Inertial Measurement Unit – Data Fusion and Visualization using MATLAB

R. Baranek*

**Brno University of Technology, Department of Control and Instrumentation, Brno, Czech Republic (e-mail: xbaran10@ stud.feec.vutbr.cz).*

Abstract: The present paper is concerned with the development of an algorithm for the processing of data from gyroscopes and accelerometers such that they together form an attitude sensor. The MATLAB programming environment (only MATLAB in the following) has been chosen for the development of this algorithm. The reason has been the ease with which codes can be created and debugged. With the aid of this programming environment it is also possible to use the not very well known but still simple method of visualizing in real time individual signals from primary sensors as well as the algorithm output, i.e. the sensor attitude in space. One chapter of the paper is therefore devoted to the method of real-time visualization in MATLAB. Signals from the three-axis gyroscope are taken as the primary source of information. A disadvantage of gyroscopes is the incessant small unpredictable change of zero position, which is responsible for the error in attitude determination increasing with time. Using signals from the three-axis accelerometer, the presented algorithm tries to minimize this error. Experimental results for the presented algorithm are given in the paper.

Keywords: Inertial measurement unit, MATLAB, Data fusion, MEMS, Gyroscope, Error correction.

1. INTRODUCTION

Attitude sensors are an ever increasingly used source of information not only in robotics; they find application primarily in unmanned aerial vehicles (UAV) (Kim et al, 2006), where they serve automatic stabilization of position, which is indispensable for these applications. Most attitude sensors are based on the sophisticated processing of data from gyroscopes, accelerometers and magnetometers (Pounds et al, 2007). A great number of final sensors are available which encompass data processing, with information about attitude being the output. The price of these devices is, however, comparatively high. The elementary sensors themselves (gyroscopes, accelerometers and magnetometers) can be had at a reasonable price. Setting up one's own algorithm of data processing is thus frequently the only affordable way how to equip with the attitude sensor a given application.

In its final shape, an attitude sensor is most often based on individual elementary sensors (gyroscopes, accelerometers and magnetometers) and a microcontroller, which implements a data fusion algorithm and provides communication with the surroundings. Since in creating such algorithms the time spent on the development is an important factor, it is not suitable to create the algorithm right on the microcontroller. It is more appropriate to simply transfer raw data from the sensors to the computer, where the processing proceeds in some development environment that enables more advanced work with codes and also allows simple debugging. In this case creating the algorithm is greatly speeded up since transferring the code from the given environment into the microcontroller code is simple.

MATLAB is one of the possible alternatives. An advantage is, above all, the simplicity of writing the code since most mathematical functions and operations are part of the environment. Another advantage consists in the possibility to create GUI (Graphical User Interface) applications, which can be used to visualize in real time individual signals and algorithm outputs. Using MATLAB, there is yet another important factor, namely the possibility to communicate via a serial port. This port is used to transfer raw data from the sensors to the computer.

The above algorithm for data fusion is based on only the signals from three-axis gyroscope and accelerometer. The primary source of information is the signal from three-axis gyroscope. A disadvantage of gyroscopes is the incessant unpredictable change in zero position, the so-called drift. Since for the purpose of obtaining attitude information it is necessary to integrate the gyroscope signal, the error caused by the change in zero position increases with increasing time. The three-axis accelerometer is used to eliminate this phenomenon.

1.1 Organization of Paper

The paper is organized as follows. Chapter 2 describes the theory of calculating and representing spatial attitude when individual angular velocities are known. In Chapter 3 it is shown how, using MATLAB, the data measured can relatively easily be visualized in real time. Also described is the way the raw data from sensors are transferred to the computer via the serial port. The data fusion algorithm under development is described in Chapter 4. Chapter 5 gives experimental results for specific sensors.

2. THEORY OF ATTITUDE COMPUTATION AND REPRESENTATION

2.1 Methods of attitude representation

There are three most important methods of representing spatial attitude:

- using the Euler angles
- using the quaternions
- using the rotation matrix (matrix of directional cosines)

All these methods represent the attitude of a coordinate system related to the body under examination with respect to a reference coordinate system. The reference system is most often related to the surface of the earth.

