# Time Alignment Techniques for Experimental Sensor Data

Matthew Rhudy

Lafayette College, Easton, PA, 18042, USA

## ABSTRACT

*Experimental data is subject to data loss, which presents a challenge for representing the data with a proper time scale. Additionally, data from separate measurement systems need to be aligned in order to use the data cooperatively. Due to the need for accurate time alignment, various practical techniques are presented along with an illustrative example detailing each step of the time alignment procedure for actual experimental data from an Unmanned Aerial Vehicle (UAV). Some example MATLAB code is also provided.*

## KEYWORDS

*Data Processing, Delay Systems, Time Series Analysis, Timing.*

## 1. INTRODUCTION

It is important for experimental data to be properly aligned in time in order to conduct accurate post-processing analysis. Measurement systems are subject to data loss with time, which causes gaps in the data which need to be properly located and filled in order to maintain the appropriate time scale. It is also important to identify delays between different measurement systems in order to align each data set properly, so that the data from each individual system can be combined and used together for analysis. This is especially important for information fusion applications, which thrive on having as many sources of information as possible [1]. For example, the alignment of measurements from Global Positioning System (GPS) and Inertial Navigation System (INS) [2] is necessary for applications such as human lower limb kinematics [3], positioning [4], and attitude estimation [5]. Distributed systems such as sensor networks also require accurate time synchronization [6,7,8]. Time alignment techniques are also useful for determining delays in a system, which can be an important consideration for certain problems, such as acoustic wave propagation [9] and human pilot modeling [10]. This work addresses a few techniques which are useful for time alignment, and aims to serve as an instructive guide for using these techniques. Note that this work is intended only to provide some information about a few time alignment techniques. Other more complex time alignment methods can be found e.g. in [11,12,13].

The rest of this paper is organized as follows. The time alignment techniques are presented in Section 2. In Section 3 the experimental platform is discussed, followed by a time alignment example in Section 4. Section 5 provides a brief conclusion. Some example code using MATLAB is provided in the Appendix.

## 2. TIME ALIGNMENT TECHNIQUES

### 2.1. Alignment of Two Signals using Correlation Functions

The pure time delay between two measurement signals can be captured using correlation functions. These functions include the auto-correlation function $R_{xx}(\tau)$ and the cross-correlation function $R_{xy}(\tau)$, of two signals $x(t)$ and $y(t)$, given by

$$R_{xx}(\tau) = \frac{1}{T}\int_0^T x(t)x(t+\tau)dt \tag{1}$$

$$R_{xy}(\tau) = \frac{1}{T}\int_0^T x(t)y(t+\tau)dt \tag{2}$$

Note that the auto-correlation function is just the cross-correlation function of a given signal with itself. Note also that $R_{xy}(\tau) = R_{yx}(-\tau)$, and the auto-correlation function is always an even function. The auto-correlation function represents time properties in the data that are separated by fixed time delays. For example, consider the auto-correlation function for a signal $x(t) = \sin(t)$ of length $T$, where $n$ is an integer. For this example, the auto-correlation function is given by

$$R_{xx}(\tau) = \frac{1}{T}\int_0^T \sin(t)\sin(t+\tau)dt \tag{3}$$

Using the trigonometric identity

$$\sin(u)\sin(v) = \frac{1}{2}\left[\cos(u-v) - \cos(u+v)\right] \tag{4}$$

and performing the integration, the auto-correlation function becomes

$$R_{xx}(\tau) = \frac{1}{2}\cos(\tau) + \frac{1}{4T}\left[\sin(\tau) - \sin(\tau+2T)\right] \tag{5}$$

If the length of the signal $T$ is a multiple of the period of the signal, *i.e.*, $T = 2\pi n$, where $n$ is an integer, then the equation reduces to

$$R_{xx}(\tau) = \frac{1}{2}\cos(\tau) \tag{6}$$

Therefore, the auto-correlation function of a sine wave has peaks at any time shift of $2\pi n$, where $n$ is an integer. This corresponds to the signal aligning with itself due to its natural periodicity. An illustration of this result is given in Figure 1. The auto-correlation functions for non-periodic signals such as random noise or impulse noise are much different than that of periodic signals. Consider, for example, data sampled from a standard normal distribution, *i.e.*, Gaussian random noise with zero mean and standard deviation one. The auto-correlation for this signal should only align with itself when there is no time shift ($\tau = 0$). When $\tau$ is nonzero, the randomness of the signal forces the value of $R_{xx}(\tau)$ to approximately zero, as shown in Figure 2.

Another interesting type of signal to consider is an impulsive signal. The auto-correlation for a signal of this type will have some peaks around $\tau = 0$ due to the alignment of the absolute maximum with the subsequent local maxima and minima. To help illustrate this, Figure 3 shows an example military impulse wave [14]. The cross-correlation function between two signals represents their similarity based on a time shift, $\tau$. This can be useful for various applications, including detection of time delays, noise source identification, and radar and sonar applications [15]. In general, unlike the auto-correlation function, the cross-correlation function will not be an even function. In fact, one useful feature of this function is measuring the value of $\tau$ where the function is at its absolute maximum. This value represents a time delay between the two signals. To help illustrate this concept, using the previous impulse signal from Figure 3, another signal was created by delaying the original signal by 0.1 seconds. Figure 4 shows the original and delayed signals together, as well as their cross-correlation function. The absolute maximum of the cross-correlation function occurs at a value $\tau$ = -0.1 *s*, which agrees with the time delay between the signals. Here the minus sign indicates that there is a lag in time as opposed to a positive sign which would represent a lead in time. Notice that the cross-correlation function for this example is the same as the auto-correlation function in Figure 3 except for the shift in time.

