IJCCCE, VOL.7, NO.1, 2007

INTELLIGENT SINS NAVIGATOR BASED ON ANN

Dr. Salam A. Ismaeel [*]

Received on: 29 / 12 / 2005
Accepted on: 5 / 12 / 2006

Abstract

Most of the positioning technologies for modern inertial navigation systems have been available for the last 25 years that has focused on development of the Strap-down Inertial Navigation Systems (SINS) because of its low cost.

This paper presents an intelligent navigator to overcome the limitations of existing SINS algorithms. This algorithm is based on Artificial Neural Network (ANN).

Using window based weight updating strategy; the intelligent navigator was evaluated using several SINS hypothetical field tests data and the results demonstrated superior performance to traditional navigator in the position domain.

الخلاصة

ان أغلب التطوير في منضومات الملاحة من النوع الثابت (SINS) للـــ 25 سنة الماضية، ركّزتْ على تطويرِ خوارزمية الملاحة الأرضية، بسبب كلفتِها المنخفضةِ.

يقدّمُ هذه البحث مَلاحَ ذكيَ للتَغلّب على التقييدات الموجودة في خوارزمية الـــ (INS). هذه الخوارزميةِ مستندة على الشبكة العصبيةِ الإصطناعيةِ (ANN).بإستعمال إستراتيجية النافذةِ لتجديد أوزان الشبكة، قُيّمَ المَلاح الذكي بأستخدام عِدّة بيانات إختباراتيةِ مطبقة على منظومة (SINS). والنَتائِج بينت أداءَ متفوّقَ على خوارزمية المَلاحةِ التقليدية في مجالِ تحديد الموقعَ.

1. Introduction

Navigation is the determination of a physical body's position and velocity relative to some reference coordinate frame or coordinate grid [1]. Inertial instruments (i.e., accelerometers and gyros) do not rely on detection of fields created external to the vehicle. They are self-contained and rely on only physical laws of motion. Therefore navigation systems based on inertial instruments are inherently more robust to interference than other systems [2]. This concept refers to inertial navigation systems (INSs).

Since there is a lack of researches towards the conceptual intelligent navigator, this paper is devoted to develop an intelligent navigator

That consists of artificial neural networks

(ANNs) based on Terrestrial SINS algorithm described in [3]. Ultimately, the intelligent navigator is expected to overcome or, at least,

Reduce the limitations of the conventional Based SINS algorithms. As each of these limitations contributes to certain amount of positional error accumulation during computational errors [4], therefore, the proposed new algorithms are expected to reduce the impact of these limitations by reducing the positional error accumulation during navigation phase. Figure (1)

Illustrates the core components of this navigator.

2. ANNS Based SINS Architecture

Being inspired by figure (1), the first step toward building an intelligent navigator is to provide the SINS architectures that can be applied to generate and acquire necessary navigation knowledge. Consequently, two ANNs based SINS architectures are discussed in the following sections.

2.1 Velocity Update Architecture

The first SINS architecture is called the Velocity Update Architecture (VUA) which consists of a two layered feed-forward neural network with Levenberg-Marqudrt and conventional generalized Delta-Rule back

Propagation learning algorithms applied for off-line and on-line learning [5], respectively. It integrates the data from inertial measurement unit (IMU) and mimics the dynamical model of the vehicle to generate navigation knowledge. Thus based on six degree of freedom (6DoF) equations of motion block shown in figure (2), designed in *SimuLink*, acquired navigation knowledge can be applied to predict the vehicle's velocity during IMU errors in real time.

The topology of the VUA is illustrated in figure (3). The input neurons receive the acceleration ($A_{INS}\ (t)$) and angular velocity at current epoch ($AV_{INS}\ (t)$).

The number of hidden neurons is decided empirically. The output neurons generate velocity in the local level frame at the current epoch in North, East, and Down velocities ($N_{VUA}\ (t)$, $E_{VUA}\ (t)$, and $D_{VUA}\ (t)$), respectively. Thus, the navigation knowledge can be learnt, stored and accumulated during the availability of the INS signal. On the other hand, during INS signal absence or IMU errors, the latest acquired navigation knowledge can be retrieved from the navigation information database, see section three for more details, of the intelligent navigator to predict the velocity in real time.

