Optimization and Comparison of Two Data Fusion Algorithms for an Inertial Measurement Unit

S.A.Quadri
Collaborative Microelectronic Design Excellence Centre Universiti Sains Malaysia, Engineering Campus
Malaysia
reachquadri@yahoo.com

Othman Sidek
Collaborative Microelectronic Design Excellence Centre
Universiti Sains Malaysia, Engineering Campus
Malaysia
othman.cedec@usm.my

***Abstract*—Data fusion is a multilevel and multifaceted process that deals with the combination of data and information from single and multiple sources to achieve enhanced accuracy and precision. Development of algorithm plays significant role in the performance of data fusion system. We present two algorithms to fuse the data obtained from an accelerometer and gyroscope in an inertial measurement unit (IMU). First, we employ well-known Kalman filter algorithm and then we propose a new algorithm, namely decentralized data fusion algorithm based on Factor analysis model. After comparing the performance of both the algorithms, we switch our study to optimize the code. Matlab profiler carries out comparison and analysis. The code is optimized to speed up the execution time.**

Keywords— Data fusion; algorithm ; inertial measurement unit; Kalman filter; functional programming

I. Introduction

Sensor fusion is a process of integration and extraction of desired information from two or more sensors. In other words, it is a process of combining multiple sensors to provide more useful information than the sum of individual sensors. Fused sensor data from various sensors offers several advantages compared to the data from a single sensor [1].

Multisensor data fusion uses many techniques, such as the method of least squares, Bayesian method, Kalman filters, Dempster–Shafer's method, Fuzzy logic and neural networks [2]. Kalman filtering [3] is one of the most significant techniques used during the past decade. The widespread use of Kalman-based solutions is a testament to their accuracy and effectiveness; however, they have few disadvantages, as discussed by Madgwick [4].

On the surface, the notion of data fusion may appear to be straightforward but the design and real-time implementation of fusion systems is an extremely complex task. The most critical issues related to implementation of data fusion systems include, requirements analysis, sensor selection, architecture selection, algorithm selection, software implementation, and testing and evaluation. In order to meet the growing demands of applications, new algorithms have continually been designed and developed. The choice of the most appropriate algorithm depends on the complexity of the target problem; obviously the more complex the problem is, the more complex the algorithm would be. There is no perfect algorithm that is optimal under all conditions [5].

Data fusion algorithms are legionary, including mainly the physical models, feature-based inference techniques and cognitive-based models [6]. By the virtue of scalability and modularity, decentralized fusion algorithms have significant role in data fusion systems [7-9].

Although, many researchers have worked on sensor fusion and error analysis in IMU [10-11], considering the potentials of decentralized data fusion notion, the authors presented decentralized data fusion algorithm utilizing factor analysis [12]. However, decentralized data fusion notion facilitated the study of noise parameter associated with individual sensors, and the factor analysis model could compensate the missing data in sensor fusion system, but at the cost of performance. Thus, poor performance of our proposed method made us to suggest Kalman filtering to be better option for data fusion.

A computer program may be optimized so that it executes more rapidly, becomes capable of using less memory storage or other resources and draws less power [13]. Algorithm development is based on the successful construction of equivalent platform and global models. The algorithm should be designed to achieve maximum time and space efficiency, by exploiting algorithm-engineering guidelines [14]. A better programming style has significant role to optimize a given code. Thus, selecting Kalman filter to be better option, few guidelines are implemented to optimize the code.

In this study, we compared two data fusion algorithms developed to fuse data in an IMU, and finally optimized the code to minimize the execution time. The article is organized as follows, Section 2 concisely outlines IMU, and Section 3 briefly discusses development of data fusion algorithm using Kalman and Factor analysis method. In Section 4, we analyze and compare all three algorithms using profiler and propose functional programming methods to optimize the code.

II. INERTIAL MEASUREMENT UNIT

An inertial measurement unit (IMU) is an electronic device that measures and reports a craft's velocity, orientation, and gravitational forces, using a combination of accelerometers and gyroscopes. IMUs are typically used to maneuver aircraft including unmanned aerial vehicles (UAV), and spacecraft including shuttles and satellites. An IMU works by detecting the current rate of acceleration using one or more accelerometers, and detects changes in rotational attributes like pitch, roll and yaw, using one or more gyroscopes. A basic unit with assembly of components is shown in Fig.1. Detailed description on principle, working and application of IMU is available in the literature [15-16].

