EXECUTION TIME OF QUEST ALGORITHM IN EMBEDDED PROCESSORS

Nelson S. Takahashi, seiji8503@gmail.com
André L. S. M. de Souza, andresouza1000@gmail.com
Marcelo C. Tosin, mctosin@uel.br
State University of Londrina (UEL), Londrina, PR, Brazil

Abstract. *Robustness and speed of optimal quaternion attitude estimation algorithms has been issued by several works using different criteria and environments. This work provides QUEST algorithm execution times in two embedded platforms, HCS12 and ARM Cortex M-3, not to contribute with the discussion of which algorithm is the fastest, but with the purpose to serve as orientation for general applications attitude determination systems designers, helping them to optmize tradeoff between system performance and cost. The algorithm was compiled using embedded software development tools and execution times were first represented as the number of clock cycles and then converted to execution time using 25 MHz clock frequency. To verify data, the algorithm was also implemented in hardware with single and double precision floating point operations at different clock rates. Execution times for different stages of the algorithm were also measured. HCS12 was slower than ARM Cortex M-3, although this difference was smaller for single precision. Major advantage of ARM Cortex M-3 over HCS12 was due to its maximum core clock frequency.*

Keywords*: attitude estimation, QUEST, embedded systems, algorithm speed, algorithm execution time*

1. INTRODUCTION

Attitude determination systems can be applied in several areas, such as biomedics and robotics and are underlying navigation and control systems, including spacecraft's. The attitude determination process referred in this work can be simply divided into three parts. First one is obtaining the reference and observation vectors from sensor signals, from star trackers, magnetometers or inertial sensors; second one is the computation of the attitude representation using these vectors; and the last one is sensor data fusion and correction and error minimization using dynamics models. In navigation systems, a very small error of attitude can lead to an extremely large deviation of the vehicle position after a certain time interval. To increase reliability of data, several methods have been developed related to sensors data correction and calibration and to take into account systems dynamics, using Kalman filters in different forms and with different parameters. Attitude can be represented in several ways, leading thus to different methods of attitude estimation from reference and observation vectors, such as TRIAD algorithm and its variations, which result in the Direct Cosine Matrix (DCM), the Modified Rodrigues Parameters (MRP) algorithm, which uses the MRP or Gibbs vectors, and algorithms that use quaternion. Quaternion is a very convenient attitude representation, as it has no singularities and does not require trigonometric or transcendental functions, although it has signal ambiguities.

Markley presented a review of the main quaternion attitude estimation algorithms and the Singular Value Decomposition (SVD) algorithm including comparisons among them related to robustness, the correct estimation free of singularities in any situations, and speed. All the presented algorithms have the objective of minimizing Wahba's loss function. The algorithms can be classified into two categories, in this work denoted as the slow and the fast algorithms. Slow algorithms, such as q-Method and SVD, obtain the attitude using standard algebraic algorithms. SVD algorithm, for example, uses SVD decomposition of B matrix, the attitude profile matrix obtained by reorganizing the loss function to gain function, and a modified diagonal matrix to obtain the DCM. The q-Method algorithm uses matrix B and other data to compose the K matrix. The optimal quaternion, then, is the eigenvector corresponding to the largest positive eigenvalue of matrix K. These algorithms are extremely stable but computationally burdensome and slower than the fast algorithms since they perform operations that might not be totally needed to solve Wahba's problem, which depends also on software libraries used. The fast algorithms perform mathematical manipulations in order to reduce calculations involved in obtaining the largest eigenvalue of K and its corresponding eigenvector. The main fast algorithms are QUEST, ESOQ-1, ESOQ-2 and FOAM, all well described in Markley's review. In that work, QUEST algorithm was regarded slower than ESOQ algorithms, using the number of floating points operation criteria, and less robust, not converging in a determined situation. However, Cheng and Shuster's subsequent works have showed that a trivial rearrangement in the eigenvalue characteristic polynomial makes QUEST as robust as other fast algorithms. Besides, the situation in which QUEST did not converge was shown to be impossible with existing sensors in a reasonable system design and cannot be regarded as a parameter for choosing an algorithm or another. Cheng and Shuster showed also that measurement of an algorithm's speed using the number of floating point operations does not provide a consistent parameter and that MATLAB, due to accelerators and implementation details of floating point operations counters, may not be an adequate platform to measure speed of algorithms, which is best represented as execution time. To evaluate these parameters, all the algorithms covered by the Markley's review were implemented in C language and

executed in a desktop computer, with the algebraic method implemented using the GNU Scientific Libraries (GSL). Difference between QUEST and ESOQ-2 algorithm, the fastest, were shown to be very small and unimportant for the current processors. With this, standardizing slow algorithms, especially q-Method, instead of developing new faster algorithms might not be an unreasoned decision, since the difference of execution time of slow and fast algorithms tends to become increasingly smaller with the evolution of computers' processing power.

