How to obtain angular velocity and angular acceleration values directly from 3D accelerometer array data using simple matrix operations in Propeller/SPIN

...the merit of service is seldom attributed
to the true and exact performer.
William Shakespeare (1564-1616)
All's Well That Ends Well, Act III, scene vi

In the followings we shall reproduce the algorithm described in

K. Parsa, J. Angeles and A. K. Misra
Rigid-body pose and twist estimation using an accelerometer array
In **Applied Mechanics**, 74 (2004) pp. 223-236.

without the agonizing pain of abstract tensor and matrix algebra. The method will be demonstrated with many numeric examples. To calculate these examples I used only Propeller/SPIN and the **FPU_Matrix_Driver.SPIN** object from OBEX (**http://obex.parallax.com/objects/317/**). First the arrangement of four 3-axis acceleration sensors will be described, then the algorithm will be introduced and exercised via numeric examples.

The arrangement of the sensors
Let us put two H48C 3D accelerometers at the opposite corners of a square plate as shown in Fig. 1.

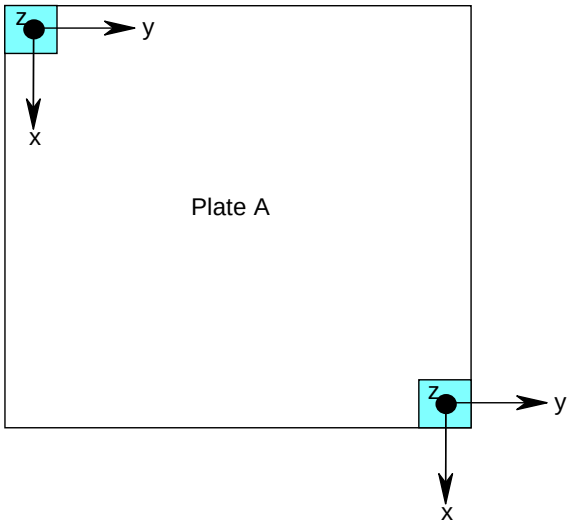

Figure 1. The z-axis of the H48C accelerometers are pointing towards the reader.

Let us make another square plate, equipped with two other sensors, like in Fig. 2.

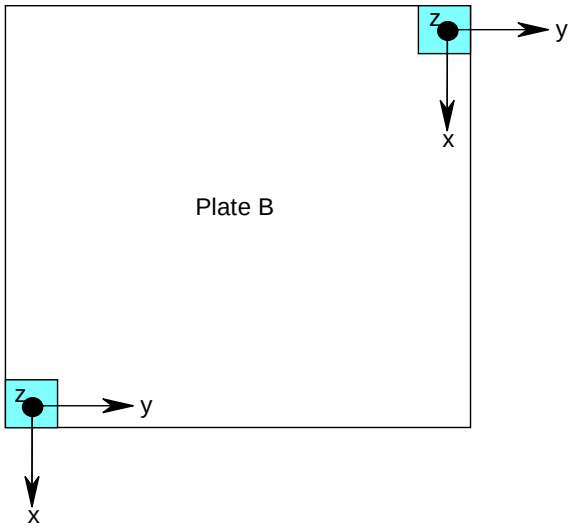

Figure 2. The z-axis of the H48C accelerometers are pointing towards the reader.

Now let us mount Plate A on top of Plate B to form a regular cube as shown in Fig. 3.

