm. As a general conclusion, the rapid-prototyping approach chosen in the implementation has permitted the design of the algorithms and data acquisition scheme in an efficient manner and in a short development time period.

Résumé

Dans une publication précédente, les auteurs ont démontré les avantages d'utiliser une approche de prototypage rapide dans le développement d'algorithmes pour les systèmes inertiels à bas coûts hybridés avec un récepteur GPS. De façon plus spécifiques, ils ont discuté de deux aspects fondamentaux de la conception des algorithmes d'intégration de systèmes inertiels: la validation d'un simulateur conçu dans l'environnement Simulink; et l'évaluation de la performance des méthodes d'intégration numérique disponibles dans Simulink pour l'intégration numérique des capteurs inertiels.

La présente publication détaille la prochaine étape dans le développement d'un algorithme complet d'un système inertiel hybridé avec un récepteur GPS: l'inclusion d'un filtre de Kalman généralisé dans l'environnement de prototypage rapide, l'évaluation de la capacité de traitement d'une telle architecture de fusion de données, ainsi que la mise en œuvre expérimentale de l'algorithme. Comme conclusion générale, l'approche par prototypage rapide utilisée a permis la conception d'un algorithme de fusion SNI/GPS et d'acquisition de données inertielles de façon efficace et en minimisant le temps de conception.

Nomenclature

$[{}^{I}\vec{a}_B]_B$	Accelerometer data;
${}^{N}C_B$	Direction cosine matrix;
${}^{E}C_N$	Position matrix (lat-long);
h	Altitude;
$\mathcal{H}(\,\cdot\,)$	The function mapping the set of variables to the external measurements;
K_k	Matrix weight of correction of the a priori estimated external measurements (Kalman gain);
$\mathcal{U}_k$	The inputs to the system;
$[{}^{E}\vec{v}_N]_N$	Velocity (East-North-Up);
$\widehat{\mathcal{X}}_{k\|k}$	The estimate of the variables at time instant *k*, a posteriori the external measurement (knowledge of the external signal at time *k*);
$\widehat{\mathcal{X}}_{k\|k-1}$	The estimate of the variables at time instant *k*, a priori the external

1. The research work associated to this publication has been performed during the Ph.D. studies of the principal author. Part of this research has also been done at the Australian Centre for Field Robotics (ACFR), The University of Sydney, Australia.
2. Since October 2004, the main author is a research fellow at the Canadian Space Agency John H. Chapman Space Centre, St-Hubert, Canada.

	measurement (knowledge of the external signal at time *k*-1 only);
$\mathcal{Y}_k$	The external measurements;
γ_k	The innovation, representing the error between the external measurements and the estimation of the external measurement based on the a priori estimation of the variables;
$[{}^E\vec{\delta v}_N]_N$	Velocity error;
$[{}^E\vec{\delta r}_N]_N$	Position error;
$\Phi_k(\,\cdot\,)$	State transition matrix;
$\vec{\psi}_N$	Attitude error;
$[{}^I\vec{\omega}_B]_B$	Rate gyros data;

Introduction

The design of low-cost inertial navigation system (INS) fused with GPS generates great interest since many years now. Many approaches have been investigated in order to mitigate the poor performance of low cost sensors, but few people have addressed the issue of the design process itself.

Simulation of aided-INS systems is mandatory prior to real implementation in order to validate the design. Furthermore, numerical analysis of algorithms behavior is necessary since the highly non-linear equations governing the system prohibit extensive analytical analysis. If we exclude proprietary simulators, commercial simulation package tools available until now to achieve this goal are Matlab script files (www.gpsoftnav.com). However, modular and easy graphical design allowed by Simulink incited us to create a simulator in this user-friendly environment. Also, it permits rapid real-time testing, which is of great interest for the physical implementation of low-cost system. To our knowledge, only Eck et al. (2001) have presented a simulator that uses Simulink for INS/GPS algorithm design. However, no detail on the algorithm processing performance is given.

It has been already shown that Simulink is an interesting tool for the integration of the equations of motion (Giroux et al., 2003). However, the implementation of an extended Kalman filter (EKF) for INS/GPS fusion in a graphical coding environment is not a trivial task. This follow-up paper highlights the design and implementation phases of the fusion algorithm, and its assessment with real data.

The first section reviews several solutions to the problem of INS/GPS fusion. Then, the architecture of the Simulink-based fusion algorithm is presented. The flow of information between functions of the EKF is also described. An important aspect of this paper relates to the computation effort required to process the EKF algorithm, and to some extent the entire INS/GPS fusion algorithms. Following that, practical implementation is shown and experimental results are given which show the advantages of the fast-prototyping approach. The last part of the paper is dedicated to some discussions and conclusions.

Review of INS/GPS fusion schemes

It is well-known that the solution of the equations of motion (direct integration) is not accurate over time because the error in the solution grows due to the following (Garg et al., 1978):

- Sensor errors;
- Inaccurate initial value of the solution;
- Inexact gravity acceleration model;
- Finite computation capabilities of numerical computers.

Modern-day computer tends to mitigate the latest issue and permits the use of high-order integration methods, as it has been highlighted in a previous paper of Giroux et al. (2003). Also, the error in the initial value of the solution can be minimized given an appropriate initializing time for the system. On the other hand, the gravity model can be fine tuned to represent in a more accurate manner the actual gravity field (Kriegsman et al., 1986). This is of importance for highly accurate INS systems which use inertial-grade sensors and no external sensors (or very few external measurements). However, at the other end of the spectrum, low-cost INS system performance is mainly driven by the sensor errors which make the equation of motion solution drifting very rapidly. Hence, a fusion scheme with external sensors is mandatory to ensure proper performance.

