

# Evaluation of Inertial Measurement Unit Performance for Water Column Velocity Estimation of Coastal Profiling Floats Nicholas Raymond, University of California Davis

Mentor: Gene Massion Summer 2016

## Keywords: Coastal Profiling Float, Velocity Estimation, Inertial Measurement

## ABSTRACT

The Monterey Bay Aquarium Research Institute is developing a coastal profiling float to enable long-term continuous measurement of near-shore marine environments using biogeochemical sensors. To prevent these new floats from drifting out of their designated measurement volume, it is desirable to monitor their underwater drifting velocity using inexpensive inertial measurement units. This project developed a systematic approach for evaluating these types of sensors using both simulations and experimental data. The approach outlined in this paper was successfully used to process raw accelerometer data from a Bosch BNO055 sensor and obtain modestly accurate velocity estimates from accelerations that were smaller than the resolution of the sensor. In the future, this methodology can be applied to test more sensitive sensors as well as different digital filters to determine the optimal combination that would result in accurate and reliable velocity estimation for the coastal profiling float project.

# INTRODUCTION

Scientists and engineers want to use biogeochemical sensors to collect data over larger temporal and spatial scales as a way of monitoring the health of the ocean in an effort to better understand the ocean's role in the global carbon cycle [Johnson K. et al. (2009)] While ongoing efforts like the Argo profiling float project monitors the open ocean, a comparable program does not yet exist for coastal areas. However, there has been an increased interest in studying coastal zones as these regions represent a significant wealth of ecological diversity and financial benefit that is highly susceptible to human influence.

Profiling float technology has been found to be a cost effective and robust platform for continuous ocean observations. These autonomous devices can change their buoyancy to float at various depths within the water column and passively drift in underwater currents while taking sensor measurements. Due to the success of these types of floats, the Monterey Bay Aquarium Research Institute (MBARI) has been developing a new type of profiling float that is optimized for coastal deployment. These coastal profiling floats (CPF) will be outfitted with biogeochemical sensors and operate at a maximum depth of 500 meters. The CPFs are deigned to last for five years without maintenance or human intervention, and unlike the conventional profiling floats, the CPF must stay within a very narrow region along the coastline. The most recent MBARI CPF design is shown in figure 1, with the CAD model provided by ocean engineer Gene Massion.



Figure 1: MBARI coastal profiling float design (left) and cross section view (right).

Once deployed, the CPF is programmed to monitor a specific coastal measurement volume and should remain within that measurement volume over the duration of the deployment. The CPF should not drift too close, nor should it drift too far from the coastline. In this way, a network of floats could be deployed in a grid pattern to gather high-resolution data sets over a large region of ocean. In an effort to keep the CPF design simple and cost effective, floats are not equipped with active propulsion systems. This inhibits the ability for CPFs to stay within the designated ocean measurement volume while being exposed to strong currents and seasonal weather patterns. One proposed solution for keeping the CPF within the measurement volume is to monitor the CPF's drifting speed and direction as it ascends to the surface. When the float encounters a strong subsurface current, it's velocity would quickly change and information about the strength and direction of the current could be estimated. An onboard pressure sensor provides continuous depth measurements and would be used to determine the depth where currents were strongest.

After reaching the surface and uploading all of its data via iridium satellite link, the CPF would dive down and then passively drift back towards the location where it began its mission. It is critical that the underwater velocity of the CPF be determined so that identification of subsurface currents moving in opposite direction to the dominant CPF drift direction would be possible. A typical mission cycle is shown in figure 2.



Figure 2: Example of CPF mission cycle using subsurface currents to drift back to the starting position.

Traditionally, the Global Positioning System (GPS) has been used to provide accurate velocity and position-monitoring for devices that operate at the surface of the ocean. However, GPS signal is not strong enough to travel through the water and tracking ends when the device is submerged. In very special applications, inertial measurement units (IMU) have been used to provide "dead reckoning" navigation for submarines, planes, and AUVs [Titterton, D. H., & Weston, J. L. (2004)]. While it is not possible to directly measure velocity or position with an IMU, a vehicle's acceleration can be measured and transformed into a global reference frame if the orientation of the sensor is known. The transformed acceleration measurements are then integrated to obtain an estimate for velocity and integrated a second time to obtain estimates of position. Traditional IMU systems implement accelerometers and gyroscopes with very fine resolution that require careful calibration. This is done in an effort to reduce the accumulation of errors that are inherent with this process, but this contributes to the high cost of these units.

Advancements in the semiconductor industry have led to the rapid growth of small, inexpensive IMU sensors. These microelectromechanical systems (MEMS) based IMUs are attractive solutions because they are small, light weight, consume less power, and have been used to provide orientation and velocity estimates for small aircraft and robots over relatively short time intervals where error accumulation can be mitigated [Titterton, D. H., & Weston, J. L. (2004)]. The disadvantage of this technology is that the sensor outputs are much noisier and typically require signal processing or filtering which makes velocity and position estimation difficult over long time spans due to significant error accumulation. Wanting to explore the potential benefits of MEMS sensor technology, the scope of this internship project was to develop a systematic methodology to evaluate the performance of a generic MEMS IMU for the purpose of CPF velocity estimation so that future researchers could compare and evaluate the performance of a range of different sensors of varying cost and sensitivity. The following sections provide details on the methodology and experiments conducted, as well as the results from running the velocity estimation algorithm on both simulated and experimental data.