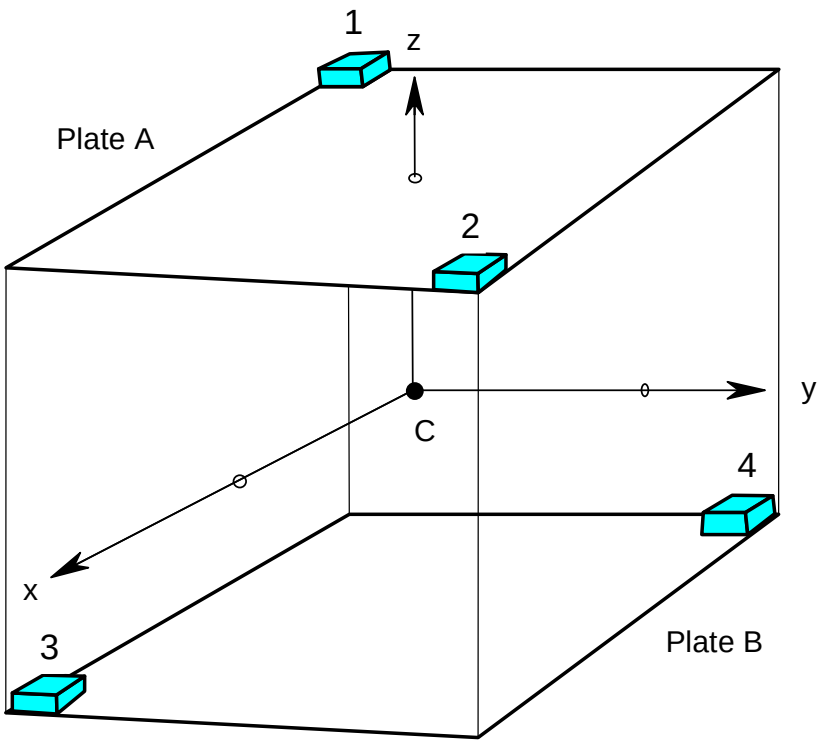

Figure 3. The four H48Cs arranged at the vertices of a tetrahedron.

The three corresponding axis of the sensors are parallel, and, by design, mutually orthogonal. The centroid of the pickup points is denoted by C and the sensors are numbered as shown.

Definition of matrices

Let us define two [3 by 4] matrices. These are matrix **R** of the relative positions and matrix **A** of the relative accelerations. The position of the sensors is related to the centroid C. Let us take the length of the side of the cube as one, then the coordinates of the sensors are

$$r_1 = [-0.5, -0.5, 0.5]$$
$$r_2 = [0.5, 0.5, 0.5]$$
$$r_3 = [0.5, -0.5, -0.5]$$
$$r_4 = [-0.5, 0.5, -0.5]$$

as you can verify this in Fig. 3. The **R** matrix contains the coordinates of the sensors in its columns

$$\mathbf{R} = \begin{matrix} r_1 & r_2 & r_3 & r_4 \end{matrix}$$
$$\begin{bmatrix} -0.5 & 0.5 & 0.5 & -0.5 \\ -0.5 & 0.5 & -0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \end{bmatrix} \quad \text{[3 by 4] matrix}$$

The **A** matrix is the matrix of the relative accelerations. The acceleration vectors measured by the sensors are

$$a_1 = [a_{1x}, a_{1y}, a_{1z}]$$
$$a_2 = [a_{2x}, a_{2y}, a_{2z}]$$
$$a_3 = [a_{3x}, a_{3y}, a_{3z}]$$
$$a_4 = [a_{4x}, a_{4y}, a_{4z}]$$

How to make relative acceleration values from these? Let us first calculate the average acceleration vector a_C

$$a_C = 0.25 \cdot [a_{1x}+a_{2x}+a_{3x}+a_{4x},\ a_{1y}+a_{2y}+a_{3y}+a_{4y},\ a_{1z}+a_{2z}+a_{3z}+a_{4z}]$$

Then subtract a_C from the acceleration vectors to obtain relative accelerations

$$a_{r1} = [a_{1x}-a_{Cx}, a_{1y}-a_{Cy}, a_{1z}-a_{Cz}]$$
$$a_{r2} = [a_{2x}-a_{Cx}, a_{2y}-a_{Cy}, a_{2z}-a_{Cz}]$$
$$a_{r3} = [a_{3x}-a_{Cx}, a_{3y}-a_{Cy}, a_{3z}-a_{Cz}]$$
$$a_{r4} = [a_{4x}-a_{Cx}, a_{4y}-a_{Cy}, a_{4z}-a_{Cz}]$$