2.2 Position and Velocity Update Architecture

The second INS architecture is named the Position and Velocity Update Architecture (PVUA). It consists of two different two layered feed-forward neural networks that work in parallel. Similar to VUA, PVUA is applied to generate navigation knowledge which can be used to provide, in real time, the vehicle's position and velocity during IMU errors.

The topologies of PVUA are illustrated in figure (4). In fact, PVUA is the combination of

a Velocity Update Architecture (VUA) with a modified version of the PUA.

The input neurons of the PUA receive the velocity ($N_{VUA}\ (t)$, $E_{VUA}\ (t)$, and $D_{VUA}\ (t)$) from VUA and time epoch (t). The output neuron generates PUA estimated position ($N_{PUA}\ (t)$, $E_{PUA}\ (t)$, and $D_{PUA}\ (t)$).

During INS signal absence, similar to VUA, the latest acquired navigation knowledge obtained through use of the PVUA can be retrieved from the database of the intelligent navigator to predict the positions ($N_{PUA}\ (t)$, $E_{PUA}\ (t)$, and $D_{PUA}\ (t)$) in real time.

3. Navigation Information Database

The second step towards building the intelligent navigator is to store the learnt navigation knowledge provided by INS algorithms presented in the previous section. As a result, a navigation information database (NAViBASE) that contains the acquired and learnt navigation knowledge can serve as the "brain" of the intelligent navigator. Therefore, several issues regarding the NAViBASE are addressed as follows:

- **Content of NAViBASE:** The database consists of the training samples (input vectors and desired output vectors) and estimated synaptic weights during the availability of the IMU signal. Thus, these components can be regarded as the navigation knowledge. In other words, the content of the database varies with the topologies of different INS algorithm (i.e., according to navigation solutions that will be given by INS algorithm).

• **Distributed navigation knowledge storage:** The content of NAViBASE becomes more complicated with complicated INS architectures. Therefore, considering the efficiency of database maintenance and retrieval, the navigation knowledge learnt by each sub-component should be stored individually in a distributed way, as shown in figure (5).

• **Off-line database maintenance:** The simplest way to reduce the storage requirement is to remove any redundant training samples that include inputs and their corresponding desired outputs. As for the synaptic weights, they should be kept without any change as they are the core component of the navigation knowledge. Using VUA as an example, a simple procedure that can be applied prior to navigation (i.e., during alignment) or after navigation before shutting down the system, is given below:

1. Regroup the training samples: Using one of the training inputs (i.e., $A_{INS}\ (t)$ or $AV_{INS}\ (t)$) as the index; the training inputs can be regrouped to increase the efficiency for maintenance.
2. Locate redundant navigation knowledge: Although it is difficult to locate a pair of training samples that are exactly the same, searching the most similar pairs of training samples using threshold values then deciding if they are redundant or not is possible.
3. Remove the redundant navigation knowledge.

The implementation of NAViBASE was done with *MatLab* application. Up to now, the intelligent navigator has been given the ability to generate, and learn navigation knowledge and it also has been given the "space" to store navigation knowledge. However, there is still one thing missing. It requires a way to accumulate the acquired and learnt navigation knowledge and store them for further retrieving or generalization.

4. Window Based Weights Updating Strategy

As the synaptic weights are the core components of the navigation knowledge, the final step towards building the intelligent navigator is to develop a strategy to accumulate the acquired navigation knowledge by updating the synaptic weights whenever the IMU signal is available (i.e., no sensor errors or absence of signal).

In most of their applications, ANNs are trained using some known training data set (input/desired output) to obtain the optimal values of the synaptic weights via off-line training. For any other set of inputs, different from those used in training, the synaptic weights can then be applied to provide prediction of the network outputs. It is worth mentioning that ANNs weights are frozen after completing the training procedure and no further modification will be made during the prediction process [6].

In fact, off-line training can work well in case of slowly changing time sequences [7]. In the case of INS navigation applications, it is required to track direction changes and mimic the motion dynamics utilizing the latest available INS data. In other words, the synaptic weights should be updated during the navigation process to adapt the network to the latest INS sensor readings whenever the INS signal is available.