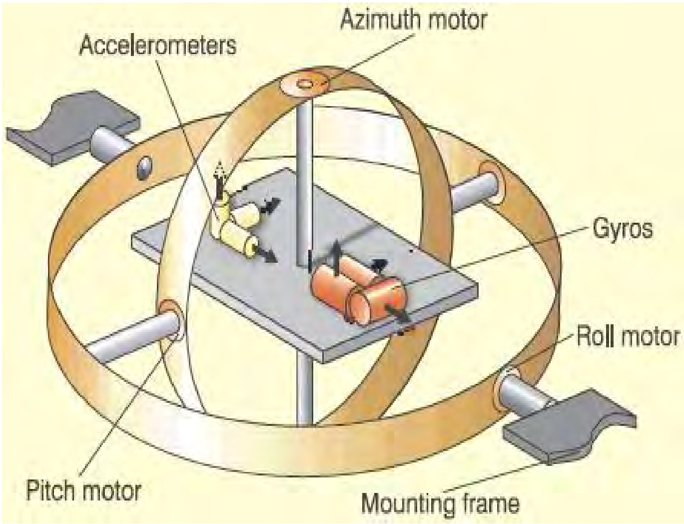


Figure 1. Inertial measurement unit [16]

III. KALMAN FILTER METHOD

The Kalman filter, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, such as noise (random variations) and other inaccuracies, and produces estimates of unknown variables; the results so obtained are expected to be more precise than those based on a single measurement alone. More formally, the Kalman filter operates recursively on streams of noisy input data to produce a statistically optimal estimate of the underlying system state. It is well-established methodology for model-based fusion of sensor data [17-18].

The first step is collection of data from sensors. Data is obtained from the SparkFun IMU that has noise variance of 0.07701688 for accelerometer and 0.00025556 for gyroscope [19]. A detailed discussion regarding data fusion using Kalman algorithm was presented in our previous paper [20]. The output of Kalman filter is shown in Fig. 2

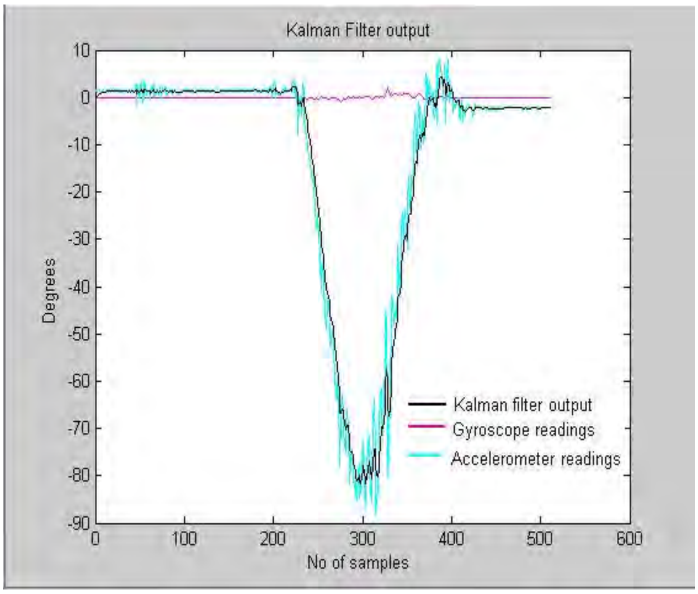


Figure 2. Kalman filter output

IV. FACTOR ANALYSIS METHOD

There exist many methods of data fusion in IMU; however as far as the authors are aware, utilizing factor analysis model in IMU data fusion has not been reported so far. Decentralized data fusion algorithm using factor analysis model was proposed [12]; in this algorithm, the features are extracted and processed individually and finally fused to obtain global estimates.

Factor analysis is a statistical method used to study the effect and interdependence of various factors within a system. The proposed algorithm has exploited guidelines and basics from various sources [21]. It fuses the accelerometer and gyroscope data in the (IMU); the data is similar to that used in the Kalman Filter method [20].

One of the main concerns in data fusion technique is the risk of producing fused estimates that are worse and lead to discontentment. The performance of the proposed algorithm was compared with Kalman method using estimation error analysis, as shown in Fig.3. Poor estimation could be responsible for poor performance of the data fusion system. It can be observed that, in the proposed algorithm, poor estimates, noise factor and other inaccuracies are more severe compared to the Kalman filter technique. Hence, there is scope for substantial refinement to improve the performance of the algorithm. Proper filtering is required at the feature generation stage to avoid the introduction of noise.