The attitude of a coordinate system with respect to the reference system via the Euler angles is defined by a trio of numbers. By means of three elementary rotations by angles given by a trio of numbers (yaw, pitch and roll) about defined rotation axes the coordinate system of the body under examination can be obtained from the reference system. Using the Euler angles, any attitude of a coordinate system with respect to a reference coordinate system can be determined unambiguously. In real situations, however, we are faced with the opposite task, namely determining on the basis of a known attitude of the coordinate system the Euler angles and thus expressing the attitude. In this case the determination of angles is no longer generally unambiguous and there are attitudes in which only the difference or sum of two angles but not all three angles can be determined unambiguously. This phenomenon is commonly referred to as gimbal lock. The Euler angles are often used just because of their simplicity. The disadvantage is, however, exactly the gimbal lock phenomenon, which is absent in the following two methods.

Attitude representation with the aid of quaternions makes use of the fact that any attitude can be expressed by a rotation by a given angle about a certain rotation axis. The direction of the rotation axis and the size of the angle are encoded in a single vector, where the direction of this vector determines the rotation axis, and its size determines the angle. The quaternion is defined as a four-element vector:

$$q = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \cos(\mu/2) \\ (\mu_x/\mu)\sin(\mu/2) \\ (\mu_y/\mu)\sin(\mu/2) \\ (\mu_z/\mu)\sin(\mu/2) \end{bmatrix}, \tag{1}$$

where μ is the vector size giving the axis and angle of rotation, and μ_x, μ_y and μ_z are the individual components of this vector with respect to the reference coordinate system. Quaternions do not exhibit the gimbal lock, but their disadvantage consists in comparatively complex rules for computation since quaternions are the result of extending the complex numbers.

The last from the possibilities given is the representation of attitude by means of a rotation matrix. It is known from linear algebra that vectors can be expressed in various bases. Converting from one base to another is done using the transformation matrix:

$$\boldsymbol{a}' = C \cdot \boldsymbol{a}, \tag{2}$$

where a' is the vector expressed in the new base, a is the vector expressed in the old base, and $\mathbf{C}$ is the transformation matrix for the conversion from the old base to the new one. Rotation can also be represented by changing the base of the coordinate system. The transformation matrix describing rotation has, of course, some specific properties:

$$C^{-1} = C^{T}. \tag{3}$$

Since in rotation both the distances and angles are maintained, the transformation matrix remains orthonormal, i.e. individual rows or columns correspond to the vectors, which are unit vectors, mutually perpendicular. If three-dimensional space is considered, then the dimension of rotation matrix is 3x3. Individual elements of the rotation matrix have a simple geometric interpretation. The matrix elements give the cosine of the angle made by the base vectors of the original system and the base vectors of the rotated system of coordinates.

The last mentioned method of representing the attitude is used in the algorithm introduced in this paper. Except for some special cases (gimbal lock), there are conversion relations between all the three kinds of representation. Thus they are informationally equivalent. Which of the methods is more appropriate depends on the given application.

2.2 Development of rotation matrix in time

Rotation matrix expresses the mutual attitude of two coordinate systems. The mutual attitude and also the rotation matrix generally vary with time. The development of rotation matrix in time is given by the equation:

$$\dot{C} = C \cdot \Omega, \tag{4}$$

where C is the rotation matrix, the dot over the letter C stands for differentiation with respect to time, and

$$\Omega = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \tag{5}$$

where ω_x, ω_y and ω_z are the individual components of angular velocity relative to the base associated with the body being examined. To solve this equation, we must know the initial state of the rotation matrix and the continuous waveforms of individual angular velocities. In real applications, the computation is performed on a microcontroller or computer. The given equation must thus be discretized. Rewriting (4) we can derive the discrete recurrent relation for the computation of the rotation matrix when the individual angular velocities are known (Titterton & Weston, 2004):

$$C\big(T(n+1)\big) = C(Tn) \cdot (I + \Omega(Tn) \cdot T), \tag{6}$$

where I is the unit matrix, and T is the sampling period. Equation (6) represents the core of the algorithm for the processing of data from elementary sensors.