To implement such criterion, a window-based weights updating strategy, which utilizes the synaptic weights obtained during the conventional off-line training procedure (or probably from previous navigation missions) is stored in the NAViBASE and is presented in this research. This criterion utilizes the latest available navigation information provided by the INS signal window to adapt the stored synaptic weights so that they can be applied to mimic the latest motion dynamic. The window-updated synaptic weights are stored after each training stage. They are then used as initial values for the weights to be estimated during the next training window or for prediction during INS signal absence. Prior to looking into the details of the window based weights updating strategy, several aspects of traditional weights updating strategies are given. Traditional methods can be classified as:

1. **Sample-by-sample training**, also known as on-line or sequential training, that modifies the weights for each input record after computing the weights updates;
2. **Batch training**, which computes the synaptic weight updates for each sample and stores these values (without changing the weights). At the end of the whole training procedure, all the synaptic weight updates are added together and then the weights are modified with the accumulated synaptic weight updates [7].

From an online operational point of view, the sequential mode of training is preferred over the batch mode since less local storage is required. In addition, the random presentation of the pattern makes it less likely for the standard back-propagation algorithm to be trapped in a local minimum if the sequential mode of training is utilized. In contrast, the use of batch mode provides a more accurate estimate of the gradient vector, thus giving more accurate estimation of the weights [7].

Another major advantage of sequential training over batch training arises if there is a high degree of redundancy in the data.

On the other hand, the sequential training updates the weights after receiving each record of the input samples. Therefore, it will not be affected by such highly redundant data. However, during batch training, the network can learn more general relationships as it utilizes most of the available training data at the same time instead of sample by sample. Both generalization and training efficiency are very critical for INS applications, therefore, developing a special weights updating strategy that can preserve the generalization ability without losing too much training efficiency is very important.

4.1 Development of Window Based Weights Updating Strategy

Although the stored weights might not be able to provide accurate prediction during all INS absence, it can be applied as the initial weights at the beginning of a new navigation mission. The INS window signal concept is then applied to introduce new navigation knowledge to modify stored synaptic weights during navigation. In fact, this method combines the advantages of both sequential mode and batch mode of training in order to make the training procedure suitable for real-time processes. In addition, the weights of each window are then updated via batch training mode. In other words, the weights of each window are updated sequentially. As depicted in figure (6), the procedure of the window-based weights updating method is given below:

- **Weights initialization:** The initial weights can be obtained using previously stored weights that are stored in NAViBASE or random initialization. In this work, the initial weights were obtained using random initialization. After that, the weights were stored in NAViBASE after each navigation mission could be applied as the initial weights for the next mission.
- **INS signal reception:** At the next INS window, INS (i+1), the stored weights or the initial synaptic weights, W (i-1), are updated utilizing the previous available INS information (INS (i)). These weights are stored as W (i) after training was completed. This step is repeated until IMU signal blockage is detected.
- **INS Absence:** As depicted in figure (6), in case of a INS blockage (after INS (i)), W(i-l) is first applied for real time prediction), the prediction using W (i-1) and the training of W (i) can be operated in parallel. For simplification, the update procedure during INS blockage can be paused thus W (i-1) is applied to provide prediction during entire INS blockage and it can be updated after the reception of next available INS signal window.

Since the ANNs training procedure takes time, updating the synaptic weights immediately at the latest available sample of INS signal before blockage is difficult. However, the utilization of the proposed method can still provide reasonable prediction accuracy during INS blockage since it provides the latest updated weights instead of real time updated weights for real time prediction. Therefore, failure in providing real time updated synaptic weights doesn't mean the intelligent navigator is not able to provide real time prediction. On the contrary, it can utilize the latest acquired and learnt navigation knowledge to provide real time solutions. Combining the latest INS window signals, stored weights can be adaptively updated to follow the latest motion dynamics thus improving the prediction accuracy during INS blockage.

4.2 Training Procedures

The training samples acquired for the window based weights updating strategy can be arranged through using the following two procedures:

- **One step training procedure:** The training samples acquired for each INS window during navigation are the combination of stored training samples (*T(NAViBASE)*) and available training samples obtained at the end of each INS window ($\sum_{i=1}^{n} TW(i)$, n is the n^{th} window). In other words, as the size of training samples increases during navigation, the size of NAViBASE grows during navigation as well.