And the **A** matrix is

$$\mathbf{A} = \begin{matrix} a_{r1} & a_{r2} & a_{r3} & a_{r4} \end{matrix}$$
$$\begin{bmatrix} a_{r1x} & a_{r2x} & a_{r3x} & a_{r4x} \\ a_{r1y} & a_{r2y} & a_{r3y} & a_{r4y} \\ a_{r1z} & a_{r2z} & a_{r3z} & a_{r4z} \end{bmatrix} \quad \text{[3 by 4] matrix}$$

By the way, a_C is the linear acceleration vector measured by the sensor array. So half of the 6DOF IMU job done. Now, we have to calculate the angular acceleration and the angular velocity values. In other words we will get 9DOF data, won't we? Up till now the operations were reading the sensors, adding, subtracting dividing values, some housekeeping to arrange values in arrays. So, we encountered not too many complications.

An offline task to be solved only once

Before we proceed, we have to calculate the Moore-penrose inverse **P** of matrix **R**. This is easy and has to be done only once for a given sensor arrangement. You can do it with the **FPU_Matrix_Driver** object. Some of the comments of the Matrix_SVD (Singular Value Decomposition) procedure will guide you. Or, you can use some simple matrix algebra as follows

$$\mathbf{P} = \mathbf{R}^T \cdot (\mathbf{R} \cdot \mathbf{R}^T)^{-1}$$

Again, every step can be done with the **FPU_Matrix_Driver**, like for example

```
Matrix_Transpose(@RT,@R,3,4)            'This calculates R^T
Matrix_Multiply(@RRT,@R,@RT,3,4,4,3)    'This calculates R.R^T
Matrix_Inverse(@RRTI,@RRT,3)            'This calculates inverse of (R.R^T)
Matrix_Multiply(@P,@RT,@RRTI,4,3,3,3)   'This calculates P = R^T.(R.R^T)^-1
```

For our sensor arrangement the result is

$$\mathbf{P} = \begin{bmatrix} -0.5 & -0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & -0.5 \\ -0.5 & 0.5 & -0.5 \end{bmatrix} \quad \text{[4 by 3] matrix}$$

Verify that the $\mathbf{R} \cdot \mathbf{P}$ matrix product gives a [3 by 3] identity matrix. O.K. We have **P**, we have to store it somewhere as we shall use it frequently.

A little bit of physics shouldn't hurt

From rigid body kinematics, a very compact formula can be derived for the a_i accelerations. This formula contains the acceleration a_C of the centroid C, the angular velocity ω of the body's rotation around an axis containing the centroid and the time derivative α of the angular velocity, the so called angular acceleration. Note that these are just the IMU quantities we would like to measure. Before I write down the formula, I emphasize again, that our sensor array **estimates directly** all these three basic kinematic vectors. In other words, neither we have to derivate ω to obtain α, or, nor we have to integrate α to obtain ω. Beware of the following formula, because it is so simple that you can even remember it, if you are not careful enough. The formula is

$$a_i = a_C + \alpha \times r_i + \omega \times (\omega \times r_i)$$

where x denotes the vector product. In SPIN using the **FPU_Matrix_Driver**, e.g. for a_1, it goes as

```
Vector_CrossProduct(@wr1,@omega,@r1,3,1)      'This calculates ωxr1
Vector_CrossProduct(@wwr1,@omega,@wr1,3,1)    'This calculates ωx(ωxr1)
Vector_CrossProduct(@alphar1,@alpha,@r1,3,1)  'This calculates αxr1
Matrix_Add(@alphar1wwr1,@alphar1,@wwr1,3,1)   'This calculates αxr1+ωx(ωxr1)
Matrix_Add(@a1,@ac,@alphar1wwr1,3,1)          'This calculates a1
```

Of course, here we use this formula only to calculate correct a_i values for our sensor array for different types of motion of the body to numerically check the decoding algorithm. Now, we have prepared the tests, let's get back to the decoding algorithm.