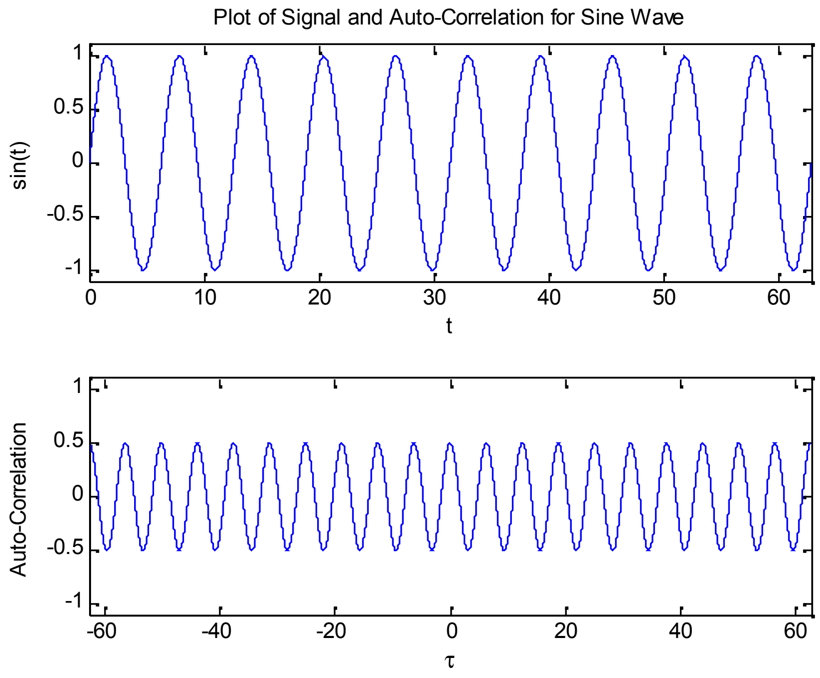


Figure 1. Plot of Signal and Auto-Correlation for Sine Wave

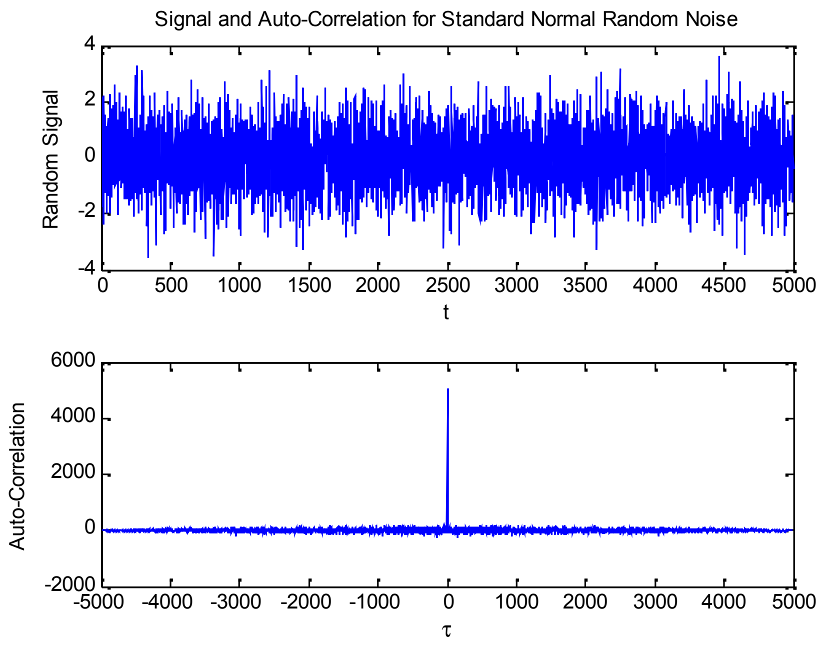


Figure 2. Plot of Signal and Auto-Correlation for Random Noise

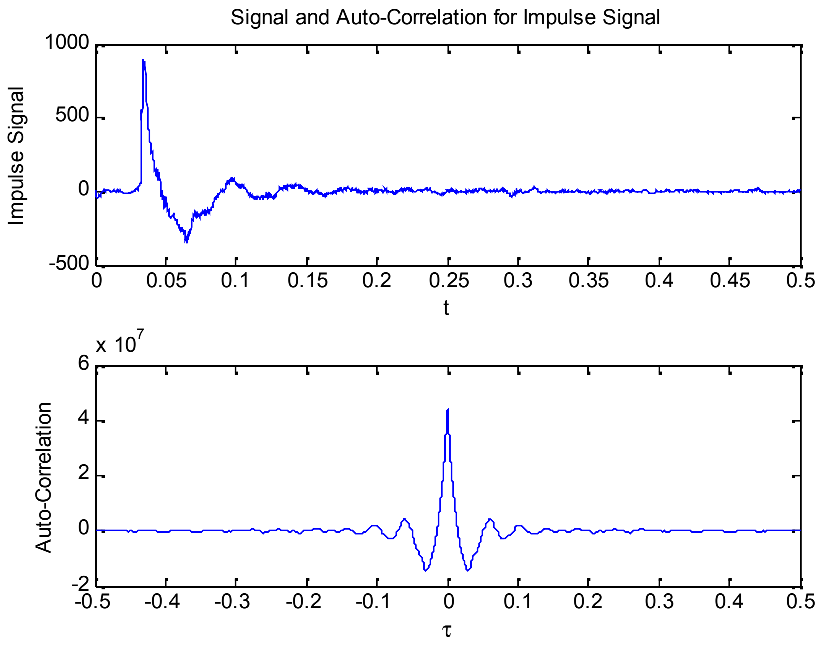


Figure 3. Example Military Impulse Wave Signal with Auto-Correlation

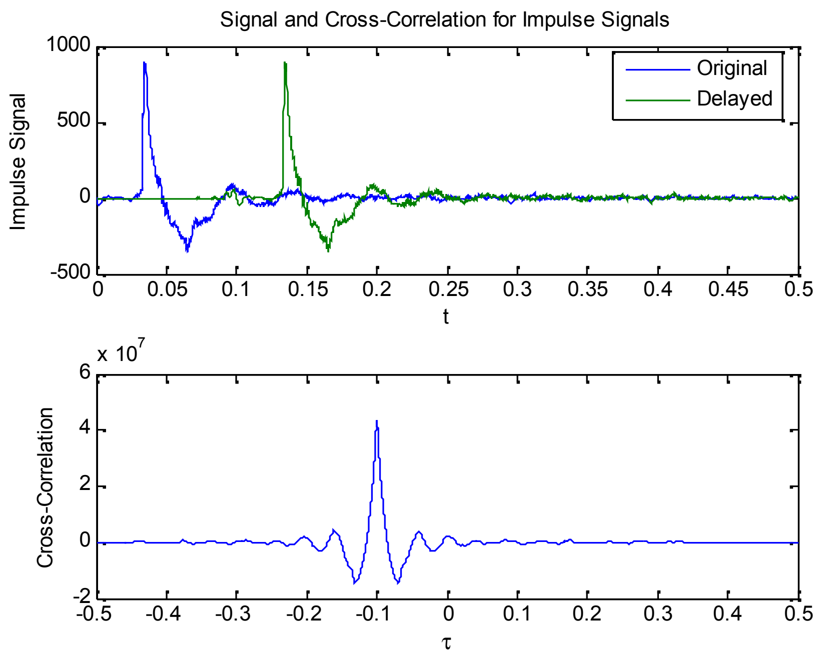


Figure 4. Plot of Original and Delayed Signals with and Cross-Correlation

The calculation of the auto-correlation and cross-correlation functions can be visualized by a shifting in time of one signal with respect to one another, while multiplying their corresponding values. This process is similar to that of a convolution integral of two functions $x(t)$ and $h(t)$, given by [16]

$$y(\tau) = \int_0^\tau x(t)h(\tau - t)dt \tag{7}$$

The primary difference between a convolution integral and a cross-correlation is the negative sign in the convolution integral, which in effect is a reflection of one of the signals about $t = 0$ before time shifting.

When dealing with discrete or digital data, the auto-correlation and cross-correlation functions cannot be calculated exactly. In this case, the auto-correlation and cross-correlation functions can be approximated by [17]

$$\hat{R}_{xx}(r\Delta t) = \frac{1}{N-r}\sum_{n=1}^{N-r} x_n x_{n+r} \tag{8}$$

$$\hat{R}_{xy}(r\Delta t) = \frac{1}{N-r}\sum_{n=1}^{N-r} x_n y_{n+r} \tag{9}$$

where $r = 0, 1, 2, \ldots, m$, with $m < N$. Here, $r$ represents the lag number, and $\Delta t$ is the time step, which is the inverse of the sampling frequency in Hz. The ^ symbol above the $R$ denotes an approximation. Note that these calculations can also be done using the *xcorr* command in MATLAB. Another quantity of interest is the cross-correlation coefficient function $\rho_{xy}(\tau)$, given by

$$\rho_{xy}(\tau) = \frac{R_{xy}(\tau)}{\sqrt{R_{xx}(0)R_{yy}(0)}} \tag{1}$$

which can be approximated by

$$\hat{\rho}_{xy}(r\Delta t) = \frac{\hat{R}_{xy}(r\Delta t)}{\sqrt{\hat{R}_{xx}(0)\hat{R}_{yy}(0)}} \tag{2}$$