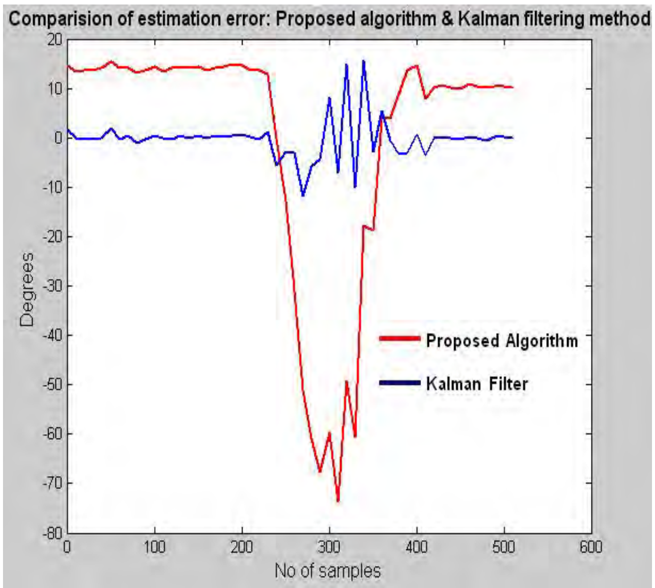


Figure 3. Comparision of estimation error in proposed method and Kalman filtering

V. PROFILER

Program analysis tools are extremely important for understanding the program behavior. Software developers, especially researchers need such tools to evaluate the performance of new architectures and techniques. Software writers need tools to analyze their programs and identify critical sections of code.

Matlab profile function helps to debug and optimize Matlab code files by tracking their execution time. For each Matlab function, Matlab sub function, or MEX-function in the file, profile records information about execution time, number of calls, parent functions, child functions, code line hit count and code line execution time. Profile history function records the exact sequence of function calls. The profile function records, by default, up to 1,000,000 function entry and exit events [22].

The profile timer clock function specifies the type of time to use; valid values for clock are:

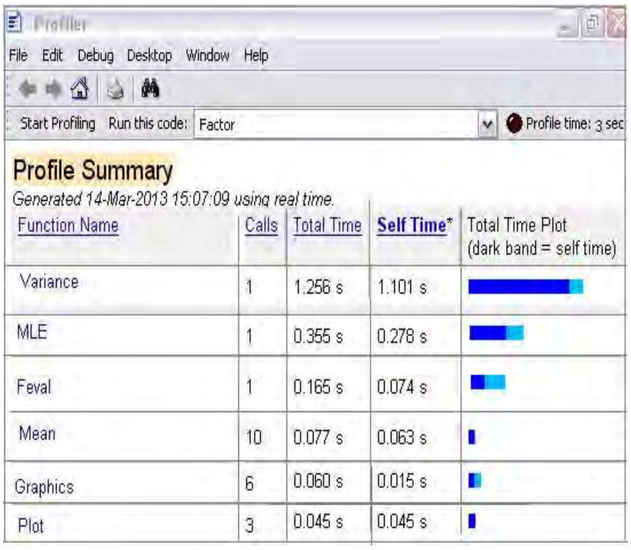
1) CPU — The Profiler uses computer time (the default setting).

2) REAL — The Profiler uses wall-clock time. (In our study, we have set the profiler to real time, which accounts for the actual time)

The profiler summarizes the time elapsed in each of the various functions of the program, as shown in Fig.4. It also provides the graph of time versus LOC (lines of code), as shown in Fig.5.

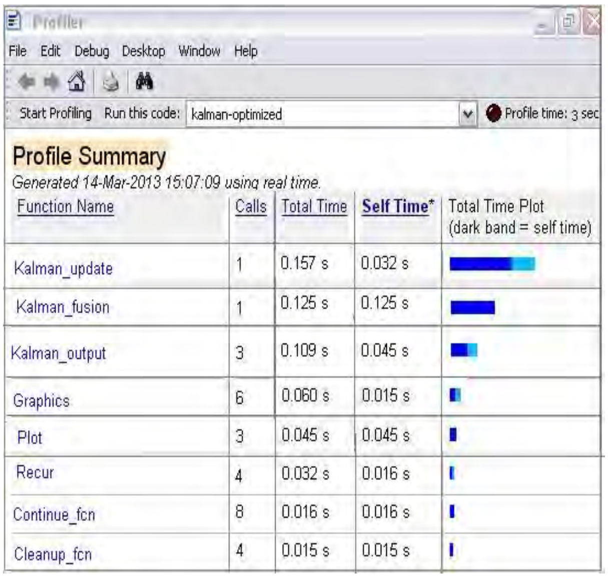
Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
Kalman_update	1	0.815 s	0.642 s	
Kalman_fusion	1	0.260 s	0.163 s	
Kalman_output	3	0.157 s	0.047 s	
Graphics	6	0.060 s	0.015 s	
Plot	3	0.045 s	0.045 s	