How to decode the angular acceleration?
Well, we have decoded the linear acceleration of the sensor array. That is simply a_C. To get the angular acceleration, we have first to multiply the **A** matrix of the relative accelerations with **P**. The matrix **A** was calculated from the measured a_i values before and **P** was stored somewhere.

$$\mathbf{W} = \mathbf{A} \cdot \mathbf{P}$$

W has a name, it is called the angular acceleration tensor. But it doesn't matter. We got it. **W** is a small [3 by 3] matrix, nine nicely arranged float values, nothing else from now on. The angular acceleration vector is simply

$$\alpha = 0.5 \cdot [\ W_{32}-W_{23},\ W_{13}-W_{31},\ W_{21}-W_{12}\ ]$$

where the double subscript of W denotes the corresponding element of the W matrix. For example W_{32} is the second element of the third row.

Yes, yes, but what about the angular velocity?
We'll get it quickly. Angular velocity components are calculated from the diagonal elements of the matrix **W**. In preparation of the final result we calculate the quantity

$$sp = 0.5 \cdot (W_{S11} + W_{S22} + W_{S33})$$

and finally

$$\omega = [SQR(W_{S11}-sp),\ SQR(W_{S22}-sp),\ SQR(W_{S33}-sp)]$$

Where SQR denotes the square root operation. These were two additions, a multiplication, three subtractions and three square roots. The correct sign of the components can be obtained easily as described, for example, in the original paper. We shall discuss the sign determination later. We can see, that the nine numbers of the **W** matrix contain all information about angular acceleration and angular velocity. So it deserves its name. Now we continue with some practical considerations and than with the numerical tests.

What to do if I arranged the sensors in a different way?
You have to compute the **R** matrix, then the **P** matrix for your arrangement. That's all. **R**, of course, has not to be singular in order to obtain a Moore-penrose inverse. In a planar arrangement, which seems to be a practical idea to place the sensors, the third row of R contains only zeroes. And **R** is singular, then. In other words, all sensors should not line up, or should not lay in the same plane.

O.K. But how long does this decoding take?
Well, in PASM this decoding takes 4-5 msec. In the FPU it takes less than 2 msec. Whichever you choose, you can handle 100 Hz (10 msec period) acceleration data. In SPIN you can cope with 20 Hz data easily.

First example: Sensor resting on a table
Let us assume that the table is horizontal and is resting on the ground. We have gravity of course (9.81 m/sec^2), but we don't have angular rotation and angular acceleration of the sensor array. The α and ω vectors are zero and the sensed a_i vectors are as follows

$$a_1 = [\ 0.0,\ 0.0,\ 9.81]$$
$$a_2 = [\ 0.0,\ 0.0,\ 9.81]$$
$$a_3 = [\ 0.0,\ 0.0,\ 9.81]$$
$$a_4 = [\ 0.0,\ 0.0,\ 9.81]$$

First we calculate a_C

$$a_C = [\ 0.0,\ 0.0,\ 9.81]$$

Then the **A** matrix is

$$\mathbf{A} = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

The **W** matrix is

$$\mathbf{W} = \mathbf{A} \cdot \mathbf{P} = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

So, both the measured α and ω are null vectors.

Why do our sensors measure a positive (upwards) 9.81 when we all know that gravity points downwards?
The proof-mass, or whatever, that measures the acceleration is pulled down by the gravity. The sensor feels that it is accelerating upwards, because the proof-mass is displaced downwards inside the sensor.

Sensor array accelerating but not rotating
Let us push the sensor array in the x direction with 1 m/sec^2 linear acceleration. The α and ω vectors are zero again , but we have a linear a_C acceleration of the whole sensor in the x direction. According to our formula

$$a_i = a_C + \alpha \times r_i + \omega \times (\omega \times r_i)$$

we calculate a_i values, taking into account the sensed g, of course

$$a_1 = [\ 1.0,\ 0.0,\ 9.81]$$
$$a_2 = [\ 1.0,\ 0.0,\ 9.81]$$
$$a_3 = [\ 1.0,\ 0.0,\ 9.81]$$
$$a_4 = [\ 1.0,\ 0.0,\ 9.81]$$

The measured a_C, the average of the four a_i vectors, is

$$a_C = [\ 1.0,\ 0.0,\ 9.81]$$

Then the **A** matrix is

$$\mathbf{A} = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

The **W** matrix is

$$\mathbf{W} = \mathbf{A} \cdot \mathbf{P} = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

So, both the measured α and ω are null vectors, again.