The purpose of this work is to provide information one step further in algorithm speed, not to contribute with discussions of which one is the fastest, but to serve as orientation to attitude determination system designers for general applications, quantifying the execution time in embedded platforms. QUEST algorithm was the chosen one for it has been the most widely used. Also it is the ideal representative of fast algorithms, since the execution time difference between QUEST and ESOQ algorithms is very small. The speed of QUEST algorithm, including the computation of the covariance matrix, for two observation and reference vectors were measured as number of clock cycles in HCS12 and Advanced RISC Machine (ARM) Cortex M-3 processor cores, an information very close to a real application environment. The execution time, then, was calculated with a core system clock of 25 MHz. Furthermore, for the HCS12 platform, clock cycles for determined steps of the algorithm were counted, allowing evaluation of execution time for different stages of the algorithm. Also, the difference of speed for single precision, IEEE 32 bits, and double precision, IEEE 64 bits, floating point operations was measured. Finally, to verify the estimated times, QUEST algorithm in single and double precision was implemented in hardware and the real execution time was measured through a microcontroller output pin signal. Knowing the execution time of an attitude estimation algorithm in distinct platforms, a system designer will be able to define the most convenient and effective architecture to fulfill the requirements of his system, optimizing tradeoff between system performance and cost.

2. IMPLEMENTATION

QUEST algorithm considerably reduces the complexity of the attitude estimation process, using the Gibbs vector definition to present an alternative way to compute the largest eigenvalue of K matrix. The corresponding eigenvector also is obtained through a straightforward way, as well as the covariance matrix. In this work, QUEST code was made in structured C language. No external libraries like GSL were used to reduce code size and complexity. To do so, the algorithm was restricted to rigid body and has three degrees of freedom. This restraint simplifies burdensome matrix calculations, since all matrixes inside the algorithm are square and have order three, independent of the number of measurements. It allows matrix inversion and determinant to be obtained through straightforward expressions in non-recursive functions. Matrixes were dynamically allocated in heap memory in double pointer format, to allow scalabity for higher number of vector observations. Heap memory used to store matrix data was not optimized, being freed at the end of the algorithm. Inputs of the algorithms are the matrix V, which columns are the reference vectors, matrix W, which columns are the observation vectors, and, σ, the measurement noise vector. Outputs are the q_{opt}, the optimal quaternion and P, the covariance matrix. The method of rotational sequences and TASTE test were not implemented.

2.1. SOFTWARE

The implemented QUEST algorithm flowchart is shown in Fig 1. The algorithm begins with the microcontroller output pin inversion to indicate that one execution of the algorithm was made. Input data were assigned directly. Measurement noise is then used to compute the weights vector which is used to compute the Attitude Profile Matrix, B. The next step is to compute matrix S, vector Z and quantities σ, Δ, and κ. The quantity κ is the sum of the diagonal elements of the adjoint matrix of S, and is obtained explicitly in terms of the elements of S. The highest eigenvalue is computed followed by the covariance matrix, P_{QQ}. Both were implemented together since the form of their equations depend on the number of measurements, in this case, two. Finally, the optimal quaternion is computed with the elements of vector X and the algorithm starts again.

The algorithm was first implemented in MATLAB, then implemented in Microsoft Visual C++ 2008 and then compiled using embedded software tools Metrowerks CodeWarrior 5.9.0 for HCS12 platform and Keil uVision 3 with RealVision ARM C compiler for the ARM Cortex-M3 platform. Breakpoints were inserted into specific points of the algorithm compiled by the CodeWarrior, allowing the evaluation of execution clock cycles of each stage of the algorithm. Besides, the same software was compiled for single precision and double precision floating point operations. To minimize performance differences due to implementation details of specific embedded software libraries, which could not be quantified, external libraries were not used except for the *malloc()* and *sqrt()* functions, which respectively perform heap memory allocation and square root.