3. DATA ACQUSITION AND REAL-TIME VISUALIZATION USING MATLAB

MATLAB is a highly universal development tool, which enables creating classical algorithms that make use of a great number of embedded mathematical functions. Of course, it also enables creating the graphical user interface (GUI). Within these applications any data are simple to visualize in real time. This way of monitoring signals from gyroscopes and accelerometers can be of help when creating the whole algorithm and thus make the whole development more effective.

Another beneficial property of MATLAB is the possibility of controlling the serial port. Using this port, data can be easily and effectively transferred from sensors to the computer. On the hardware side, just a microcontroller with simple firmware is required which regularly reads data from the sensors and then encodes these data into an appropriate format suitable for transfer via the serial port, and sends these encoded data to the PC.

3.1 Real-time Data Visualization

Creating a GUI application forms the basis for visualizing data in real time. In MATLAB this is possible due to GUIDE, which is, to put it simply, an editor of the form window, which forms the graphical part of the application. With regard to visualizing data in real time, this form should contain at least the START and STOP push-buttons, and also the graph object, into which the time rate of the required data will be visualized. The functionality of the form window is assured by an attached m-file, which is a file containing the code of the MATLAB environment. In this file, functions are defined that handle the individual events of the elements of the form (click the push-button, etc.). Visualizing data in real time is taken to mean cyclic display of a certain value or several values with a sufficiently fast repeat period. Visualizing in real time is thus provided by an infinite cycle, which is triggered by the START push-button. The cycle can only be stopped using the STOP push-button. In the cycle body the following processes take place:

- data acquisition

- conversion or computation of other quantities

- visualization of obtained or computed data.

With regard to MATLAB, the following comments need to be taken into consideration for an optimum real-time visualization of data. The primary command for visualizing the graph is the `plot` function. Cyclic calling of this function would greatly strain the computer processor and could lead to an increase in the repeat cycle period. Since MATLAB is object-oriented, there is a more elegant way of updating data in a graph created by the plot function. An object created by this function has several properties that can be changed any time during the existence of this object. One of these properties is the visualized data themselves. Data can thus be updated by the simple command:

```
set (plot_object,‘YData‘, data);
```

where `data` is the variable containing updated data. Another important command for real-time visualization of data in MATLAB is the command `drawnow`. When this command is called, all the events waiting in the queue will be handled.

To give a simple example, an application can be created that will visualize in real time randomly generated numbers. The form window of an application created in this way is shown in Fig. 1. The code running in a loop for such an application looks like this:

```
data (:) = rand ( size (data));

set (plot_object,‘YData‘, data);

drawnow;
```

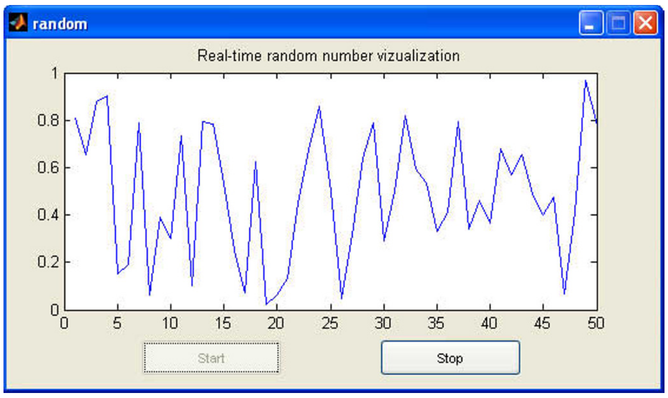

Fig. 1. Form window of an application visualizing random numbers in real time.

3.2 Data Acquisition

Although MATLAB contains toolboxes that enable acquiring data from special measuring cards, for less demanding applications there exists a simpler and cheaper way of sending data from an external device to MATLAB, namely via the serial port. What is concerned here is a common and relatively simple communication interface supported in various devices. MATLAB communicates with the serial port via an object. After initializing the object (setting the transfer speed, etc.) the serial port in MATLAB looks like a file. It can therefore be read from and written into. Since the serial port is adapted to sending individual ASCII characters, it is appropriate at the microcontroller end to convert somehow raw data to the text format. To convert the text back to numbers the MATLAB function `strread` is used.