## **MATERIALS AND METHODS**

The research project was split into three separate tasks. First, a dynamic simulation was created to estimate the range of expected acceleration and velocity magnitudes that a CPF would experience during a typical profiling mission. Next, the

values obtained from the simulation were used to conduct a series of experiments using a linear translation stage to emulate the motion of the CPF while drifting underwater in the ocean currents. During these experiments, the sensor was rigidly mounted to the stage and acceleration data was collected as the table moved back and forth with a specified acceleration and terminal velocity. Next, this experimental data was processed through a velocity estimation algorithm designed to filter the raw sensor output and then perform numerical integration. Finally, the experimental and simulated velocity profiles were compared using two different digital filtering methods and a performance metric was used to quantify the effectiveness of each approach.

#### DYNAMIC SIMULATION

A MATLAB Simulink model was created to evaluate the forces acting on the CPF in the North/South and East/West directions and estimate the float's acceleration and velocity as a function of time. The hydrodynamic drag acting on the float was computed with the following equation, operating under the assumption that the CPF's shape is approximated as a short cylinder:

$$F_{drag} = \frac{1}{2} C_{drag} \rho \left( U - V \right)^2 Area$$

In the above equation the variable  $C_{drag}$  represents the drag coefficient,  $\rho$  is the density of sea-water, U is the speed of the current, V is the speed of the CPF, and Area is the cross sectional area of the cylinder which is normal to the direction of the moving fluid. By applying Newton's 2<sup>nd</sup> law and summing all forces acting on the CPF at a given instant of time, it was possible to solve for the three mutually perpendicular instantaneous accelerations aligned in the North-East-Down (NED) global coordinate system. The flow chart in figure 3 shows each step of the dynamic model calculation.



Figure 3: Flow chart showing the force balance and integration steps used in dynamic simulation.

A closed-loop hydraulic dive engine developed at MBARI modulates the CPF's buoyancy. This directly controls the descent and ascent velocity of the CPF by pumping hydraulic fluid between a large internal oil bladder and two smaller flexible oil bladders mounted on the CPF's exterior. Test tank experiments conducted by engineer Gene Massion have shown that this system provides reliable velocity control and that the CPF floats to the surface in a near vertical orientation. As a result, the dynamic model assumed a constant vertical velocity of 10 cm/sec and a perfect vertical orientation (no pitch or yaw). The Simulink model for the North/South direction is shown in figure 4, while the architecture for the East/West and Vertical axes are virtually identical.



Figure 4: MATLAB Simulink CPF dynamic model for the North/South direction.

During the simulation, it is assumed that the +X axis of the sensor remained aligned with the North direction while the +Y axis of the sensor remained aligned with the East direction. In practice, the true NED orientation of the CPF will need to be monitored; however, this was outside the scope of this work and will be included in future work. The Simulink model for the North/South direction is shown in figure 4, while the architecture for the East/West and Vertical axes are virtually identical.

Once the outputs from the model were verified using step and ramp inputs, a series of ocean current profiles were input into the simulation. These current profiles represented historic data obtained from the M1 mooring in Monterey Bay, and were gathered from the MBARI Live Access to Data website <u>http://dods.mbari.org/</u>. Figure 5 gives a visual representation of two days worth of current measurements and range from 10 - 300 meters of depth. The warm colors represent currents moving in the East direction and the cooler colors represent the currents moving in the West direction. This data set was chosen because it represented the type of velocity profiles that are of most interest to the CPF project; strong subsurface currents moving in the opposite direction to currents at the ocean surface.



Figure 5: Current velocity measurements from the M1 mooring in Monterey Bay.

After running the dynamic model on the five data sets from February 2002, it was estimated that a CPF would experience accelerations with a magnitude of 0.02 - 1 mm/sec<sup>2</sup> and a range of velocities from 10 - 160 mm/sec. An example of the simulation outputs are shown in figure 6 and figure 7 comparing both the input current profiles and the resulting CPF velocity and acceleration profiles.



Figure 6: CPF simulated velocity profiles based on historic data from Monterey Bay currents.



Figure 7: CPF simulated acceleration profiles based on historic data from Monterey Bay currents.

### **BNO055** Inertial Measurement Unit

Prior to the start of the summer internship, a Bosch BNO055 IMU breakout board as seen in figure 8 was purchased from Adafruit Industries at a cost of \$35. The MEMS accelerometer within the IMU had a resolution  $\pm 9.8$  mm/sec<sup>2</sup> with a maximum sampling rate of 100 Hz [Bosch Sensortec (2016)]. Based on the dynamic simulation results, the

sensor resolution was at least 10x larger than the acceleration signal of interest. In ideal circumstances, the resolution of the sensor should be smaller than that of the smallest signal to be measured. While it is possible to purchase more sensitive accelerometers from different vendors, those sensors are typically 10 to 100 times more expensive than the BNO055.