Figure 4a. Profiler summary for Kalman algorithm



Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
Variance	1	1.256 s	1.101 s	
MLE	1	0.355 s	0.278 s	
Feval	1	0.165 s	0.074 s	
Mean	10	0.077 s	0.063 s	
Graphics	6	0.060 s	0.015 s	
Plot	3	0.045 s	0.045 s	

Figure 4b. Profiler summary for Factor analysis method


Profiler
File Edit Debug Desktop Window Help
Start Profiling Run this code: kalman-optimized Profile time: 3 sec

Profile Summary

Generated 14-Mar-2013 15:07:09 using real time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
Kalman_update	1	0.157 s	0.032 s	
Kalman_fusion	1	0.125 s	0.125 s	
Kalman_output	3	0.109 s	0.045 s	
Graphics	6	0.060 s	0.015 s	
Plot	3	0.045 s	0.045 s	
Recur	4	0.032 s	0.016 s	
Continue_fcn	8	0.016 s	0.016 s	
Cleanup_fcn	4	0.015 s	0.015 s	

Figure 4c. Profiler summary for optimized Kalman method

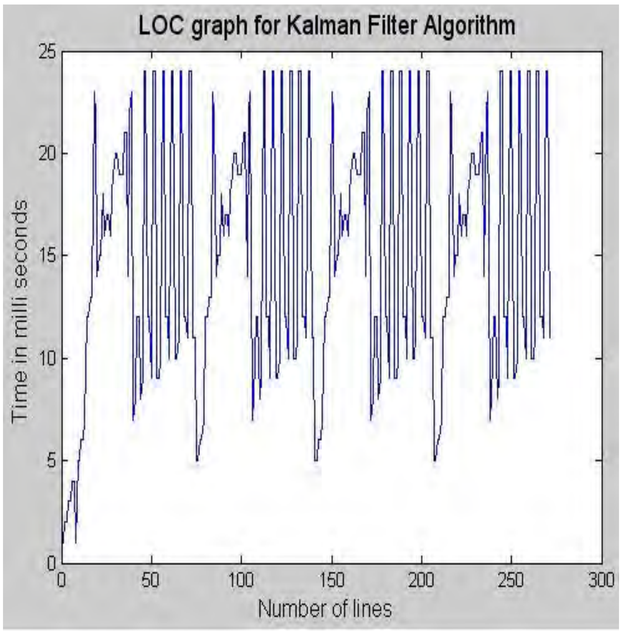


Figure 5a. Lines of code (LOC) graph for Kalman filter algorithm

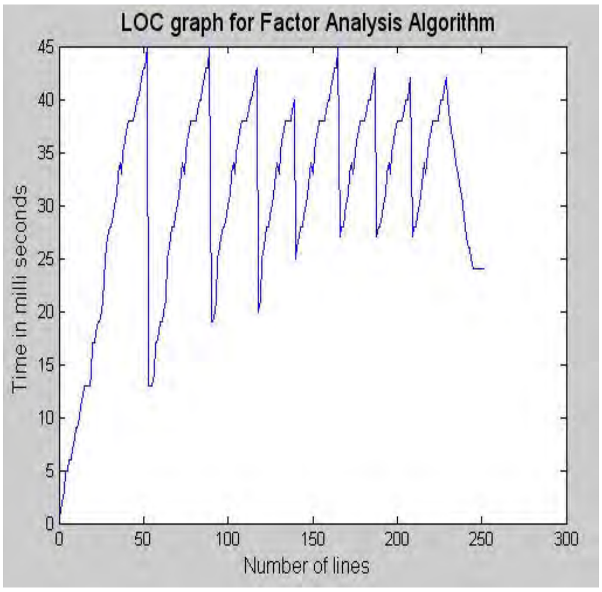


Figure 5b. LOC graph for Factor analysis algorithm

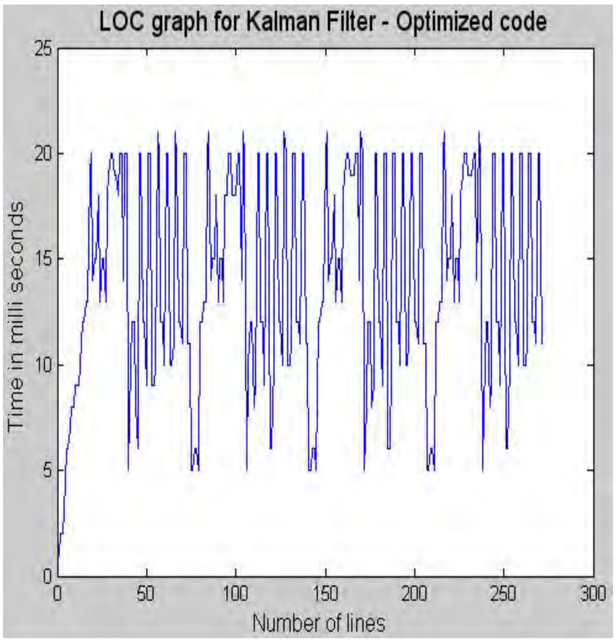


Figure 5c. LOC graph for optimized Kalman filter code

VI. FUNCTIONAL PROGRAMMING

Functional programming is a programming paradigm that treats computation as the evaluation of mathematical functions and avoids state and mutable data. It emphasizes the application of functions, in contrast to the imperative programming style, which emphasizes changes in state [23].

By virtue of functional programming, one can write the functions that operate on other functions; thus the peculiar style of program increases the efficiency of code. Functions can be passed as arguments in functional programming. It can be implemented using Anonymous functions which are powerful functions written smartly and precisely. Matlab provides some basic powerful functions for functional programmers [22].

In order to optimize the Kalman filter algorithm, we changed the programming style a little, rather than changing the whole code. The recursion and conditional-if statements were rewritten using anonymous functions, |recur| and an inline-if |iff| functions. Seeking better looping efficiency ‘For loop’ (Fig. 6) was replaced by ‘while-do’ loop (Fig.7).

```
for (i = 0; i < sample_count; ++i) % For loop
iteration;  sample_count=512;
{
 % Get the gyro and accelerometer input.
 gyro_input = sample_data[i][0];
 accel_input = sample_data[i][1];