Combining data acquisition and data visualization in real time can yield a suitable environment for the development of attitude sensor. The computation algorithm can be included in the visualization loop and thus its function can be monitored. The simultaneous view of raw and computed data enables easy removal of errors in the algorithm, which speeds up the overall development of the algorithm.

4. ALGORITHM OF COMPUTATION OF ATTITUDE FROM ELEMENTARY SENSORS

The algorithm for processing data from elementary sensors (gyroscopes, accelerometers) is based on the computation of a discrete recurrent formula for the time rate of rotation (6). Thus only signals from the three-axis gyroscope ω_x, ω_y and

ω_z (rad/s) need to be known. Since the given computation is performed in the discrete form, i.e. a certain approximation is performed, the condition of rotation matrix orthonormality is not satisfied. If this error were not corrected, then after several iterations the whole algorithm would provide absurd information. The reason is that in the derivation of formula for the time development the property of orthonormality is made use of.

In the correction of the non-orthonormality of the rotation matrix the following relations are used (Premerlani & Bizard):

$$E = \boldsymbol{x} \cdot \boldsymbol{y}, \tag{7}$$

$$\boldsymbol{x}_{new} = \boldsymbol{x}_{old} - \frac{E}{2}\boldsymbol{y}_{old}\ , \tag{8}$$

$$\boldsymbol{y}_{new} = \boldsymbol{y}_{old} - \frac{E}{2}\boldsymbol{x}_{old}\ , \tag{9}$$

where E is the scalar product of vectors **x** and **y,** which form the rows of the rotation matrix. The last row of the rotation matrix is obtained as the vector product of vectors **x** and **y**:

$$\boldsymbol{z}_{new} = \boldsymbol{x}_{new} \times \boldsymbol{y}_{new}. \tag{10}$$

Equations (7) – (10) correct the non-orthogonality error of individual vectors forming the rotation matrix. Furthermore, it is also necessary to correct the non-unit size of individual vectors:

$$\boldsymbol{x}_{norm} = \frac{1}{2}(3 - \boldsymbol{x}_{new} \cdot \boldsymbol{x}_{new})\boldsymbol{x}_{new}. \tag{11}$$

The same is also done for the remaining two vectors.

By periodical computation of the rotation matrix and correction to its non-orthonormality a reasonable response to changes in the sensor attitude can be obtained. In steady state, however, there is, due to the incessantly random change in the zero value of gyroscopes, a slow increase in the error between the actual and the sensor-determined attitude value. To be able to utilize the sensor, this drift of the zero value of gyroscope position must be corrected too.

There are many ways how to correct this error using magnetometers and accelerometers. An example can be seen in the application of the Kalman filter (Kang-hua et al, 2007). Implementing an algorithm with the Kalman filter and its very development are a comparatively complex matter. Another possibility is the design of a PI controller for gyroscope offsets (Premerlani & Bizard). In this case, use is made of signals from three-axis accelerometer, for example. If gravity force alone acted on the sensor, the signal from the accelerometer would correspond to the current sensor attitude (except rotation about the vertical axis). Using the deviation between of the attitude determined by means of accelerometer signal and the gyroscope attitude it is possible to determine the error, which is the input for the given PI controller. A problem arises if a force other than gravity force acts on the sensor. One possibility is to use auxiliary sensors (velocity, etc.) to estimate the size of acceleration not caused by gravitation and subtract it from the accelerometer value. But it is not always that auxiliary sensors can be used and therefore a PI controller with dynamically changing constants is introduced in this paper in contrast to (Premerlani & Bizard). It results from the accelerometer properties that the PI controller should be set such that it slowly follows the attitude given by the accelerometer. This will ensure that in cases of short-term action of non-gravity force the attitude given by the sensor will be close to the actual attitude and the long-term integration error will be eliminated.