There are different approaches to perform the fusion of external information with the solution of the equations of motion (Anderson, 1999). The most practical algorithm is the Kalman filter implemented in an extended form to accommodate the non-linear behavior of the equations of motion. The filter itself can be implemented using different strategies: direct, indirect feedforward and indirect feedback implementations (Maybeck, 1994); and they will be described in the following sections.

But first, the definition of some mathematical expressions used throughout the paper is required.

To simplify the mathematical description of the equations at stake, a navigation variables set $\mathcal{N}$ is defined by Equation 1 and comprises the usual INS parameters. The equations of motion are then stated by Equation 2, where the variation of the navigation parameters is a function of the parameters themselves, and the inputs of accelerations and rotation rates. The expansion of these common equations can be found in the books of Savage (1998), whereas the specific notation and details on the reference frames are explained in previous published piece of work (Giroux et al, 2003, 2004). By using the direct representation of the variables, there are 22 components to numerically integrate to obtain the navigation solution. However, there are ways to reduce the number of variables to 12.

$$\mathcal{N} = \left\{ {}^{N}C_B, [{}^{E}\vec{v}_N]_N, {}^{E}C_N, h \right\} \in R^{22} \quad (1)$$

$$\dot{\mathcal{N}} = \mathcal{F}\Big(\mathcal{N}, [{}^{I}\vec{a}_B]_B, [{}^{I}\vec{\omega}_B]_B\Big) \quad (2)$$

For the indirect implementation of the Kalman filter, an error model has to be derived. Again, a set of error variables is defined by Equation 3 and its dynamics given by Equation 4. The error model used is the standard ψ -error model, found in many books and papers, like the ones of Pitman (1962), Goshen-Mesken (1992), Chatfield (1997) and Savage (2000).

$$\delta\mathcal{N} = \left\{ [{}^{E}\delta\vec{v}_N]_N, [{}^{E}\delta\vec{r}_N]_N, \vec{\psi}_N \right\} \in R^9 \quad (3)$$

$$\delta\dot{\mathcal{N}} = \mathcal{F}^{\mathsf{err}}\Big(\mathcal{N}, [{}^{I}\vec{a}_B]_B\Big)\delta\mathcal{N} \quad (4)$$

Finally, the extended Kalman filter applied to an arbitrary set $\mathcal{X}$ of variables to be estimated is presented as follows (Maybeck, 1994 ; Bryson et al, 1975):

$$\widehat{\mathcal{X}}_{k|k} = \widehat{\mathcal{X}}_{k|k-1} + K_k\gamma_k \quad (5)$$

$$\gamma_k = \mathcal{Y}_k - \mathcal{H}\Big(\widehat{\mathcal{X}}_{k|k-1}\Big) \quad (6)$$

$$\widehat{\mathcal{X}}_{k|k-1} = \Phi_k\Big(\widehat{\mathcal{X}}_{k-1|k-1}, \mathcal{U}_k\Big) \quad (7)$$

Direct integration

In the direct integration approach, the set of variables linked to the equations of motion is considered as the estimated variable in the extended Kalman filter. Therefore, Equation 2 represents the process to be estimated by the Kalman filter and the arbitrary set of variables in Equations 5 to 7 is replaced by the set of navigation variables:

$$\mathcal{X} \rightarrow \mathcal{N}$$

Also, the external measurement relation is given by Equation 8, where the set of measurement variables is given directly by the set of external measurements.

$$\mathcal{Y}_k^{\mathsf{dir}} = \mathcal{H}^{\mathsf{dir}}\Big(\mathcal{N}_{k|k}\Big) = \mathcal{M}_k \quad (8)$$

In the case of GPS measurement, the set of external measurements is given as follows:

$$\mathcal{M}_k = \left\{ P_k^{GPS}, V_k^{GPS} \right\} \in R^6 \quad (9)$$

Figure 1 depicts the signals flow. The extended Kalman filter operates in the “navigation filter” block resulting in the estimate of the set of navigation variables. The inputs are the accelerometers and the rate gyros, and the GPS is the external measurement signal.

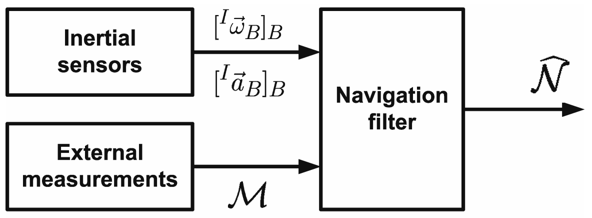

Figure 1: Direct integration approach

This configuration is not used very often due to its computation burden. Also, the Kalman filter operates more efficiently with state variables around zero, and that implementation is called indirect integration.