The advantage of the one step training is that it can provide better generalization of the navigation knowledge by incorporating stored and previous training samples during navigation. The one step training procedure is recommended at the early stage for building the intelligent navigator as the navigation knowledge acquired by the navigator at this moment might not be enough to provide acceptable accuracy during INS blockages. As the size of NAViBASE is quite small, the incorporation of stored training samples doesn't slow down the learning process during each window; actually, it can provide better generalization of the navigation knowledge, but when the size of NAViBASE increases this will slow down learning process.

- **Two steps training procedure:** The training samples acquired for each INS window during navigation are obtained at the end of each INS window (*TW(i)*). After navigation, all the training samples acquired during the navigation are recalled and combined with the stored training samples (*T(NAViBASE)*) then fed into the navigator to improve the generalization of navigation knowledge using a conventional off-line batch training method. This procedure is recommended for the regular operational stage for building the intelligent navigator. After several field tests, the navigator might accumulate enough navigation knowledge to provide navigation solutions during navigation without incorporating stored training samples. In other words, the size of training samples is the same as the INS window. Therefore, the training speed during each window is expected to be faster than the previous procedure.

After navigation, all the training samples acquired during the current navigation are recalled and combined with the stored training samples first to remove redundant navigation knowledge, and are then re-trained to improve the generalization of the navigation knowledge for future navigation missions. This will ensure to keep NAViBASE at specific size to avoid slowing down learning process.

The key factor that can accelerate the learning is the generalization of navigation knowledge. The perfect solution is to obtain the most generalized navigation knowledge that can then be fed into the navigator in one field test. However, that is not the case for real life applications. Therefore, the navigator must have the ability to evolve during each navigation mission to provide generalized navigation knowledge for future missions. Thus, using the proposed INS architectures, NAViBASE, and window based weights updating strategy, the intelligent navigator has the ability to generate and accumulate the navigation knowledge. In other words, it can learn and evolve continually to provide updated navigation knowledge and fill the gap between INS blockages.

5. Performance Analysis of the Intelligent Navigator

The evaluation and the performance of ANN based SINS architectures, NAViBASE, and the window based weight updating strategy, will be discussed in the section.

- **Prediction by NAViBASE**

The durations of both tests were about 350 seconds. During duration of the first test no IMU signal blockage periods were intentionally introduced for the purpose of using trajectory to obtain the stored weights for the estimation of the second field test. The first trajectory in terms of position in three directions North, East, and Down is shown in the Figures (7a-c), respectively. During the duration of the second test INS signal blockage periods were intentionally introduced for the whole navigation period (i.e., 350 second) in regular form.

VUA and PVUA were implemented as the SINS architectures for the intelligent navigator. IMU measurements obtained during the 1st field test were fed into the VUA and PVUA to obtain navigation knowledge (stored synaptic weights) for the 2nd field test. Thus, the NAViBASE was applied to fill the gap between IMU signal blockages during the second field test. In order to compare the performance of the intelligent navigator against ANN based INS technique (PVUA), and conventional INS techniques, three different periods were considered as gaps in time domain at:

1. (50-100), 50 seconds,
2. (150-200), 50 seconds,
3. (250-275), 25 seconds.

These periods represent blockage of IMU signal for three times.

It can be noticed from Figures (8a-c), the intelligent navigator provided stable solutions along the whole test of 350 seconds.

Although the redundant navigation knowledge resulted in several error peaks or oscillations, intelligent navigator based on both VUA, and PUVA demonstrated the ability to reduce the impact of time growing errors in the long term.

The last three Figures demonstrated the prediction ability of proposed architectures. It is a significant difference and improvement in comparison with time growing error characteristics of the conventional SINS described in [4].

- **Performance of the window based weight updating strategy**

To evaluate the performance of the window based weight updating strategy, two different window sizes were considered (5s, 10s). In addition, 70 windows, 35 windows were implemented with stored weights obtained via the 1st field test for each size respectively. Data were then updated utilizing the proposed method and 2nd field test data with the availability of an IMU signal. In case of an INS blockage, the latest updated weights were applied to provide real time prediction, as shown in figure (9).