Let us take some increasing spin

Now, we accelerate the sensor array as in the previous example, but this time we start to rotate it with

$$\alpha = [\ 0.0,\ 0.0,\ 0.5]$$

[rad/sec^2] angular acceleration around the z axis. According to our formula

$$a_i = a_C + \alpha \times r_i + \omega \times (\omega \times r_i)$$

we calculate a_i values again. First let us calculate the $\alpha \times r_i$ vectors

$$\alpha \times r_1 = [\ 0.25, -0.25, 0.00\ ]$$
$$\alpha \times r_2 = [\ -0.25, 0.25, 0.00\ ]$$
$$\alpha \times r_3 = [\ 0.25, 0.25, 0.00\ ]$$
$$\alpha \times r_4 = [\ -0.25, -0.25, 0.00\ ]$$

then the a_i vectors

$$a_1 = [\ 1.25, -0.25, 9.81\ ]$$
$$a_2 = [\ 0.75, 0.25, 9.81\ ]$$
$$a_3 = [\ 1.25, 0.25, 9.81\ ]$$
$$a_4 = [\ 0.75, -0.25, 9.81\ ]$$

The a_C vector, the average of a_i is the same as before

$$a_C = [\ 1.0,\ 0.0,\ 9.81]$$

But the **A** matrix of the relative accelerations is filled not only with zeroes now

$$\mathbf{A} = \begin{bmatrix} 0.25 & -0.25 & 0.25 & -0.25 \\ -0.25 & 0.25 & 0.25 & -0.25 \\ 0.00 & 0.00 & 0.00 & 0.00 \end{bmatrix}$$

The **W** matrix is

$$\mathbf{W} = \mathbf{A} \cdot \mathbf{P} = \begin{bmatrix} 0.0 & -0.5 & 0.0 \\ 0.5 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

From this, using the formula for the decoded, measured α

$$\alpha = 0.5 \cdot [\ W_{32}-W_{23},\ W_{13}-W_{31},\ W_{21}-W_{12}\ ]$$

we obtain

$$\alpha = [\ 0.0,\ 0.0,\ 0.5\ ]$$

and the decoded, measured angular velocity vector is

$$\omega = [\ 0.0,\ 0.0,\ 0.0\ ]$$

Well, so far, so good.

Accelerating and rotating sensor array

Four seconds have passed and the sensor array is rotating now with

$$\omega = [\ 0.0,\ 0.0,\ 2.0\ ]$$

[rad/sec] angular velocity, while accelerating linearly and angularly as before

$$a_C = [\ 1.0,\ 0.0,\ 9.81\ ]$$

$$\alpha = [\ 0.0,\ 0.0,\ 0.5\ ]$$

The constituents to the a_i accelerations at the pickup points are

$$a_C = [\ 1.0,\ 0.0,\ 9.81\ ]$$

$$\begin{aligned}
\alpha \times r_1 &= [\ \ 0.25, -0.25, 0.00\ ] \\
\alpha \times r_2 &= [\ -0.25,\ \ 0.25, 0.00\ ] \\
\alpha \times r_3 &= [\ \ 0.25,\ \ 0.25, 0.00\ ] \\
\alpha \times r_4 &= [\ -0.25, -0.25, 0.00\ ]
\end{aligned}$$

$$\begin{aligned}
\omega \times (\omega \times r_1) &= [\ \ 2.00,\ \ 2.00, 0.00\ ] \\
\omega \times (\omega \times r_2) &= [\ -2.00, -2.00, 0.00\ ] \\
\omega \times (\omega \times r_3) &= [\ -2.00,\ \ 2.00, 0.00\ ] \\
\omega \times (\omega \times r_4) &= [\ \ 2.00, -2.00, 0.00\ ]
\end{aligned}$$