Fig. 2 gives the block diagram of an algorithm in which the elimination of integration error of gyroscopes is included. The determination of errors in establishing the attitude, when the attitude determined by the accelerometer is taken as the reference attitude, proceeds in several steps. First, the vector product is determined of the normalized vector, which corresponds to the signals from the three-axis accelerometer, and the base vector of axis z, which is formed by the last column of the rotation matrix:

$$\boldsymbol{E} = \boldsymbol{a} \times \boldsymbol{z}, \tag{12}$$

where **a** is the normalized vector corresponding to the signal from the three-axis accelerometer. It is obvious from the above that if these two vectors are identical, i.e. the two attitudes match, the vector product is a zero vector and the error is also zero; in other words there is nothing to correct. But if, for example, due to the drift of the zero position of the gyroscope detecting the rotation about axis y, the attitude determined by gyroscopes is different from that determined by accelerometers, then the vector product for the determination of error has the direction of axis y. Thus the numerical expression of the error of zero positions of gyroscopes detecting the rotation about axes x and y is established by the scalar product of error vector **E** and the given base vector z of the rotation matrix:

$$errx = \boldsymbol{E} \cdot \boldsymbol{x}, \tag{13}$$

$$erry = \boldsymbol{E} \cdot \boldsymbol{y}. \tag{14}$$

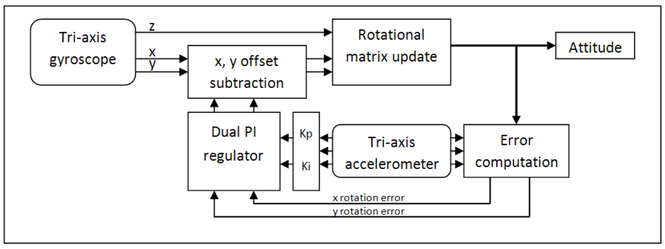

Fig. 2. Block diagram of algorithm with elimination of integration error of gyroscopes.

The values determined by (13) and (14) are inputs for PI controllers. Outputs from PI controllers are the zero values of gyroscopes. With each reading, these values are subtracted from the gyroscope values. In this way the monitoring of incessant changes in the zero position of gyroscopes is assured. With regard to the required properties of the monitoring of the attitude from accelerometer it is evident that it is no use implementing a typical PID controller. An output in the form of gyroscope offset value is determined as follows (given for axis x only):

$$offx_n = K_p \cdot errx_n + K_I \textstyle\sum_{i=1}^{n} errx_n. \tag{15}$$

To ensure higher precision, controller constants K_P a K_I are changed while the application is running. This is because if accelerating forces act on the sensor the data about the attitude from the accelerometer are irrelevant. Changing the controller constant in dependence on the absolute value of the acceleration vector can lead to a higher precision of the attitude determined. The constants are changed in accordance with the following relation (this holds for both constants):

$$K = K_{ref} \cdot e^{-|g-|a||}, \tag{16}$$

where g is gravity acceleration, and a is the size of the acceleration from the three-axis accelerometer. This relation is derived from the fact that if the constants were zero, the accelerometer would be of no effect on the attitude. The value of constants approximates infinitesimally the zero value when the size of acceleration vector approximates infinity.

5. EXPERIMENTAL RESULTS

The algorithm presented in this paper was tested using MEMS sensors L3G4200D and LSM303DLH (STMicroelectronics). The algorithm was implemented in MATLAB. A GUI application was also created in which individual signals from a three-axis gyroscope and accelerometer were visualized together with data about attitude, output and error of PI controllers. The GUI application window is shown in Fig. 3.

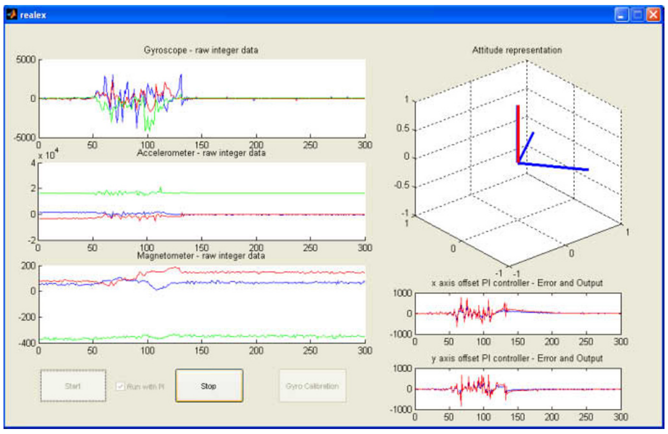
realex
Gyroscope - raw integer data
Accelerometer - raw integer data
Magnetometer - raw integer data
Attitude representation
x axis offset PI controller - Error and Output
y axis offset PI controller - Error and Output
Start
Run with PI
Stop
Gyro Calibration

Fig. 3. GUI application window.