This quantity always satisfies $-1 \leq \rho_{xy}(\tau) \leq 1$ [17]. Since the cross-correlation coefficient is a normalized quantity, it is ideal for use in comparing different waveforms. This value is a measurement of the agreement between the two signals as a function of the time delay. For time alignment, this value can be used to gauge how well the signals are aligned, with 1 indicating perfect positive correlation, -1 indicating perfect negative correlation (same signal, but reflected about the *x*-axis), and 0 indicating no correlation.

## 2.2. Aligning a Signal with the Proper Time Scale

A problem that can occur with real measurement systems is the loss of information with time. I.e., the data recording system might miss a packet (or more) of information, either due to delays in the system or processing constraints. Typically in this situation, the measurement system simply skips the missed packet and logs the next data point instead. For example, if a data stream is supposed to be (1, 2, 3, 4, 5, …), and the 3rd packet is skipped, the data will be logged as (1, 2, 4, 5, …). As this occurs over time, the data loss accumulates and therefore compresses the data along the time scale. To restore the true time vector, an accurate time reference is needed. Because of this, it is useful to obtain some information about the timing of the data in each packet of information, such as from a counter on a microprocessor or an independent timing source, e.g. in GPS data, the GPS time is encoded in each data packet. Using this independent timing source, the locations of skipped data packets can be identified. These locations can be found easily by taking the difference in successive time steps of the counter.

Consider the following example of a counter which skips packet 3 and packet 10. Figure 5 shows the true ideal counter (blue) and the recorded counter from the measurement system (red). Taking the difference between successive time steps gives Figure 6, which shows that 2 steps occurred in the counter instead of 1 at time steps 2 and 8. Note that it is time step 2, because data packet three was skipped, therefore at step 2 it skipped 3 and went to 4. For time step 8, it is off by an additional time step because of the previous data loss at time step 2. These points can be used to identify the jumps in the data. Then, the data stream can be “stretched” out in time by the number of data losses, and then these gaps can be filled in using interpolation.