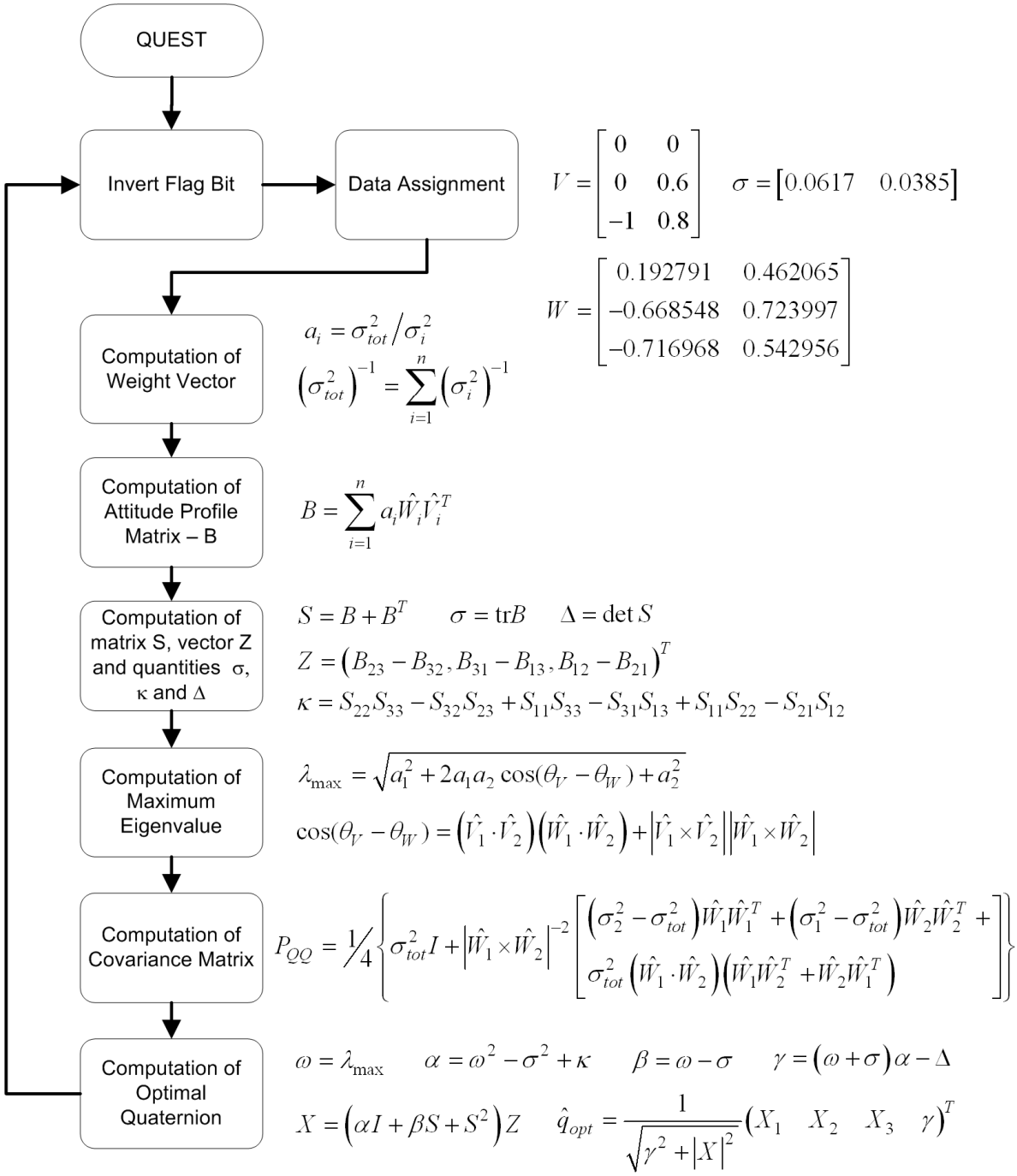

Figure 1. Implemented QUEST algorithm flowchart.

HCS12 is a Complex Instructions Set Computer (CISC) architecture, uses a three-stage 16 bits instruction queue always finishing the execution of one instruction before starting another one and has 16 bits registers. The basic instructions have 8 or 16 bits with operands also of 8 or 16 bits, or even 32 bits to extended division operation. The HCS12 microcontroller used in this work was Freescale MC9S12NE64, which has 64 Kbytes of Flash, 8 Kbytes of RAM and maximum operation frequency of 25 MHz. Instructions execution time varies considerably depending on the instruction or the operand. Basic arithmetic instructions, such as sum, subtraction and multiplication, use an average of 6 clock cycles, with 16 bits operand, although signed extended division and extended multiply and accumulate take much more cycles to execute, respectively 15 and 22 clock cycles.