The window based weights updating strategy with stored weights can provide alternative weight updating algorithms for INS and can improve the position accuracy during INS blockages. The preliminarily results demonstrated the potential to incorporate the intelligent navigator as the alternative navigation algorithm for next generation of navigation systems as it can overcome or reduce the limitations of conventional techniques.

6. Conclusions

The conclusions drawn from the results presented in this paper are:

1. The parameters of the intelligent navigator are included in the navigation knowledge. Thus, they can be updated without a human expert during navigation whenever newly updated navigation knowledge is acquired.
2. The PVUA architecture is recommended as the INS architecture for systems using a navigation grade IMU. It can achieve high level positioning accuracy requirement for real time prediction without INS computation components.
3. The results presented in this work strongly indicate the potential of including the intelligent navigator as the core navigation algorithm for the next generation navigation system.

7. References

1. Kenneth R. Britting. (1971), *Inertial Navigation Systems Analysis*, by Johan Wiley and Sons, Inc.
2. Xiaohong Zhang. (2003): "Integration of GPS with A Medium Accuracy IMU for

Meter_Level Positioning", M.Sc. Thesis, Department of Geometrics Engineering, University of Calgary.

3. Salam A. Ismaeel. (2003): “Design of Kalman Filter of Augmenting GPS to INS Systems” Ph.D. Thesis, Computer Engineering Dept., College of Engineering, Al-Nahrain University.
4. Salam A. Ismaeel and Sameir A. (2005): “Development of Six-Degree of Freedom Strapdown Terrestrial INS Algorithm”, *Journal of Um-Salama for Science*, vol. 2, no.1, pp.155-164.
5. Matlab Toolboxes
6. Ham, F.M. and Kostanic, I. (2001): *Principles of Neuro Computing for Science and Engineering*. McGraw-Hill.
7. Saad, D. (1998): "On Line Learning in Neural Networks", Cambridge University Press.

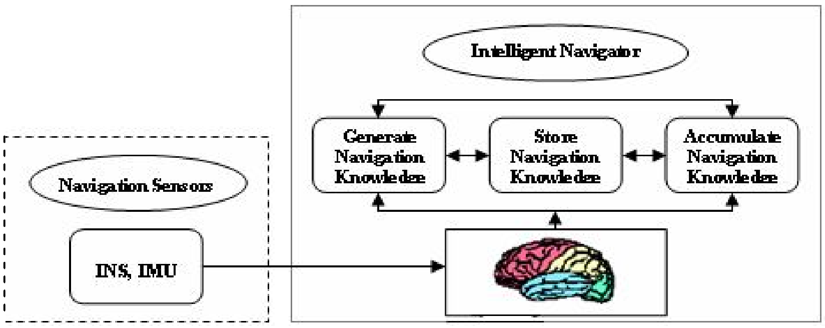
Intelligent Navigator
Generate Navigation Knowledge
Store Navigation Knowledge
Accumulate Navigation Knowledge
Navigation Sensors
INS, IMU