Indirect integration

In the indirect integration approach, the set of variables linked to *the error* in the equations of motion is considered as the estimated variable in the extended Kalman filter. Hence, Equation 4 represents the process to be estimated by the Kalman filter and the arbitrary set of variables in Equations 5 to 7 is replaced by the set of error variables:

$$\mathcal{X} \quad \rightarrow \quad \delta\mathcal{N}$$

Also, the external measurement relation is given by Equation 10, where the set of measurement variables is the difference between the set of navigation variables and the set of external measurements.

$$\mathcal{Y}_k^{\text{ind}} \quad = \quad \mathcal{H}^{\text{ind}}\left(\delta\mathcal{N}_{k|k}\right) = \mathcal{H}^{\text{ext}}\left(\mathcal{N}_k\right) - \mathcal{M}_k \qquad (10)$$

As for the direct integration, the set of external measurement can be given as follows in the case of GPS measurement:

$$\mathcal{M}_k \quad = \quad \left\{P_k^{GPS}, V_k^{GPS}\right\} \in R^6 \qquad (11)$$

However, the set of navigation variables as to be mapped to correspond to the same physical meaning as the external measurements, and can be represented by:

$$\mathcal{H}^{\text{ext}}\left(\mathcal{N}_k\right) \quad = \quad \left\{P_k^{INS}, V_k^{INS}\right\} \in R^6 \qquad (12)$$

Feedforward mode

In the feedforward mode of the indirect integration approach, the estimated navigation error from the Kalman filter is subtracted to the navigation solution, resulting in a corrected navigation solution:

$$\widehat{\mathcal{N}} \quad = \quad \mathcal{N} - \widehat{\delta\mathcal{N}} \qquad (13)$$

Figure 2 shows a block diagram of this configuration. The “Equations of motion solution” block numerically integrates Equation 2 to give an open loop solution. The extended Kalman filter operates in the “navigation filter” block resulting in the estimate of the set of error variables. The inputs are the accelerometers, and the GPS data and the open loop navigation solution are considered as external measurements.

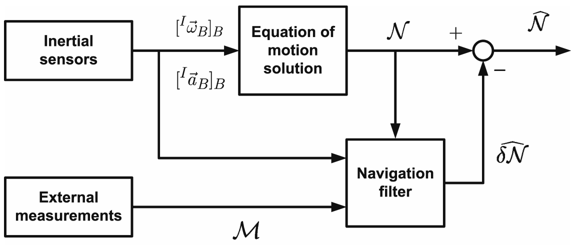

Figure 2: Indirect integration in feedforward mode

This configuration operates well if the error variables are kept small. But given the low-cost nature of the sensors, the estimated (and true) error in the navigation solution get out of bound rapidly. The feedback mode of the indirect integration answers this problematic.

Feedback mode

In the feedback mode of the indirect integration, the corrected set of navigation variables replaces the previous value of the set of navigation variables. The set of error variables is then reset to zero for the next round of estimation.

$$\widehat{\mathcal{N}} - \widehat{\delta\mathcal{N}} \quad \rightarrow \quad \widehat{\mathcal{N}} \qquad (14)$$

Figure 3 illustrates the approach. As for the feedforward approach, the “Equations of motion integration” block numerically integrates Equation 2 but is corrected by the estimated navigation error at each timestep. The extended Kalman filter still operates in the “navigation filter” block resulting in the estimate of the set of error variables. The inputs are the accelerometers, and the GPS data and the *corrected* navigation solution are considered as external measurements.

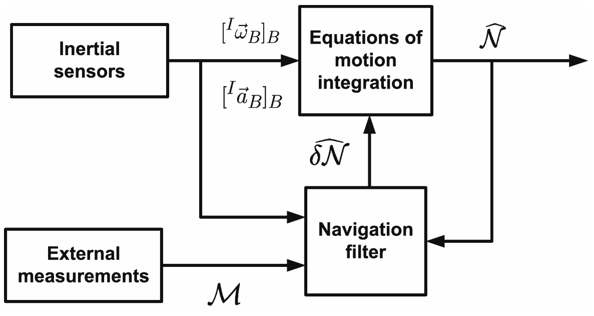

Figure 3: Indirect integration in feedback mode

This configuration is the most robust one and is necessary when operating with low-cost sensors.

This approach has been implemented in a Simulink environment and is the subject of the next section.

Description of the Simulink fusion algorithm architecture

The Simulink environment is a graphical-interface utility that provides powerful tools to design and prototype real-time algorithms. The hierarchical structure of the algorithm is presented at Figure 4. The top-level functions, depicted at Figure 5, are the equations of motion solution, the navigation filter and the online calibration of the sensors. The former has been presented in a previous paper (Giroux et al., 2003) and the later will be part of future publications. Herein, the focus is on the navigation filter implementation and experimental results.