ARM Cortex M-3 is based on Reduced Instructions Set Computer (RISC) architecture ARM version 7 M profile, focused on microcontrolled applications. The instructions are executed in three-stage pipelines and registers have 32 bits. The Thumb-2 instruction set supports both 32 bits and 16 bits instructions. The microcontroller used was ST Microelectronics ARM Cortex M-3 STM32F103ZE, which supports clocks of up to 72 MHz, has 512 Kbytes of Flash, 64 Kbytes of SRAM and 1.25 DMIPS/MHz (Drystone 2.1 Millions Instructions per Second) with SRAM wait time 0.

Multiplications take 1 to 4 clock cycles, depending on the number of significant digits, multiplications resulting in 64 bits data take 3 to 7 clock cycles and divisions 2 to 12 cycles, depending on the size difference between dividend and divisor. Both architectures do not perform floating-point operations. These operations are emulated by software. Thus, the results of the experiments made in this work are affected not only by the performance of the processor architectures but also by the performance of the floating point operations libraries implementation of each embedded software development tool. Although, specific floating point libraries performance aspects were not evaluated.

2.2. HARDWARE

To verify the estimated data, real measurements of algorithm speed were performed after embedding it into microcontrollers. The processor core clocks were configured to be the same, 24 MHz, and an output pin was set to invert its value every time the algorithm reached its start point inside an infinite loop. The only software being executed by the microcontroller, besides the startup code, was the QUEST algorithm. The output pin signal was sampled with an oscilloscope and its frequency, which represents the number of times QUEST algorithm was executed in one second, was measured. Inverting this value it is possible to obtain the execution time of one QUEST run. Each signal transition corresponds to one execution. Therefore, each signal period corresponds to two algorithm runs and the measured frequency was doubled before inversion to obtain the execution time. The ARM Cortex M-3 processor' core clock was later configured to 72 MHz to evaluate its maximum speed. The maximum core clock frequency is a great advantage of ARM Cortex M-3 over HCS12, which supports a 25 MHz maximum core clock, and this advantage for this application was also quantified.

3. RESULTS

Table 1. Clock cycles and estimated execution times for HCS12.

HCS12 Architecture	**Single Precision (IEEE 32-bits)**		**Double Precision (IEEE 64-bits)**	
Main Functions	**Clock Cycles**	**Estimated Time**	**Clock Cycles**	**Estimated Time**
Startup	10,093	403.72 us	10,094	403.76 us
Quest	191,621	7,664.84 us	1,405,759	56,230.36 us
Stages of the Algorithm				
Variables assignments	1,945	288.84 us	2,048	81,92 us
Computation of weight vector	7,221	642.96 us	44,331	1,773.24 us
Computation of matrix B	16,074	642.96 us	236,812	9,472.48 us
Computation of matrix S, vector Z and quantities σ, κ and Δ	17,234	689.36 us	81,771	3,270.84 us
Computation of maximum eigenvalue	30,558	1,222.32 us	196,028	7,841.12 us
Computation of covariance matrix	63,733	2,549.32 us	535,088	21,403.52 us
Computation of optimal quaternion	46,437	1,857.48 us	301,261	12,050.44 us

Table 1 shows clock cycles of QUEST algorithm in HCS12. Estimated execution times were determined for 25 MHz core clock, since this is the maximum core clock of the HCS12. Startup is the number of cycles required to initialize the microcontroller functions before the start of QUEST. Variables assignments is the assignment of matrix and vector data, which would be passed as parameters of a function call in a complete attitude determination system. The inverse of the execution time would lead to a theoretical number of attitude estimates per second, about 130 for single precision and 17 for double precision, although all the attitude estimation process is made up of several other procedures. Still, it was possible to evaluate that 33% of the execution time in single precision was spent to compute the covariance matrix, 24% to compute the optimal quaternion and 15% computing the highest eigenvalue. In execution time by stages, memory release routines were not taken into account.

Table 2. Clock cycles and estimated execution times for ARM Cortex M-3.