These are sensed accelerations at the pickup points and according to our formula

$$a_i = a_C + \alpha \times r_i + \omega \times (\omega \times r_i)$$

they are added (scrambled) in the sensors

$$\begin{aligned}
a_1 &= [\ \ 3.25,\ \ 1.75, 9.81\ ] \\
a_2 &= [\ -1.25, -1.75, 9.81\ ] \\
a_3 &= [\ -0.75,\ \ 2.25, 9.81\ ] \\
a_4 &= [\ \ 2.75, -2.25, 9.81\ ]
\end{aligned}$$

Now let us see, how the algorithm unscrambles the a_C, α and ω vectors. The a_C is simply the average of the four a_i vectors

$$a_C = [\ 1.0,\ 0.0,\ 9.81\ ]$$

The **A** matrix of the relative accelerations is

$$\mathbf{A} = \begin{bmatrix} 2.25 & -2.25 & -1.75 & 1.75 \\ 1.75 & -1.75 & 2.25 & -2.25 \\ 0.00 & 0.00 & 0.00 & 0.00 \end{bmatrix}$$

We obtain the [3 by 3] **W** matrix with a simple matrix multiplication

$$\mathbf{W} = \mathbf{A}\cdot\mathbf{P} = \begin{bmatrix} -4.0 & -0.5 & 0.0 \\ 0.5 & -4.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

And we unscramble the angular acceleration vector immediately, using the formula

$$\alpha = 0.5\cdot[\, W_{32}-W_{23},\ W_{13}-W_{31},\ W_{21}-W_{12} \,]$$

resulting

$$\alpha = [\ 0.0,\ 0.0,\ 0.5\ ]$$

Then we calculate the quantity

$$sp = 0.5\cdot(W_{S11} + W_{S22} + W_{S33}) = -4.0$$

And finally, from the formula

$$\omega = [SQR(W_{S11}-sp),\ SQR(W_{S22}-sp),\ SQR(W_{S33}-sp)]$$

we get the unscrambled ω vector

$$\omega = [\ 0.0,\ 0.0,\ 2.0\ ]$$

Right, again.

Summarizing the steps of the algorithm

To check and to follow the steps of the numeric examples one can use the Propeller/SPIN language and the **FPU_Matrix_Driver**. Some practice will ensure the user how simple it is and how easy to program the algorithm in Propeller/SPIN. Now I summarize briefly the main steps of the process.

The four sensor readings (a_i vectors) are stored in a [3 by 4] matrix, column wise.

The average of the acceleration vectors gives the linear acceleration ***a_c*** *of the sensor array.*

This average vector is subtracted from each column of the matrix, and the resulting matrix is multiplied with a precomputed one.

The [3 by 3] product matrix is used to estimate the ***α*** *and the* ***ω*** *vectors.*

α *is obtained directly from this matrix with a simple formula.*

ω *is obtained directly again with simple formulas.*

One can get the sign of the components of the ω vector, for example, for ω_x easily as

$$Sign\_of\_\omega_x = SIGN[(\omega_x\_Now - \omega_x\_Previous)\cdot\alpha_x\_Now]$$

based upon the fact that the components of α are measured with sign and they are directly related to the change of the consequtive components of ω. Another, and numerically more robust way to get the sign of the components of the ω vector is to store the sum of the α components. The sign of the stored sums will yield the sign of the measured ω components at any moment. Yes, I know that this is something like an integration. But the difference between using the sign of a value, or using the value itself, is huge.

Some words about calibration

It is beyond the scope of this document to discuss the calibration of the sensor array in details. I will describe the calibration process and the involved simple math of that later. Now I just mention, that each of the H48C sensors should be calibrated statically. After that the whole sensor array can be calibrated kinematically as described in the original paper. This kinematic calibration will improve the precision of the sensor array with several orders of magnitude! The kinematic calibration can be done *en-route* during an arbitrary maneuver of the body that carries the sensor array.

cessnapilot, 14.03.2009