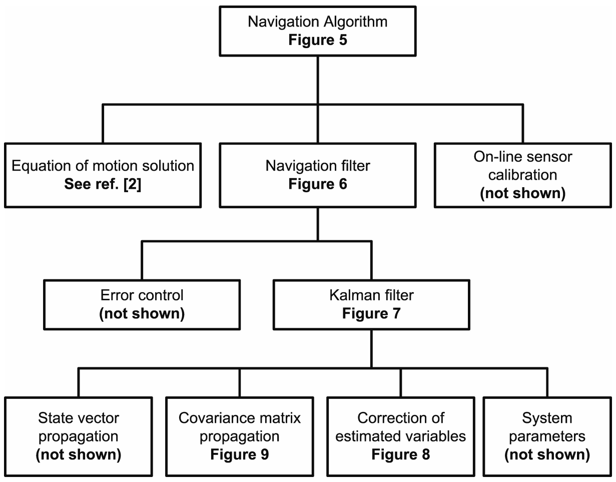

Figure 4: Hierarchical structure of the algorithm

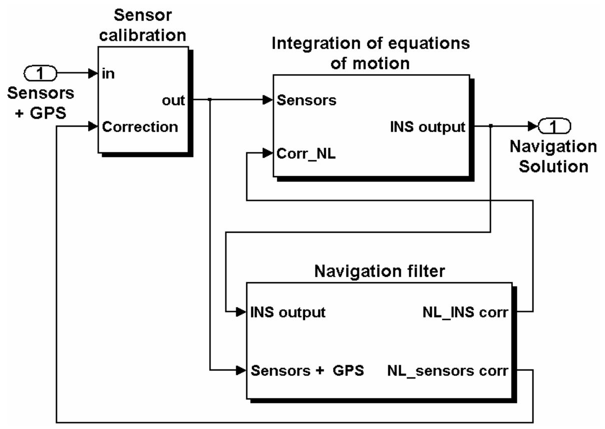

Figure 5: Top-level functions

The navigation filter is composed of two main parts: the extended Kalman filter and the error control. Figure 6 shows the signal flow of this blockset. The “error control” block manages the relation between the “extended Kalman filter” and the “integration of equations of motion” block (and the “sensor calibration” block when this functionality is used). If the indirect integration in feedforward mode is used, the “error control” block has no effect and no correction is sent to the “integration of equations of motion” block. However, the feedback mode implies the correction of the solution of the equations of motion and the reset of the estimated error variables. That is the main function of the “error control” block. It is driven by a state-machine which sends events that trigger the correction and the reset of the values.

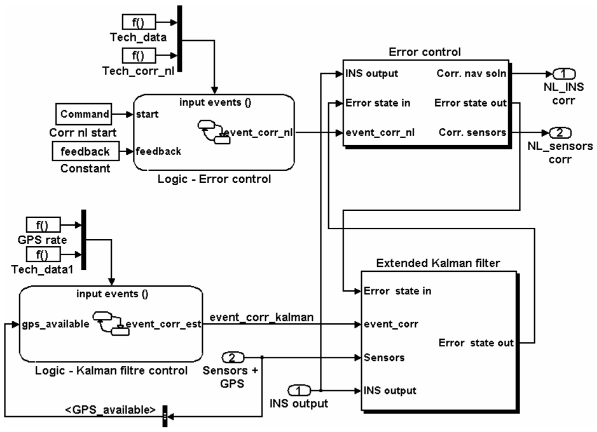

Figure 6: Navigation filter functions

The extended Kalman filter block of Figure 6 is also driven by a state-machine. The propagation of the estimates (state variables and covariance matrix) is done at each time step, whereas the event “correction” is driven by the reception of an external measurement. Figure 7, Figure 8 and Figure 9 detail the extended Kalman filter operation and the explanation of the two modes (with or without external measurements) follows.

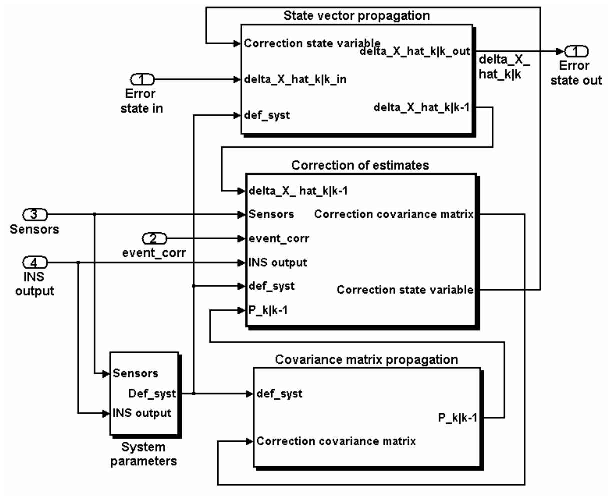

Figure 7: Extended Kalman filter functions

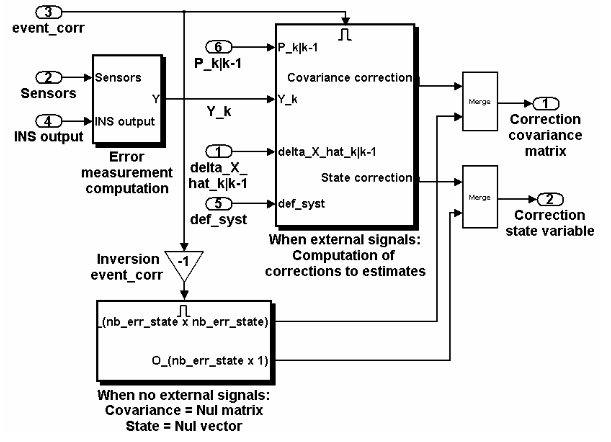
event_corr
Sensors
INS output
P_k|k-1
Covariance correction
Y_k
delta_X_hat_k|k-1
State correction
def_syst
Error measurement computation
When external signals: Computation of corrections to estimates
Merge
Correction covariance matrix
Correction state variable
Inversion event_corr
-1
(nb_err_state x nb_err_state)
O_(nb_err_state x 1)
When no external signals: Covariance = Nul matrix State = Nul vector

Figure 8: Correction of estimates functions

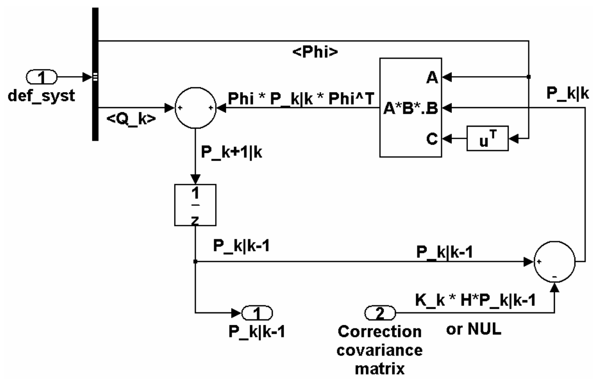

Figure 9: Propagation and correction of covariance matrix estimate

When there is no external measurement, the extended Kalman filter operates in the propagation mode. The "event correction" signal provided to the block "Correction of estimates" of Figure 7 is negative. Figure 8 shows an inside look of this block, and a negative "event correction" drives a nul output for the correction of the state variables and the covariance matrix. This correction (nul for the propagation phase) is then sent to the "State vector propagation" and "Covariance matrix propagation" blocks of Figure 7. An inside look at the block "Covariance matrix propagation" is shown at Figure 9. Since the correction is nul, only the propagation of the covariance matrix is performed.