ARM Cortex M-3 Architecture	**Single Precision (IEEE 32-bits)**		**Double Precision (IEEE 64-bits)**	
	Clock Cycles	**Estimated Time**	**Clock Cycles**	**Estimated Time**
Quest	141,600	5,664 us	524,495	20,979.8 us

Table 2 shows clock cycles of QUEST algorithm in ARM Cortex M-3. As the instructions are executed in pipelines and the compiler optimizes code implementation and execution, feature set to maximum, level 3, it was not possible to evaluate cycles per stage. The clock cycles number was obtained from the core peripheral timer System Tick Timer, measuring the difference between its value at the beginning and at the end of the algorithm. The ratio of HCS12 clock cycles over ARM Cortex M-3 was about 1.35. For double precision, this ratio was about 2.68. With 32 bits registers, ARM Cortex M-3 requires less clock cycles to perform arithmetic instructions, an advantage that becomes more evident

in double precision, as the amount of arithmetic operations increase. The inverse of execution time for 25 MHz leads a theoretical number of about 176 attitude estimates per second for single precision and 47 for double precision.

Table 3. Measured frequency and execution times for both architectures at different core clock rates.

Architecture	**HCS12**		**ARM Cortex M-3**			
Core Clock	**24 MHz**		**24 MHz**		**72 MHz**	
Precision	**Single**	**Double**	**Single**	**Double**	**Single**	**Double**
Frequency	118.62 Hz	16.946 Hz	135.86 Hz	41.5 Hz	409.2 Hz	131.4 Hz
Execution Time	8.430 ms	59.011 ms	7.631 ms	24.096 ms	2.444 ms	7.610 ms
HCS12/ARM Ratio	1	1	1.10	2.45	3.45	7.75

Table 3 shows the measured frequency and execution times for QUEST running at both platforms at different clock rates. The processor core clocks were configured to be 24 MHz, to allow a better performance comparison, using external 8 MHz crystals. For ARM Cortex M-3, to verify the actual core clock, this signal (SYSCLK) was derived to the MCO pin and measured. In Tab. 3, frequency is double of the measured frequency of the algorithm execution output pin, since each algorithm execution corresponds to one pin inversion and, therefore, there are two executions in one signal period, as cited before. Ratio HCS12/ARM is the ratio of execution time in HCS12 platform over respective execution time in ARM Cortex M-3 platform with the same precision. With the same clock speed, 24 MHz, which is very close to the maximum HCS12 core clock rate, difference between HCS12 and ARM Cortex M-3 for single precision was very small. For double precision HCS12 is almost 2.5 times slower than ARM Cortex M-3. Although, the major advantage of ARM Cortex M-3 is relative to its maximum core clock frequency, 72 MHz. In this case, for single precision, ARM Cortex M-3 was almost 3.5 times faster than HCS12 running close to its maximum core frequency. For double precision, this difference is even higher, with ARM Cortex M-3 being more than 7.5 times faster than HCS12.

From these data, also, execution time of q-Method in the same environment might be roughly previewed using the ratio between QUEST and q-Method obtained by Cheng and Shuster, which is 60. This would lead to q-Method execution times of 505,8 ms for single precision in HCS12 platform or 456,6 ms for double precision in ARM Cortex M-3 platform with core clock frequency of 72 MHz, although that ratio did not include the computation of covariance matrix.

4. CONCLUSION

Execution times of QUEST algorithm for two vector observations were presented in number of clock cycles and execution time for two embedded processor cores, HCS12 and ARM Cortex M-3. The quantities were obtained from the respective embedded software compilers and through hardware implementation. ARM Cortex M-3 is a faster platform than HCS12. However, this difference is very small for single precision and same core clock rate. For double precision, this difference is very higher, but the major advantage of ARM Cortex M-3 over HCS12 is its maximum core clock speed. According to results, HCS12 might be a better cost-effective choice for applications that require moderate number of attitude estimates per second with low precision, while ARM Cortex M-3 might be the best choice for systems that require high speed and high precision. Future work is to perform the same analysis for the fastest of the slow algorithms, the q-Method, and compare it with the fast algorithms representative, QUEST. The information presented in this work can be helpful for general applications attitude determination system designer's decisions and contribute with the discussion of standardizing q-Method for optimal attitude estimation in embedded systems.

5. ACKNOWLEDGEMENTS

The authors acknowledge AEB, CNPq, CAPES and Fundação Araucária for supporting this work.

6. REFERENCES

Bachmann, E.R., Mcghee, R.B., Yun, X. et al, 2001, "Inertial and magnetic posture tracking for inserting humans into networked virtual environments", Proceedings of the ACM symposium on Virtual reality software and technology, pp. 9-16.