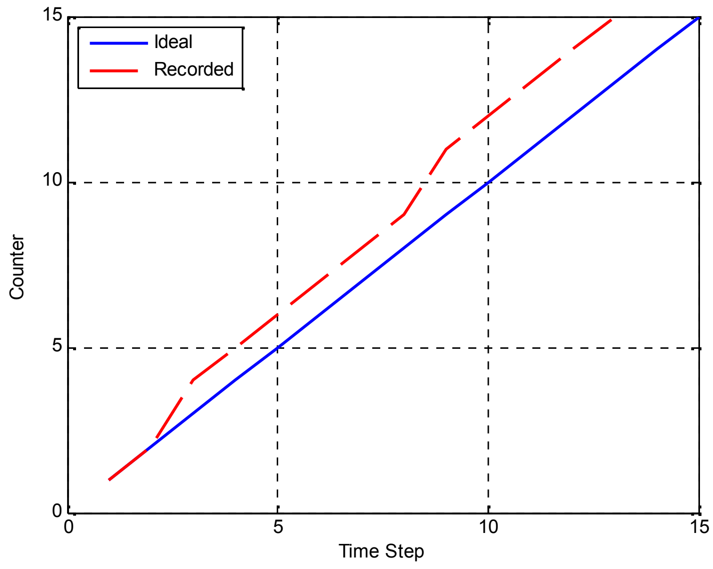


Figure 5. Data Packet Loss Example: Ideal and Recorded Counter Signals

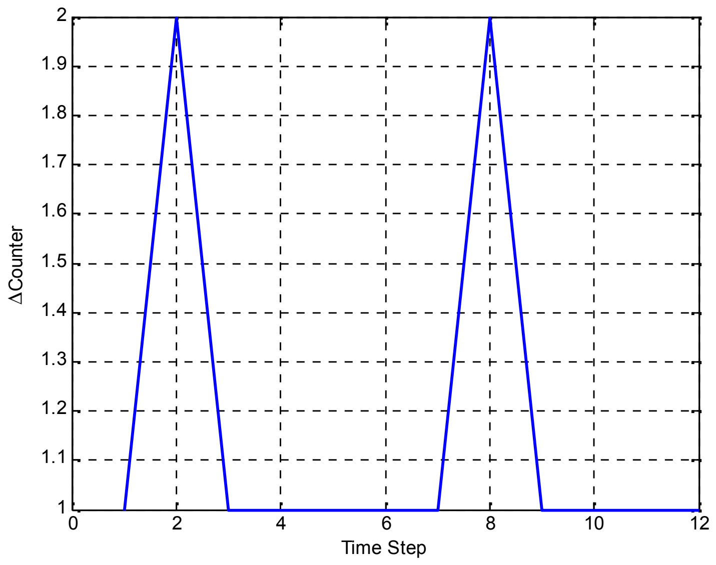


Figure 6. Data Packet Loss Example: Difference between Time Steps

### 2.3. Precision Time Alignment using Pulse Per Second (PPS) Signals

When aligning signals from two different data streams, a precision time alignment method can be used to align these measurements using a Pulse Per Second (PPS) signal. First a coarse time alignment method should be applied to align the signals within one second of accuracy. Then, a fine alignment can be performed using a PPS signal, which is an electrical signal that precisely indicates the start of a second. This signal takes the shape of a square wave, with period of 1 second. A PPS signal should be generated within the first data system and sent to the other data system for logging. Then this gives two sources of the same information, which can then be aligned in order to obtain precisely aligned measurements. Any phase shift between the two signals is eliminated by shifting one of the signals until a perfect time alignment is achieved. For this time alignment, the cross-correlation technique can be used.

## 3. Experimental Platform

The flight data used for this study were collected with the Red Phastball aircraft which was designed, manufactured, and instrumented by researchers at the Flight Control Systems Laboratory (FCSL) at West Virginia University (WVU). The avionic payload includes a custom designed printed circuit board (PCB) featuring four Analog Devices® Inertial Measurement Units (IMUs) and a Novatel OEM-V1 GPS receiver, as shown in Figure7.

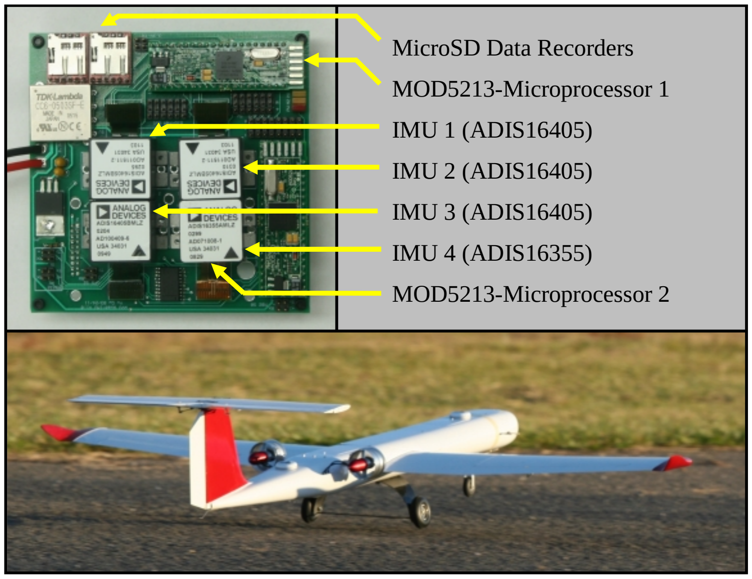


Figure 7. WVU Phastball Aircraft and Avionics

The PCB shown in Figure 7 records measurements using two MicroSD data loggers which are interfaced with the various measurement systems using two MOD-5213 microprocessors. The GPS data is recorded on a third MicroSD data logger. Of the four IMUs implemented in this system, three are ADIS16405, and one is an ADIS16355. Although each IMU has an actual resolution of 14-bit, the resolution is improved by oversampling the signals at 200 Hz, then averaging down to 50 Hz, thus achieving 16-bit resolution. A Pulse Per Second (PPS) signal from the GPS receiver is recorded with the IMU data using an Analog to Digital (A/D) port on the MOD-5213 microprocessor. This PPS signal is utilized for precision time alignment between the IMU and GPS data. The GPS receiver uses satellite information to calculate Cartesian position and velocity. In addition to IMU and GPS data, a high-quality Goodrich® mechanical vertical gyroscope was used to obtain direct measurements of the roll and pitch of the aircraft, which are used as 'truth' measurements for this study. These measurements are recorded using a 3.3 V A/D at 16-bit resolution. All data were recorded at 50 Hz.

## 4. Time Alignment Results using Experimental Sensor Data

This section goes through an example time alignment procedure using a data set collected with the measurement system described in Section 3. The first step in the time alignment procedure is to align each data stream individually with the proper time scale. First, the GPS data is considered. The GPS time is logged with each GPS packet, and is shown in Figure 8 (left). The difference in successive time steps is shown in Figure 8 (right), which shows were data loss occurred, and how much time was lost. A zoomed in segment of this data is shown in Figure 9 Figurein order to provide an example of a typical data loss. Using the method described in Section 2.2, the GPS data can be reconstructed with the proper time scale. Note that the data "stretching" is applied to all of the channels in the GPS data, not just the GPS time.