On the other hand, if there is an external signal available, the "event correction" signal is positive and the extended Kalman filter superposes the correction phase to the prediction phase. In fact, this event triggers the computation of a correction in the block "Correction of estimates" of Figure 7, and detailed at Figure 8. In this detailed figure, the difference between the external signal (here a GPS signal) and the INS solution is first computed, and the result is used in the "Computation of corrections to estimates" block. These corrections are sent to the "State vector propagation" and "Covariance matrix propagation" blocks of Figure 7. Again, a look at the functionalities of the block "Covariance matrix propagation" illustrated at Figure 9 shows that the correction is applied to the propagation of the estimated covariance matrix. A similar process is performed for the correction of the estimated state variables. Also, it should be noted that a "system parameters" function updates the time varying parameters of the system model as depicted in Figure 7.

Computation assessment

In order to physically implement the algorithm on an embedded platform, it is important to assess the level of computation effort required to run the different functions of the algorithm. The embedded computer is a PC104 type with a Pentium Mobile processor of 266MHz and 128Mb of RAM. The Real-Time Workshop (RTW) toolbox of Simulink is used to generate C-code from the Simulink blockset-code, and then an executable format of the code is transferred to the embedded computer with the xPCTarget application. The computation time information (amount of time required to process all computations during a cycle) is obtained through the use of xPCTarget real-time run data log.

Following the companion paper previously published, Figure 10 shows the real-time computation effort of different integration methods.

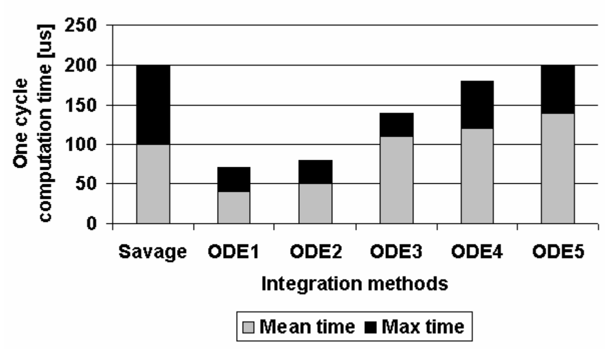

Figure 10: Computation time of different integration methods

Although not at issue in this paper, it is important to show that the preliminary assessment for the choice of an integration method made with simulation computation time still holds with real-time computation effort. Indeed, the Runge-Kutta of 4th order (ODE4) is of comparable burden with respect to the Savage algorithm. Then, the RK4 integration algorithm is still used in all the simulation and real-time implementations. For more information on the pros and cons, the reader is invited to consult the previously published paper (Giroux et al., 2003).

For the Kalman filter algorithm computation time (Figure 11), three phases have been studied: the linearisation of non-linear function, the prediction phase and the combined prediction/correction phase. No specialized form of Kalman filter algorithm was coded, meaning full matrix multiplications were performed. The linearisation and prediction phase are performed at each cycle. However, the prediction/correction mode is only activated when an external measurement is available. The number of state variables has a major influence on the computation time, as we know that the number of multiplications in the prediction phase of the Kalman filter is proportional to the cube of the number of state variables (Mult α n^3). The trend can be seen on Figure 11. Part on the navigation filter, the error control has also been investigated but was found to be negligible compared to the Kalman filter computation burden (mean computation time of 80 µs per cycle).

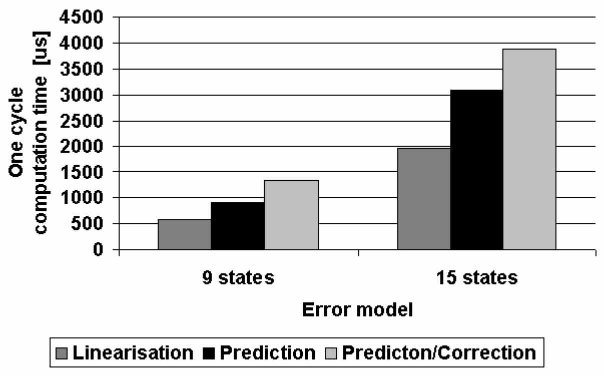

Figure 11: Computation time of Kalman filter

Although the previous graphs provide insight information about the complexity of the different functions, it is the total computation time that is relevant to determine the sample rate of the inertial sensors (Figure 12). Two common hypotheses have been compared: the full set of equations of motion and the flat-earth assumptions, reducing the complexity of the equations to integrate and the error control. As it can be seen, there are no significant differences between both approaches, mainly caused by the Kalman filter implementation which is the same for both schemes. Then, with a maximum total computation time of 4300 µs, a sampling rate of 200 Hz is achievable for acquiring the sensor data and processing the algorithm.