Cheng, Y., Shuster, M.D., 2007a, "Robustness and Accuracy of the QUEST Algorithm", Paper No. AAS-07-102, AAS/AIAA 17th Space Flight Mechanics Meeting, Proceedings of Advances in the Astronautical Sciences, Vol. 127, Sedona, Arizona, pp. 41–61.

Cheng, Y., Shuster, M.D., 2007b, "The Speed of Attitude Estimation," Paper No. AAS-07-105, AAS/AIAA 17th Space Flight Mechanics Meeting, Proceedings of Advances in the Astronautical Sciences, Vol. 127, Sedona, Arizona, pp. 101–116.

Crassidis, J. L., Markley, F. L, 2003, « Unscented Filtering for Spacecraft Attitude Estimation", Journal of Guidance, Control, and Dynamics, Vol. 23, No. 4, pp. 536-542.

Granziera, F., Lopes, R.V.F., Tosin, M.C., 2007, "The attitude determination problem from two reference vectors - a description of the TRIAD algorithm and its attitude covariance matrix". Semina, Vol. 28, No. 1, pp. 21-35.

Idan, M, 1996, "Estimation of Rodrigues Parameters from Vector Observations", IEEE Transactions on Aerospace and Electronic Systems, Vol. 32, No. 2, pp. 578-586.

Keat, J., 1977, "Analysis of Least Squares Attitude Determination Routine (DOAOP)", Computer Sciences Corporation, CSC/TM-77/6034.

Lawrence, A., 1992, "Modern Inertial Technology: Navigation, Guidance, and Control". Ed. Springer, New York, United States of America, 278 p.

Lefferts, E.J., Markley, F.L., Shuster, M.D., 1982, "Kalman filtering for spacecraft attitude estimation", Journal of Guidance, Vol .5, No .5, pp.417-429.

Lotters, J. C., Schipper, J., Veltink, P.H., et al, 1998, "Procedure for in-use calibration of triaxial accelerometers in medical applications", Sensors and Actuators A, Vol. 68, No. 1-3, pp. 221-228.

Markley, F.L., 1988, "Attitude Determination Using Vector Observations and the Singular Value Decomposition", Journal of the Astronautical Sciences, Vol. 36, No. 3, pp. 245-258.

Markley, F.L., Mortari, D, 2000, "Quaternion Attitude Estimation Using Vector Observations", The Journal of the Astronautical Sciences, Vol. 48, No. 2, pp. 359-380.

Mortari, D., 1997, "ESOQ: A Closed-Form Solution to the Wahba Problem", The Journal of the Astronautical Sciences, Vol. 45, No.2, pp. 195-204.

Mortari, D., 2000. "Second Estimator of the Optimal Quaternion", Journal of Guidance, Control, and Dynamics, Vol. 23, No. 5, pp. 885-888.

Sabatini, A.M, 2006, "Quaternion-Based Extended Kalman Filter for Determining Orientation by Inertial and Magnetic Sensing", IEEE Transactions on Biomedical Engineering, Vol . 53, No .7, pp. 1346-1356.

Shuster, M.D., 1978, "Algorithms for Determining Optimal Attitude Solutions", Computer Sciences Corporation, CSC/TM-78/6056, 77 p.

Shuster, M.D., Oh, S.D., 1980, "Three-Axis Attitude Determination from Vector Observations", AIAA Journal of Guidance and Control, Vol. 4, No. 1. pp. 70-77.

Tanygin, S., Shuster, M.D., 2007, "The Many TRIAD Algorithms," Paper No. AAS-07-104, AAS/AIAA 17th Space Flight Mechanics Meeting, Proceedings: Advances in the Astronautical Sciences, Vol. 127, Sedona, Arizona, pp. 81–99.

Wahba, G., 1966, "A least squares estimate of satellite attitude", Siam Review, Philadelphia, Vol .8, No .3, pp. 384-386.

Wertz, R.D., 1978, "Spacecraft Attitude Determination and Control", Ed. D. Reidel, Dordrecht, Netherlands, 858 p.

Yiu, Joseph, 2007, "The Definitive Guide to the ARM Cortex-M3", Ed. Elsevier, Massachusetts, United States of America, 400 p.

Yun, X., Bachmann, E.R, 2006, "Design, Implementation, and Experimental Results of a Quaternion-Based Kalman Filter for Human Body Motion Tracking", IEEE Transactions on Robotics, Vol .22, No. 6, pp. 1216-1227.

7. RESPONSIBILITY NOTICE

The authors are the only responsibles for the material included in this paper.