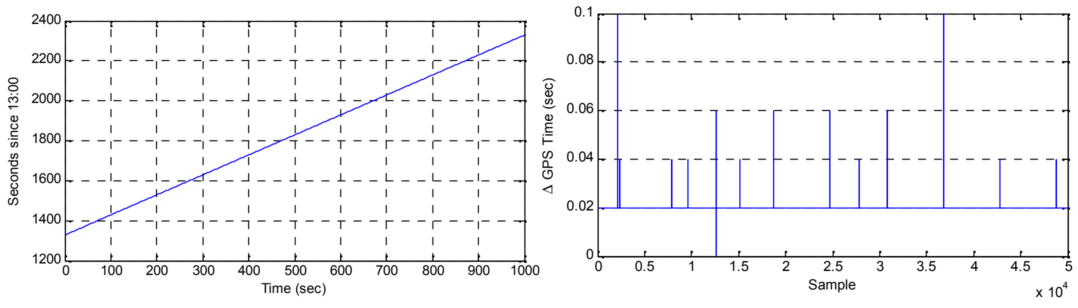


Figure 8. GPS Time (left) and Change in Time (right)

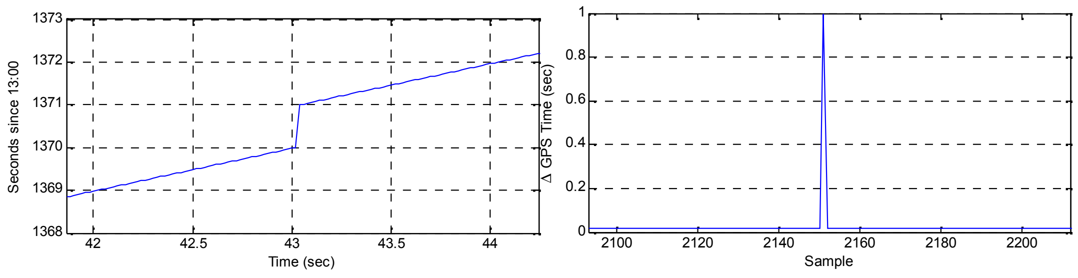


Figure 9. Example of GPS Data Loss

Similarly as for the GPS, the microprocessor data is subject to packet loss with time. The microprocessor logs a counter at each time step, which is used to identify locations of data loss. The counter is shown in Figure 10 (left) while the change in counter is in Figure 10 (right). A zoomed in segment of this data is shown in Figure 11 in order to provide an example of a typical data loss. Using the method described in Section II B, the microprocessor data can be reconstructed with the proper time scale. Note that the data "stretching" is applied to all of the channels in the microprocessor data, not just the counter. See Example Code 1 in the Appendix for more information about how to perform time alignment using a counter.

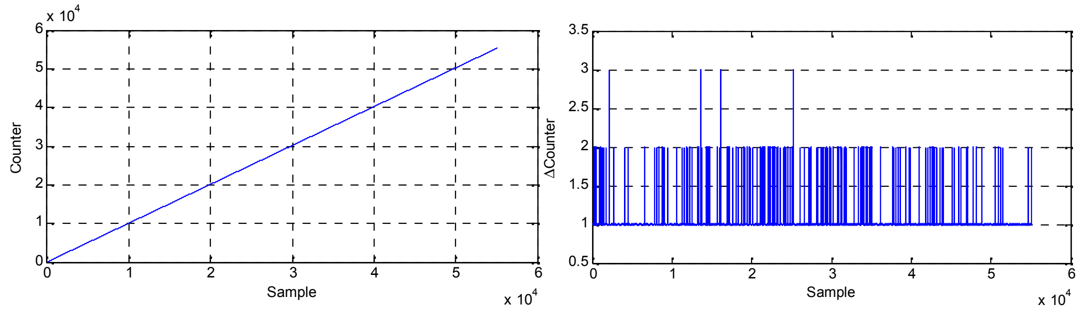


Figure 10. Microprocessor Counter (left) and Change in Counter (right)

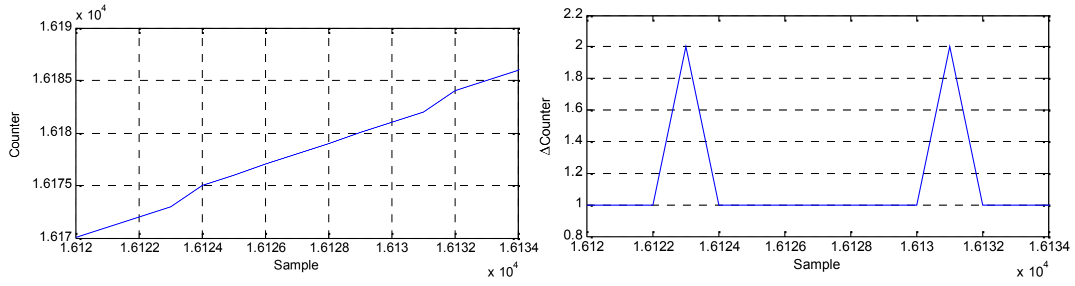


Figure 11. Example of Microprocessor Data Loss

Since this data set contains measurements from two different microprocessors, each needs to be individually aligned in time. Then, these two measurement systems must be aligned with one another. For this application, each measurement system contains an IMU, which measures equivalent values on each system. Therefore, the time alignment between these two measurement systems can be done using cross-correlations between common channels between the two systems. An example of this time alignment is shown in Figure 12, where the cross-correlation function is shown, while the aligned signals are shown in Figure 13 after shifting by the maximum of the cross-correlation. See Example Code 2 in the Appendix for more information about how to align different measurement systems using cross-correlations.