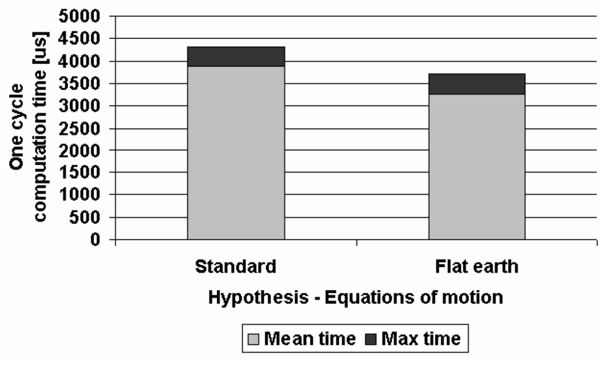

Figure 12: Total computation time for one cycle

Experimental results

The experiments have been conducted using a high-speed vehicle. The IMU, PC104 and the GPS receiver were fixed in the back trunk of the vehicle as illustrated in Figure 13. The Tetrad IMU unit is made up of four accelerometers and four gyros, each set of measuring devices arranged in a tetrahedral configuration. Acquisition and processing of data received from the IMU is carried out by the PC104 computer stack as seen in Figure 13. An ADC card is used to sample the IMU and temperature sensor measurements at 200 Hz. IMU data can be graphically displayed in real-time using an external monitor connected via the video card or can be transferred to a host computer for real-time or post processing via the Ethernet network card. Communication between the PC104 and the GPS receiver is made through a serial connection. A more detailed description of the experimental setup can be found in a previous publication (Giroux et al., 2004).

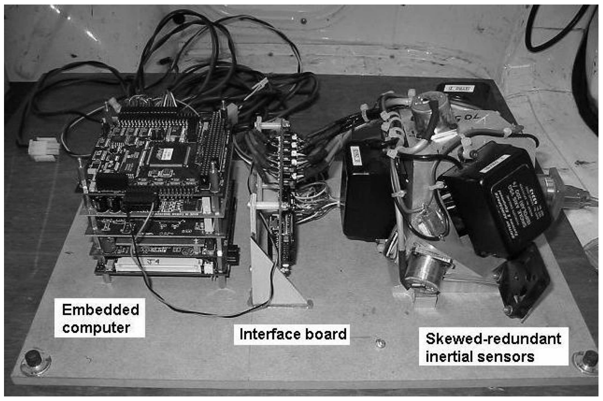

Figure 13: Experimental setup

Data acquisition

The xPCTarget environment allows the direct acquisition of data. For the IMU, a driver for the acquisition card already exists in the xPCTarget library so simple configuration of the blockset permits the acquisition of raw data from the IMU (Figure 14).

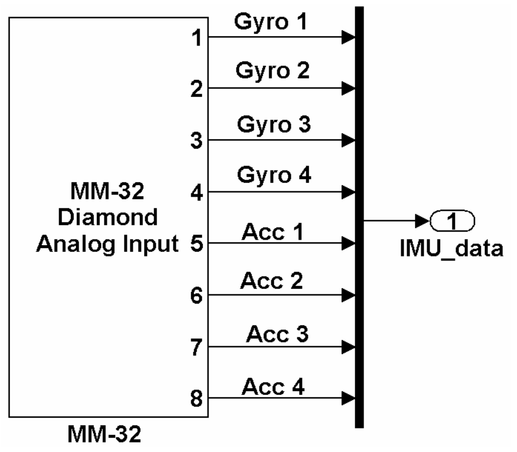

Figure 14: Blockset of IMU data acquisition

Figure 15 and Figure 16 show the orthogonal projection on the body axes of the redundant inertial sensor measurements for a trajectory of 6.5 minutes. The angular rate measurements have been compensated for based on the bias read directly from the sensors at rest. The acceleration measurements are not compensated. It can be seen that the noise in the sensors increases significantly when the vehicle starts moving (at 50 seconds), due to vibration.

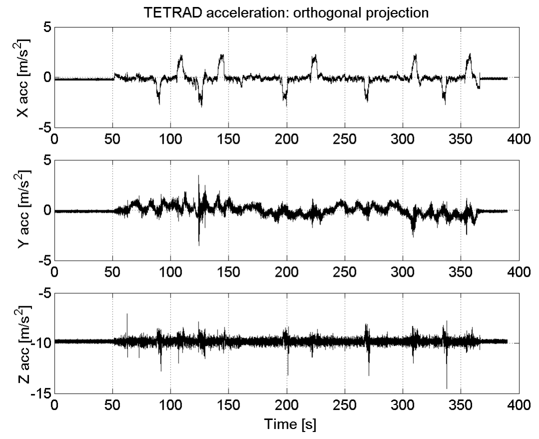

Figure 15: Orthogonal projection of accelerometer data

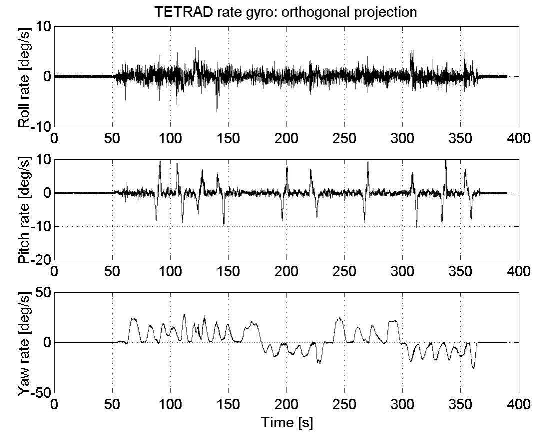

Figure 16: Orthogonal projection of rate gyro data

For the GPS signals, the use of a RS232 link is necessary and the configuration of the serial message has to be in accordance with the receiver specifications (Magellan, 2000). The decoded message (NMEA - National Marine Electronics Association) from the GPS receiver and used in the Simulink environment is composed of the following data, as Figure 17 exposed:

- Number of satellites in the field of view;
- Universal time;
- Latitude, longitude, altitude;
- Azimut;
- Longitudinal velocity;

- Vertical velocity;
- Dilution of precision (3D-position, horizontal, vertical and time).