Figure 1: Core components of the intelligent

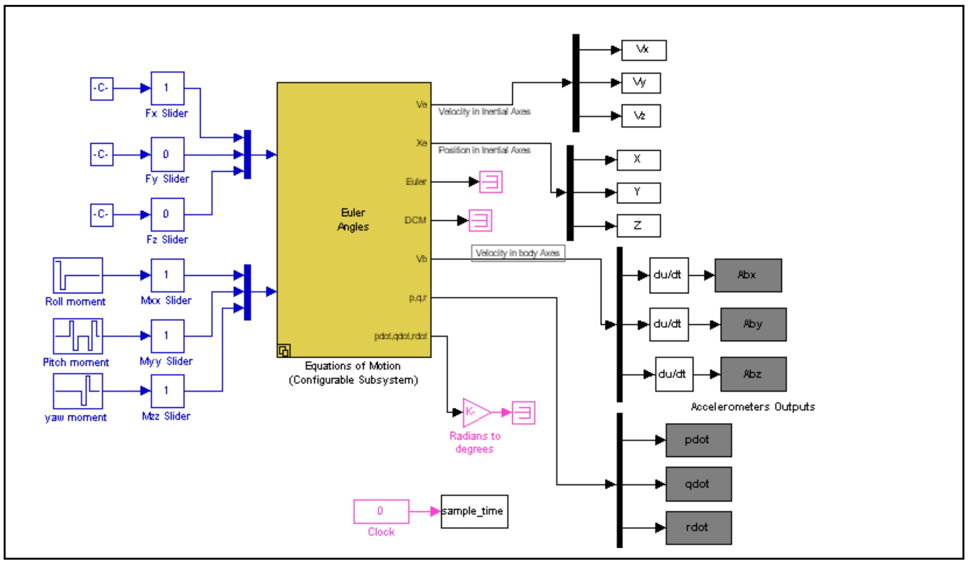

Figure 2: Six degree of freedom (6DoF) simulation

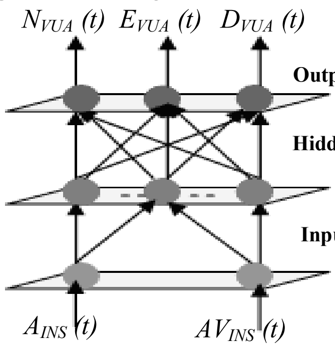
N_{VUA} (t) E_{VUA} (t) D_{VUA} (t)
Output Neurons
Hidden Neurons
Input Neurons
A_{INS} (t) AV_{INS} (t)