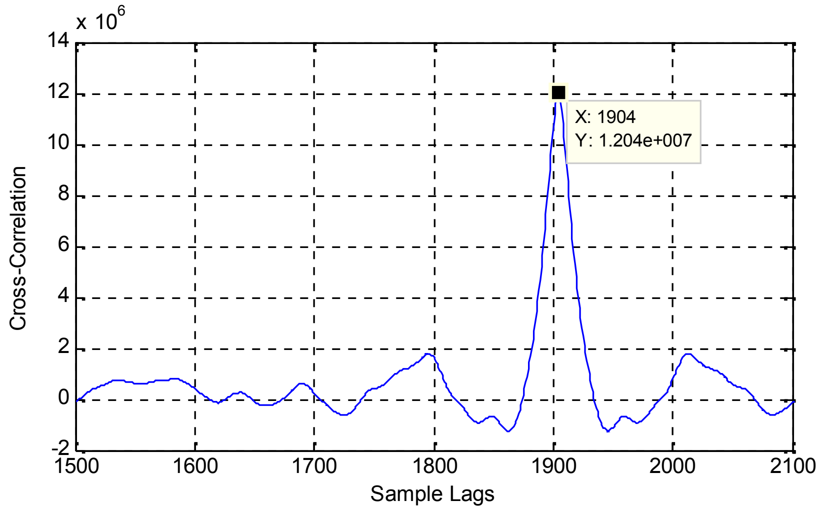


Figure 12. Cross-Correlation between Redundant Measurements

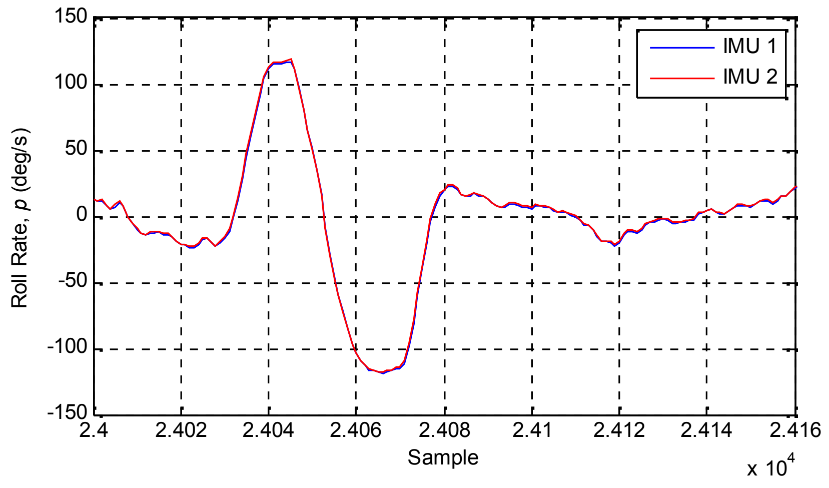


Figure 13. Aligned Redundant Measurement Signals

Now, each individual measurement system has been aligned with a proper time scale, and the two microprocessors are aligned. Next, the GPS needs to be aligned with the microprocessors. In order to apply the fine time alignment technique of Section 2.3, a PPS signal was programmed on a GPS receiver to trigger a digital high at the beginning of a second and a digital low at half of a second. This digital signal was recorded by an Analog to Digital (A/D) converter on a Netburner microprocessor and was stored with the other data channels. For the fine time alignment, an artificial PPS signal is constructed for the GPS data based on the time recorded in each packet. This artificial PPS signal is then compared with the measured PPS signal in the microprocessor data stream. Figure 14 shows the unmatched PPS signals (left), and the PPS signals after final alignment (right).

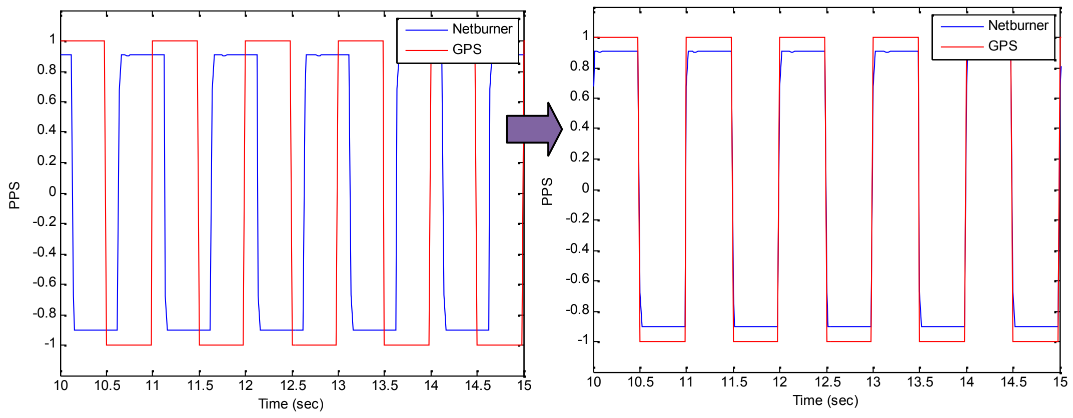


Figure 14. PPS Precision Time Alignment

Another problem that is important to consider is the possibility of spikes in the data. This is particularly important for GPS data, which is prone to data spikes. To remove data spikes, an appropriate threshold can be selected in order to limit the maximum change in the data between time steps. Then large data spikes can be identified and eliminated using a zero-order hold, i.e. when a spike occurs, just use the previous value again. An example of data spikes and their removal is shown in Figure 15.