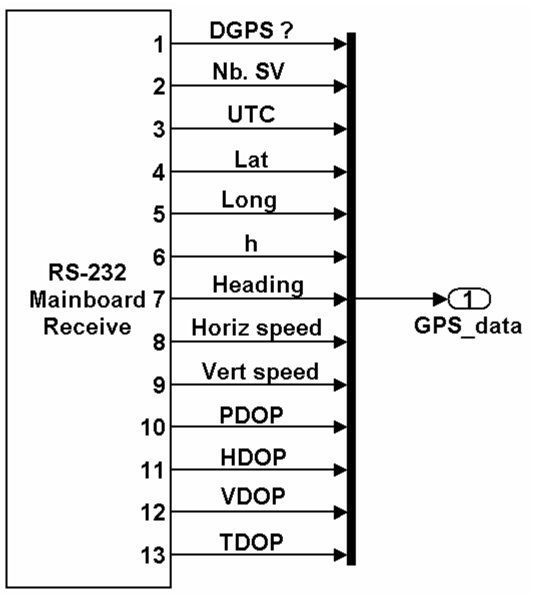

Figure 17: Blockset of GPS data acquisition

Post-processing

Figure 18 illustrates the 6.5 minutes navigation solution of the INS/GPS fused system compared to the GPS solution (represented by circles). The innovation sequence (Figure 19) of the position error in the navigation filter is mostly white noise, implying a well tuned filter. The results have shown that the INS/GPS integration algorithm with redundant low-cost measurement works appropriately when GPS fixes are continuous.

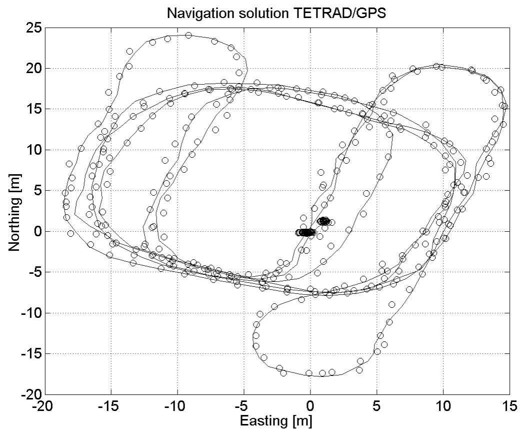

Figure 18: Horizontal solution fusion INS/GPS

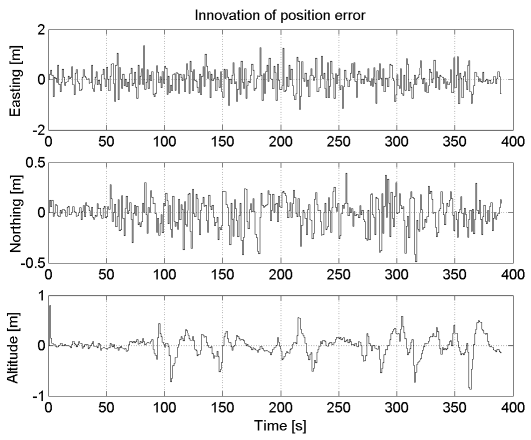

Figure 19: Innovation of position error

Discussion

Fast-prototyping for the design of INS/GPS systems is not very common and the limited amount of publications on this subject reflects that fact. The main challenge of this design choice is the level of confidence on the automatically generated code resulting from the interpretation of the high-level graphical code represented by Simulink. Although applications where human-at-risk would not be suitable for this kind of design, there are no reasons why the first stage of a project should not take advantage of the fast-prototyping approach.

The development environment used in this project was Simulink of MathWorks. Simulink's modularity and easy graphical design make it convenient for improvements and facilitate the transfer of knowledge to future collaborators (Otter et al., 2000). In addition, the Real-Time Workshop (RTW) suite and the xPC Target environment enhanced the simulation stage of Simulink by allowing rapid real-time testing. The xPC Target real-time application was used for the sake of simplicity. It generates its own real-time kernel and all the drivers associated with compatible hardware are already built-in. This design scenario (simulation of an algorithm and real-time testing via rapid-prototyping) has reduced dramatically the time between the validation of the algorithm and its real-time implementation.

However, the selected real-time scheme is not the only one available that uses the benefits of Simulink graphical-coding. It is possible to use different real-time environment and target implementation using Mathwork's third-party products. As examples, dSPACE (http://www.dspace.de) provides a real-time interface for multiprocessor systems; and OPAL-RT (http://www.opalrt.com) offers real-time development environments for Simulink with multiple real-time operating system targets (Windows NT/2000/XP, QNX, RedHawk Linux) and allowing true multi-threaded, multi-rate execution of subsystems. The basic is that RTW generates a compiled C-code from the blocksets of Simulink. Then, it only depends on the user to choose the target real-time operating system and its associated hardware drivers. The aforementioned third-party products simplify this step by providing the user with graphically-interfaced target real-time solutions.

The implementation of the extended Kalman filter using Simulink was not a trivial task. Indeed, the strategy used to re-initialize the state variable is one of many that could have been used, but is considered to be efficient. Also, the filter has been implemented in its full form (i.e. the full matrices are multiplied) and the computation effort grows rapidly with the increase of the number of state variables. Different implementations could be used to reduce the computation burden, but at a first stage the experience of implementing the extended Kalman filter in a fast-prototyping scheme is successful.