Figure 3: Topology of VUA

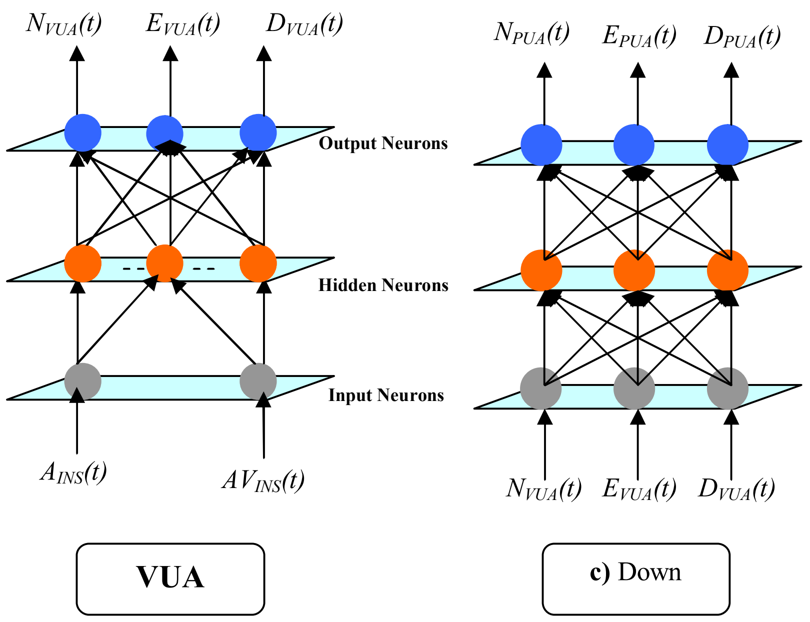

Figure 4: Topology of PVUA

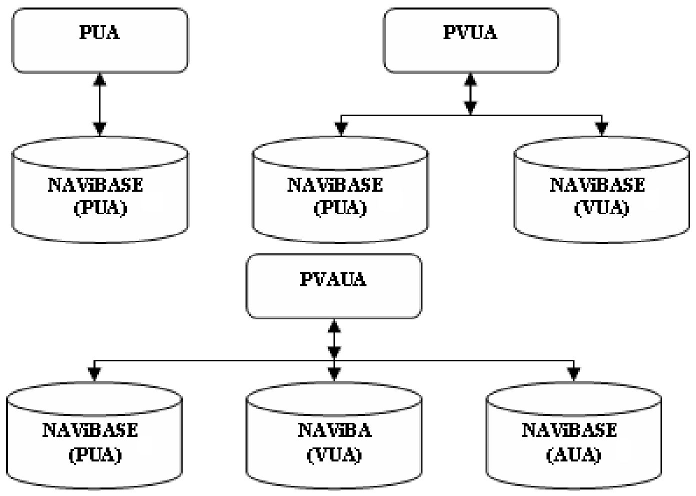

Figure 5: Distributed navigation knowledge storage

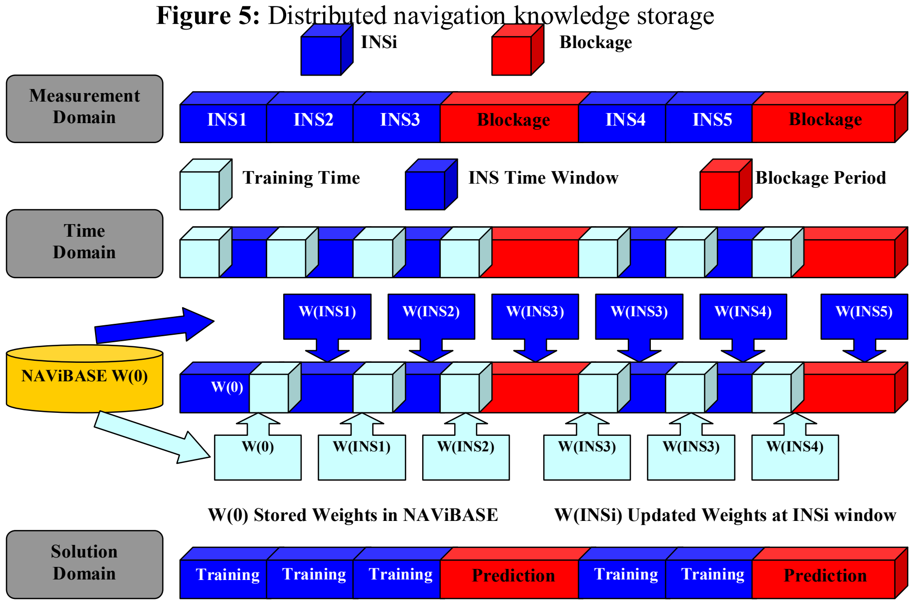

Figure 6: Window based weights updating strategy

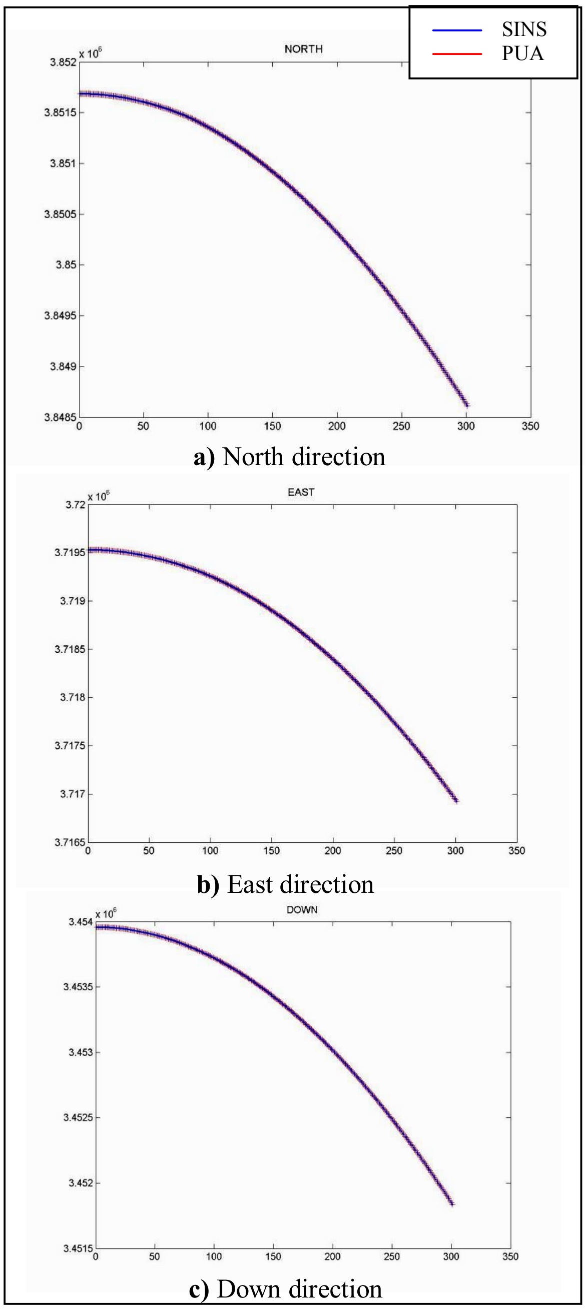