% Update the Kalman filter and get the estimates.
 kalman_update=feval((gyro_input, accel_input);

 % Kalman filter output and get the output.
 kalman_output = kalman_fusion(gyro_input,
accel_input);
     end
    }
```

Figure 6. A Sample code using for-loop in Kalman Filter algorithm

```
{
  while (sample_count = @(x, continue_fcn, cleanup-
fcn))
% Do while loop iteration;  sample_count=512;
  recur(@(gyro_input, accel_input)    % Make the
recursion.
  (continue_fcn(x), @(z) ...          %   Run the
continue_fcn, save as z.
 iif(cont(z), @() sample_count(sample_count, z), ... %
Continue?  sample_count =512;
    true,    cleanup(x))),... % Stop, clean up.
   z);   % Save Final value

 % Update the Kalman filter and get the estimates.
kalman_update = @(z);

% Kalman filter output and get the output.
 kalman_output= @(z);
  end
  }
```

Figure 7.A Sample code using while-do loop implemented using functional programming (optimized code)

Table 1. depicts minimum and average times of all the three programs. Obviously, Kalman algorithm is better than the proposed factor analysis method. The drastic reduction in execution time of optimized Kalman code algorithm affirms the potential of functional programming in the coding phase.

TABLE I. COMPARISON OF EXECUTION TIME

Algorithm	Minimum Time	Average Time
Kalman Filter	1.00891	1.00995
Factor Analysis	2.76464	2.76552
Kalman Optimized code	0.42996	0.43080

VII. CONCLUSION

Multisensor data fusion has demonstrated significant advantages in various engineering applications by improving accuracy and precision in the evaluation process. The data fusion algorithm plays significant role in achieving reasonable performance. In other words, the most crucial issue in data fusion is the selection of algorithms/techniques. Thus, there continues to be controversies in the data fusion literature, as to which algorithm is the best, optimal or robust. In this paper, the development and comparison of two algorithms for a common task are presented. As the Kalman filter was found to be superior over the proposed Factor analysis method, it was optimized in the code development phase by using functional programming technique and anonymous functions. The optimized code successfully reduced the execution time without sacrificing the data fusion performance. The future work is to test the optimized code with various input samples and check the scalability.