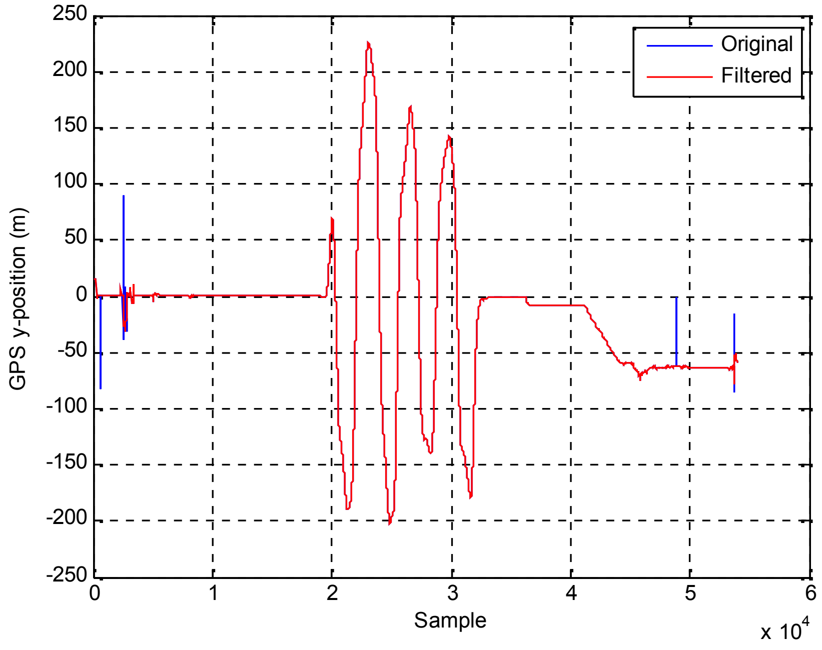


Figure 15. Example Spike Removal

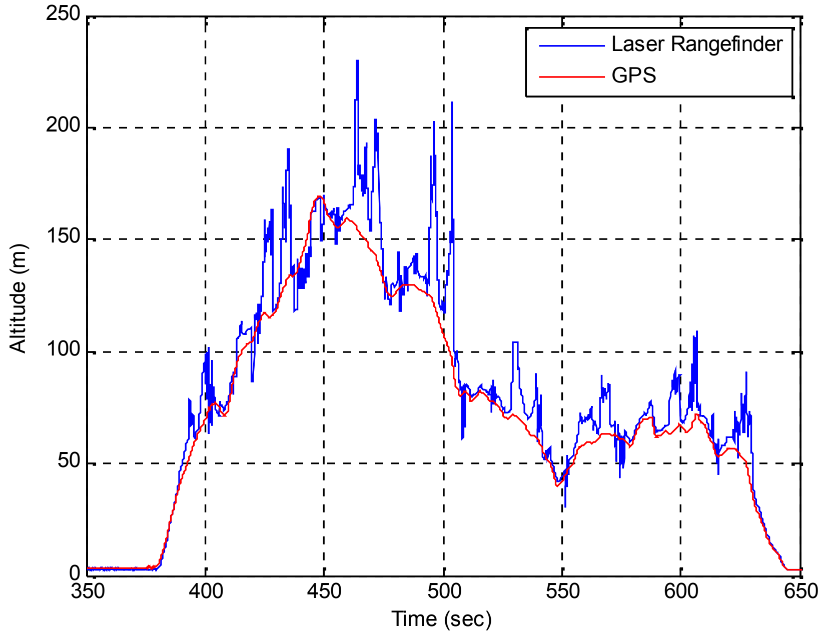


Figure 16. Example Time Alignment Check

See Example Code 3 in the Appendix for more details about how to remove spikes. Once the data has been cleaned of all spikes, and the signals have been aligned, it is useful to verify the time alignment somehow. For this particular example, the GPS has measurements of the altitude of the aircraft, while the microprocessor has measurements from a laser rangefinder which is pointing directly downward from the aircraft. While these measurements are not of exactly the same quantity, they are correlated in time, and therefore can be used to validate the accuracy of the time alignment, as shown in Figure 16.

## 5. CONCLUSIONS

This article developed the use of different techniques for time alignment of experimental sensor data. The presented approaches were based on practical experience obtained while working with flight data from an experimental UAV. The techniques presented in this article were developed in an instructive manner in order to provide a practical resource for working with multiple measurement systems.

## ACKNOWLEDGEMENTS

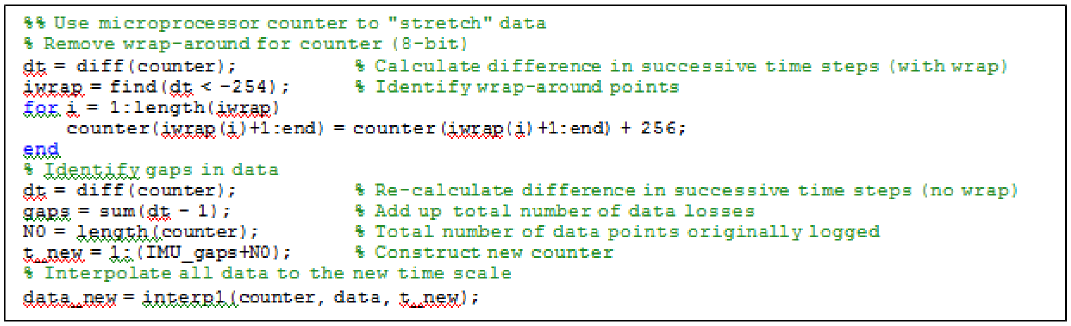
The author would like to thank Francis J. Barchesky, Dr. Haiyang Chao, Dr. Jason N. Gross, Dr. Yu Gu, Dr. Srikanth Gururajan, and Dr. Jeffrey Vipperman for their help in deriving this work.

## REFERENCES

[1] Rogova, G. L., and Nimier, V., "Reliability in Information Fusion: Literature Survey," Proc. 7th Int. Conf. Inf. Fusion, Stockholm, Sweden, 2004, pp. 1158-1165.
[2] Grewal, M. S., Weill, L. R and Andrew, A.P., Global Positioning, Inertial Navigation & Integration, 2nd. Ed., John Wiley & Sons, New York, NY, 2007.
[3] Kwakkel, S. P., "Human Lower Limb Kinematics Using GPS/INS," M.S. Thesis, University of Calgary, 2009.
[4] Li, Y., Mumford, P., Wang, J., Rizos, C., and Ding, W., "Time Synchronisation Analysis of an FPGA based GPS/INS Integrated System," International Global Navigation Satellite Systems Society IGNSS Symposium, Surfers Paradise, Australia, July 2006.
[5] Gross, J., Gu, Y., Rhudy, M., Gururajan, S., and Napolitano, M., "Flight test evaluation of GPS/INS sensor fusion algorithms for attitude estimation," IEEE Trans. on Aerospace Electronic Systems, Vol. 48, No. 3, July 2012, pp. 2128-2139.
[6] Sivrikaya, F., and Yener, Bülent, "Time Synchronization in Sensor Networks: A Survey," IEEE Network, Vol. 18, 2004, pp. 45-50.
[7] Elson, J. E., "Time Synchronization in Wireless Sensor Networks," Ph.D. Dissertation, University of California, 2003.
[8] Sichitiu, M. H., and Veerarittiphan, C., "Simple, Accurate Time Synchronization for Wireless Sensor Networks," Proc. IEEE Wireless Communications and Networking Conference, 2003, pp. 1266–1273.
[9] Rhudy, M., Bucci, B., Vipperman, J., Allanach, J., and Abraham, B., "Microphone Array Analysis Methods Using Cross-Correlations." IMECE2009-10798, Proceedings of IMECE 09: 2009 ASME International Mechanical Engineering Congress, November 13-19, 2009, Lake Buena Vista, Florida.
[10] Mandal, T., Gu, Y., Chao, H., and Rhudy, M, "Flight Data Analysis of Pilot-Induced-Oscillations of a Remotely Piloted Aircraft," AIAA Guidance Navigation and Control Conference, Boston, MA, 2013.
[11] Haffner, P., Franzini, M., and Waibel, A., "Integrating Time Alignment and Neural Networks for High Performance Continuous Speech Recognition," Proc. IEEE International Conference of Acoustics, Speech, and Signal Processing, Toronto, Ontario, Canada, April 1991, pp. 105-109.
[12] Shimodaira, H., Noma, K-I., Nakai, M., and Sagayama, S., "Dynamic time-alignment kernel is support vector machine," Advances in Neural Information Processing Systems 14, Vol. 2, Cambridge, MA: MIT Press, 2002, pp. 921-928.
[13] Cao, H., Tehrani, A. S., Nemati, H. M., Fager, C., Eriksson, T., Zirath, H., "Time Alignment in a Dynamic Load Modulation Transmitter Architecture," Proc. of the 39th European Microwave Conference, Rome, Italy, 2009, pp. 1211-1214.
[14] Rhudy, M., "Real Time Implementation of a Military Impulse Noise Classifier", M.S. Thesis, University of Pittsburgh, 2009.
[15] Norton, M., and Carczub, D., Fundamentals of Noise and Vibration Analysis for Engineers, Cambridge University Press, New York, NY, 2nd Ed., 2003.