Figure 7: Position of the intelligent navigator (without prediction) in a) North, b) East and c) down directions

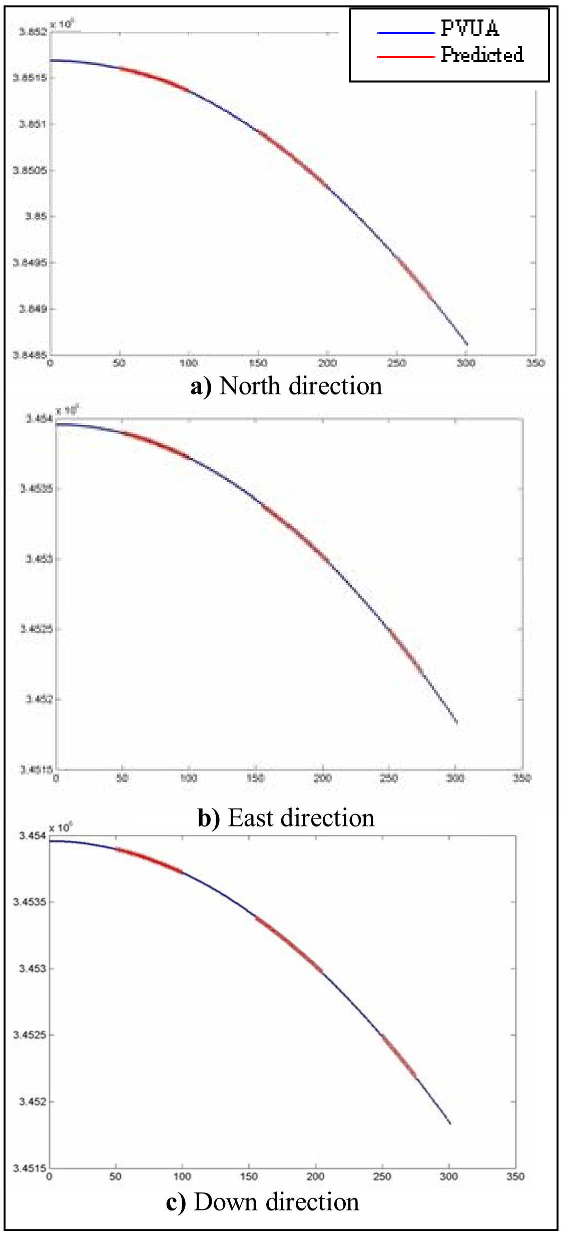

Figure 8: Stable position prediction in the intelligent navigator in a) North, b) East, and c) Down directions

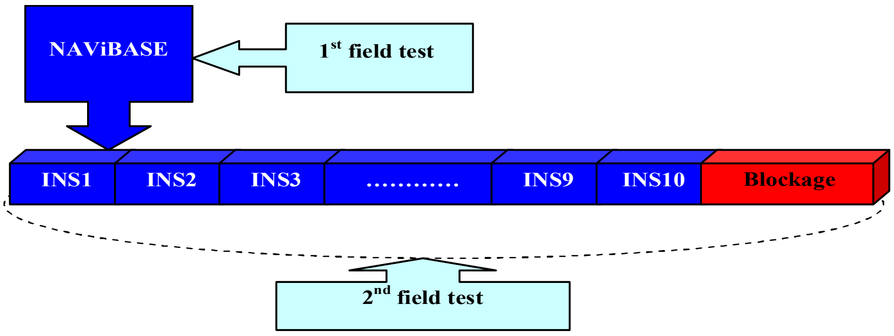

Figure 9: Window-based weights updating strategy