REFERENCES

[1] D.L Hall, and J. Llinas, Introduction to multisensor data fusion, Handbook of multisensor data fusion, 1st ed., Boca Raton: CRC press pp.1-15, 2001.
[2] M. Kokar and K. Kim, “Review of multisensor data fusion architectures and techniques”, Proceedings of the international symposium on intelligent control, pp.261–266, 1993, Piscataway, Chicago.
[3] R. E. Kalman, “A new approach to linear filtering and prediction problems”, Journal of Basic Engineering, vol. 82, pp. 35-45, 1960.
[4] S. O. H. Madgwick, An efficient orientation filter for inertial and inertial/magnetic sensor arrays, Technical report, University of Bristol University, UK, 2010.
[5] D.L Hall, “Implementation of data fusion system”, Proceedings of the NATO advance research workshop on multisensor data fusion, June 2000, pp. 416–433, Pitlochry, Perthshire, Scotland.
[6] L.A. Klein, Sensor and Data Fusion Concepts and Applications, Bellingham: SPIE Optical Engineering Press, 1st ed., pp.51-63, 1993.
[7] H. Durrant-Whyte, A Beginner's Guide to Decentralised Data Fusion, Technical Document of Australian Centre for Field Robotics, University of Sydney, Australia, 2000.
[8] K. Feng., X. Zhou., Q. Zhang., L. Duan, “A new decentralized data fusion algorithm with feedback framework based on the covariance intersection method”, Proceedings of International Conference Computer Application and System Modeling, vol.9, pp.338-341, Oct. 2010, Taiyuan, China.
[9] C. Zhang and H. Wang, “Decentralized Multi-Sensor Data Fusion Algorithm Using Information Filter”, Proceedings of International Conference on Measuring Technology and Mechatronics Automation, March 2010 , pp. 890 - 893 , Changsha City, China.
[10] F. Zha, J. Xu, B. Hu, F. Qin, “Error Analysis for SINS with Different IMU Rotation Scheme”, Proceedings of 2nd International Asia Conference on Informatics in Control, Automation and Robotics, pp.422-425, 2010, China.
[11] V. Saini, S.C. Rana, M..M. Kuber, “Online Estimation of State Space Error Model for MEMS IMU”, Journal of Modelling and Simulation of Systems, vol.1, pp. 219-225, 2010.
[12] S.A.Quadri and O. Sidek, “Decentralized Data Fusion Algorithm Using Factor Analysis Model”, Applied Mechanics and Materials, vol. 278-280, pp.1182-1186, 2013.
[13] P. Hsieh, (wwwdotazillionmonkeysdotcom/qed/optimizedothtml)
[14] O. Sidek and S.A.Quadri, “A review of data fusion models and systems”, International Journal of Image and Data Fusion, vol. 3, no.1, pp.3-21, 2012.
[15] Mark Looney, A simple calibration for MEMS gyroscopes, 28 EDN EUROPE Magazine, Issue July 2010, wwwdotedndoteuropedotcom, (accessed March 2013)
[16] A.D King, “Inertial Navigation – Forty Years of Evolution”, GEC Review, vol. 13, no. 3, pp.140-149, 1998.
[17] B. Anderson and J. Moore, Optimal Filtering, NJ: Prentice Hall, 1990.
[18] M. S. Grewal and A. P. Andrews, Kalman Filtering Theory and Practice, NJ: Prentice Hall, 1993.
[19] htp://home.comcast.net/~michael.p.thompson/kalman/kalman_test2.c (accessed March 2013)
[20] S.A.Quadri and O. Sidek, “Factors affecting data fusion performance in an inertial measurement unit”, (Accepted and in process of publication in Journal of Control Engineering and Technology, 2013)
[21] T. Jing, “An Algorithm for estimating signals using factor analysis model”, Proceedings of International Conference on Circuits and Systems, pp. 358-360, 1991, Shenzhen, China.
[22] Matlab central(wwwdotmathworksdotcom)
[23] B.W. Kernighan and R. Pike, The Practice of Programming, UK: Addison-Wesley; 1st ed., pp. 110-118, 1999.