Also, the acquisition of the GPS signal over a RS232 link in Simulink required some thoughtful work. It has been decided to use standard serial ASCII message to decipher the GPS signal. The difficulty was residing mainly in the compliance with the GPS receiver message structure, where some particular character formats were not entirely compatible with the format available in Matlab. This could have been avoided using a binary transmission format for the RS232 link.

Nevertheless, it can be stated that the rapid prototyping approach chosen in the implementation has permitted the design of the algorithms and data acquisition scheme in an efficient manner and in a short development time period.

Conclusion

This paper addressed the fast-prototyping implementation of an extended Kalman filter for INS/GPS fusion and follows a previously published paper on the simulation of INS and the integration of the equation of motions with high-order methods (Giroux et al., 2003). The fast-prototyping approach is not popular among researchers in this field because they usually feel they do not have full control of the process. However, during an initial phase of a project, this design philosophy saves time and efforts.

Given its cycle processing time, the EKF and complete INS/GPS algorithm implemented in this project are found to be practically feasible for real-time application of low-cost navigation systems. Some algorithmic problems have been encountered during the integration of such programming tools with the function requirements but they are not comparable to the effort needed to develop all low-level components of a data acquisition scheme.

Also, the real-time assessment of the entire algorithm has not been addressed with respect to CPU peak and average load requirements. As highlighted in the discussion, different real-time environments are available with Simulink (as third-party products) that permit true multi-threaded and multi-rate execution. As part of on-going research, such CPU load analysis and the use of a multi-thread real-time application could lead to the full assessment of Simulink as competitive real-time development tool.

Part of the research team effort is to upgrade the simulator and add a graphical trajectory generator to facilitate the creation of desired trajectories. Also, the simulator should be accessible as an internet application where remote users could launch simulations and/or process their scenarios on a real-time platform and download the results.

Finally, this simulator and the associated hardware constitute a powerful tool to do rapid designs and prototyping of low-Cost GPS-aided INS. Given the results shown and future work in progress, it is believed that Simulink-based simulators will form the next generation of INS algorithm validation schemes.

Acknowledgements

The financial support for this work was partly provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), the “Fonds Nature et Technologie du Québec” (NATEQ) and the “École de technologie supérieure”. Also, the author would like to acknowledge the Australian Research Council Centre of Excellence programme and the Australian New South Wales State Government, as part of the research work performed at the Australian Centre for Field Robotics (ACFR), The University of Sydney, Australia. Finally, the authors would like to thank Dr. Salah Sukkarieh, from the ACFR, for his support.

Bibliography

Anderson, B. D. (1999). From wiener to hidden markov models. In *IEEE Control Systems*, pages 41 – 51.

Bryson, A. E. J. and Ho, Y.-C. (1975). *Applied optimal control,*

Chatfield, A. B. (1997). *Fundamentals of High Accuray Inertial Navigation*, volume 174 of *Progress in Astronautics and Aeronautics*. Virginia.

Eck, C., Chapuis, J., and Geering, H. P. (2001). Software-supported design and evaluation of low-cost navigation units. In *Proceedings of the 8th St-Peterburg International Conference on Integrated Navigation Systems*, pages 163 – 172.

Garg, S., Morrow, L., and Mamen, R. (1978). Strapdown navigation technology: A litterature survey. *Journal of Guidance and Control*, 1(3):161 – 172.

Giroux, R. (2004). *Capteurs bas de gamme et système de navigation inertielle: nouveau paradigmes d'applications*. Ph.d. thesis, Ecole de technologie superieure.

Giroux, R., Leach, B. W., Landry, R. J., and Gourdeau, R. (2003). Validation and performance evaluation of a simulink inertial navigation system simulator. *Canadian Aeronautics and Space Journal*, 49(4).

Giroux, R., Sukkarieh, S., and Bryson, M. (2004). Implementation of a skewed-redundant low-cost ins in a fast-prototyping environment. In *Proceedings of the Institute of Navigation National Technical Meeting*. San Diego.

Goshen-Meskin, D. and Bar-Itzhack, I. Y. (1992). Unified approach to inertial navigation system error modeling. *Journal of Guidance, Control and Dynamics*, 15(3):648 – 653.

Kriegsman, B. and Mahar, K. (1986). Gravity-model errors in mobile inertial-navigation systems. *Journal of Guidance*, 9(3):312 – 318.

Magellan (2000). G12 GPS OEM board and sensor reference manual. Technical report, Magellan Corporation.

Maybeck, P. S. (1994). *Stochastic models, estimation, and control*, volume 1 of *Mathematics in Science and Engineering*.

Otter, M. and Cellier, F. E. (2000). Software for modeling and simulating control systems. In Levine, W. S., editor, *Control System Fundamentals*, pages 419 – 432. CRC Press.

Pitman, G. (1962). *Inertial Guidance*. Wiley, New York.

Savage, P. G. (1998a). Strapdown inertial navigation integration algorithm design part 1 : Attitude algorithms. *Journal of Guidance, Control and Dynamics*, 21(1):19 – 28.

Savage, P. G. (1998b). Strapdown inertial navigation integration algorithm design part 2 : Velocity and position algorithms. *Journal of Guidance, Control and Dynamics*, 21(2):208 – 221.

Savage, P. G. (2000). *Strapdown Analytics, Part 1*. Strapdown Associates Inc., Maple Plain, MN.