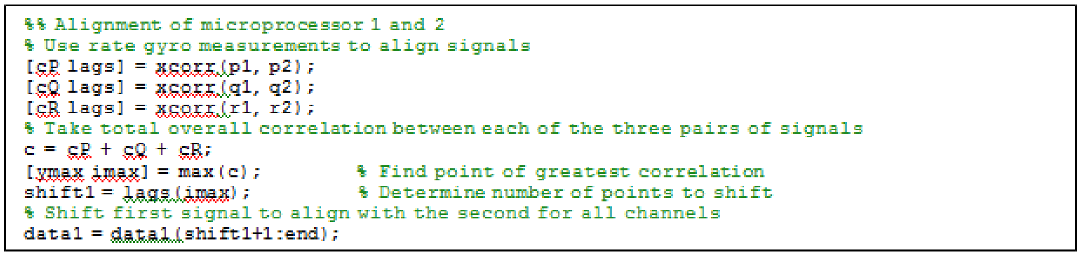
[16] Rao, S. S., Mechanical Vibrations, Pearson Education, Inc., Upper Saddle River, NJ, 4th Ed., 2004.
[17] Bendat, J., and Piersol, A., Random Data: Analysis and Measurement Procedures, John Wiley and Sons, Inc., New York, NY, 2nd Ed., 1986.

## APPENDIX



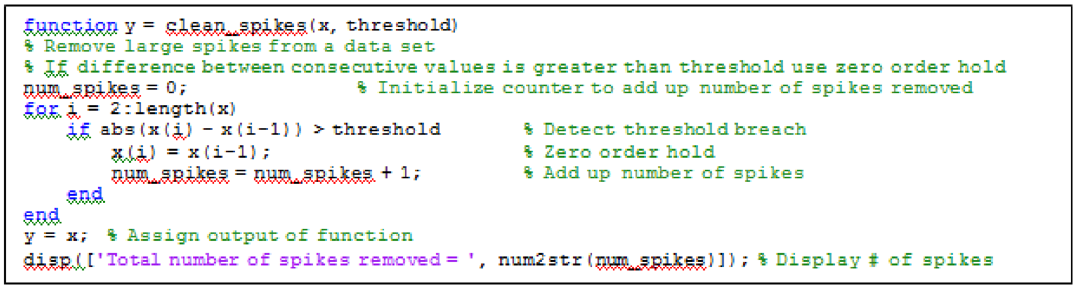
```
%% Use microprocessor counter to "stretch" data
% Remove wrap-around for counter (8-bit)
dt = diff(counter);            % Calculate difference in successive time steps (with wrap)
iwrap = find(dt < -254);       % Identify wrap-around points
for i = 1:length(iwrap)
    counter(iwrap(i)+1:end) = counter(iwrap(i)+1:end) + 256;
end
% Identify gaps in data
dt = diff(counter);            % Re-calculate difference in successive time steps (no wrap)
gaps = sum(dt - 1);            % Add up total number of data losses
N0 = length(counter);          % Total number of data points originally logged
t_new = 1:(IMU_gaps+N0);       % Construct new counter
% Interpolate all data to the new time scale
data_new = interp1(counter, data, t_new);
```

Example Code 1: Time Alignment with Counter



```
%% Alignment of microprocessor 1 and 2
% Use rate gyro measurements to align signals
[cP lags] = xcorr(p1, p2);
[cQ lags] = xcorr(q1, q2);
[cR lags] = xcorr(r1, r2);
% Take total overall correlation between each of the three pairs of signals
c = cP + cQ + cR;
[ymax imax] = max(c);          % Find point of greatest correlation
shift1 = lags(imax);           % Determine number of points to shift
% Shift first signal to align with the second for all channels
data1 = data1(shift1+1:end);
```

Example Code 2: Alignment of Signals using Correlations



```
function y = clean_spikes(x, threshold)
% Remove large spikes from a data set
% If difference between consecutive values is greater than threshold use zero order hold
num_spikes = 0;              % Initialize counter to add up number of spikes removed
for i = 2:length(x)
    if abs(x(i) - x(i-1)) > threshold        % Detect threshold breach
        x(i) = x(i-1);                       % Zero order hold
        num_spikes = num_spikes + 1;         % Add up number of spikes
    end
end
y = x;  % Assign output of function
disp(['Total number of spikes removed = ', num2str(num_spikes)]); % Display # of spikes
```

Example Code 3: Spike Removal Function

## Authors

Matthew B. Rhudy received a Bachelor's of Science in mechanical engineering in 2008 from The Pennsylvania State University, State College, PA, a Master's of Science in mechanical engineering in 2009 from the University of Pittsburgh, Pittsburgh, PA, and a Ph.D in aerospace engineering in 2013 from West Virginia University, Morgantown, WV.

Since 2013, he has been a Visiting Assistant Professor in the Department of Mechanical Engineering at Lafayette College in Easton, PA.