Experimentally established controller constants were:

$K_{Pref} = 500$

$K_{Iref} = 0.1$

To characterize the algorithm precision, the attitude was measured via simple integration of signals from gyroscopes and then by using the above algorithm. The time development of the Euler angle for the pitch for the two cases as well as for the attitude determined by accelerometer is shown in Fig. 4.

6. CONCLUSIONS

The paper is concerned with an algorithm for the fusion of data from a three-axis accelerometer and gyroscope. The algorithm output is the spatial attitude of the sensor with respect to a reference coordinate system. Also described is an effective way of using MATLAB for an effective development of the whole algorithm. Real-time visualization of raw data from sensors is also outlined here. To conclude the paper, results are given from a real experiment where an algorithm for common integration of signals from gyroscopes is compared with an advanced algorithm that corrects the incessant change in the zero position of gyroscopes. It is shown that the algorithm presented in this paper is more precise than the common integration of signals from gyroscopes but the precision of attitude determination is still insufficient for application in unmanned aerial vehicles. The source of imprecision may be the non-corrected zero value of the gyroscope detecting rotations about the vertical axis. Correcting by means of a magnetometer the zero value of this gyroscope axis will therefore be the subject of future research.

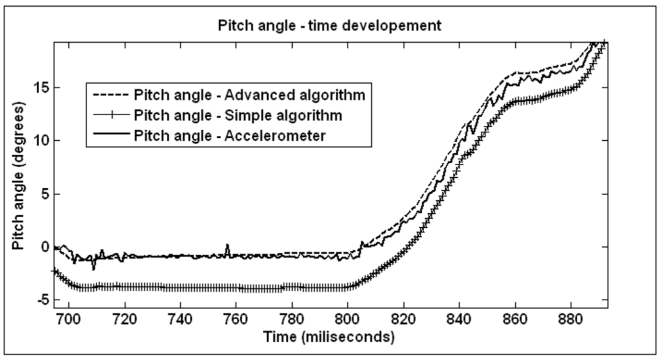

Fig. 4. Pitch angle time development for simple algorithm and for advanced algorithm.

AKNOWLEDGEMENTS

This work has been supported in part by Ministry of Education, Youth and Sports of the Czech Republic (Research Intent MSM0021630529 Intelligent systems in automation (VZ UAMT), Grant Agency of the Czech Republic (102/09/H081 SYNERGY - Mobile Sensoric Systems and Network) and by Brno University of Technology.

REFERENCES

Kang-hua, T., Mei-ping W., Xiao-ping H. (2007). Multiple model kalman filtering for MEMS-IMU/GPS integrated navigation. *ICIEA 2007*, volume(2), 2062-2066.

Kim, J. H., Sukkarieh, S., Wishart, S. (2006). Real-time navigation, guidance, and controlof a UAV using low-cost sensors. *STAR*, volume (24), 299-309.

Lou, L., Xu, X., Cao, J. Chen, Z., Xu, Y. Sensor fusion-based attitude estimation using low-cost MEMS-IMU for mobile robot navigation. IEEE ITAIC, volume (2), 465-468.

Pounds, P. Hamel, T., Mahony, R. (2007). Attitude control of rigid body dynamics from biased IMU measurements. *IEEE CDC*, volume (46), 4620-4625.

Premerlani, W., Bizard, P. Direction cosine matrix IMU: Theory. “unpublished”

Titterton, D.H., Weston, J.L. (2004). *Strapdown inertial navigation technology (second edition)*. Paul Zarchan, Editor-in-chief. Lexington, Massachusetts.