Figure 8: Image of the BNO055 IMU breakout board purchased from Adafruit Industries. Image source: https://www.adafruit.com.

The emphasis of this project was to develop a systematic approach for evaluating the performance of these MEMS sensors and it was decided that work would continue with the BNO055. Additionally, researchers at MBARI were interested in determining the minimum acceleration value that this IMU could reliably detect once the raw output had undergone signal processing. Signal processing is used to reduce the magnitude of the noise within a measurement and isolate the desired waveform so that useful measurements may be obtained even when the resolution of the sensor is larger than the magnitude of the signal of interest [Steven S. (1997)].

A data acquisition unit (DAQ) was built using a GHI G400 microcontroller to poll the sensor and save the output to an SD card. The DAQ firmware was written in C# using the Visual Basic integrated development environment and called a timer interrupt routine every 10 milliseconds to maintain a sensor sampling rate of 100 Hz. At the end of the programmed sampling period, the sensor measurements were written to the SD card and the data from each accelerometer axis was saved as its own text file in byte format. An onboard real-time clock was used to keep accurate time and date measurements, and was also used during the firmware development to verify the timer interrupt sampling rate. The fully assembled DAQ is shown in figure 9.



Figure 9: Fully assembled data acquisition unit for reading IMU sensor and saving data to memory card.

## PATH TRAJECTORY SIMULATOR

Before running physical experiments with the sensor, a path trajectory simulator was developed to generate a range of velocity and acceleration profiles that could be programmed into a linear translation stage controller. These ideal path trajectories accomplished two functions: first they were used to determine the motion of the liner translation stage and ensure that the physical limitation of the table were not exceeded, and secondly these idealized path trajectories would be used to provide a comparison with the experimental data.

To use the path trajectory simulator, the user must first specify the initial table location, constant acceleration/deceleration, and terminal velocity parameters. These values determined the shape of the ideal velocity profile and were calculated from the equations of motion for a particle having constant acceleration. The equations are shown in figure 10.

[1]  $x = x_0 + v_0 t + \frac{1}{2}at^2$ [2]  $v = v_0 + at$ [3] a = constant[4]  $v^2 = v_0^2 + 2a(x - x_0)$ 

Figure 10: Equations of motion used to calculate the desired simulation velocity profile.

Using the initial conditions and predefined motion parameters, the velocity profile was generated as a piecewise function having four unique sections. By solving for the change in time required to achieve the terminal velocity for each segment, it was possible to calculate four unique values of time and position. These position and time values were then used to generate the ideal trapezoid profile. An example of the trapezoid profiles are shown below, which was programmed to have a terminal velocity of 16 mm/sec. The first trapezoid shape indicates a positive velocity in the +Y direction, the inverted trapezoid profile represents the table moving back in -Y direction such that the table ends at the initial starting location. This represents a complete back-and-forth cycle.



Figure 11: Simulated velocity profile for a single cycle with the linear table moving back-and-forth.

Once the complete back-and-forth profile had been created, the total number of cycles was computed for a specified simulation time (ex. 10 minutes). The final simulation velocity for a 10 minute cycle is shown in figure 12:



Figure 12: Simulated velocity profile for entire simulation run time of 10 minutes, with six cycles.

Next, a simulated acceleration profile was created by taking the derivative of the velocity profile. The kinematic equations of motion assume constant accelerations, so a simple alternating square wave could be used to represent the simulated acceleration 11

profile. As with the velocity profile, once a complete back-and-forth acceleration cycle had been created, a longer acceleration profile to span the full simulation run time could be created. This was important because the error in the velocity estimate was highly dependent on the total sampling time. The linear table was only 600 mm long, placing physical limitations on the size and shape of the path trajectories. A single pass with the linear table did not produce enough data points to see a significant difference in the velocity estimation methods which is why multiple back-and-forth cycles were needed.



Figure 13: Simulated acceleration profile for one cycle of the linear translation stage.



Figure 14: Simulated acceleration profile for entire simulation run time of 10 minutes, with six cycles.

#### LABORATORY EXPERIMENTS

A bench top simulator was used to emulate the motion of the CPF drifting in subsurface currents and allowed for controlled laboratory testing. The motion control device was a single Newport IMS600CCHA high performance linear translation stage that had position resolution of 0.2 microns and a PID closed loop motion controller. The motion of the linear stage was determined by a series of ASCII text commands programmed into a desktop computer that was then sent to the motion controller which

power the DC motor servo motor. The experimental setup is shown below and included the DAQ, linear translation stage, BNO055 sensor, and desktop computer.



Figure 15: Experimental setup showing the BNO055 IMU sensor mounted to linear translation stage.

The following combination of velocities and accelerations were run with the linear translation table. During all experiments, the Y axis of the sensor was aligned with the direction of motion, and prior to running the experiments the linear table was leveled using a machinists digital level that had a resolution of 0.1 degrees.

Run	Acceleration	Velocity
	$(mm/sec^2)$	(mm/sec)
Α	150	200
В	40	100
C	10	50
D	5	35
E	1	16

Table 1: Acceleration and velocity parameters used to program the liner translation table.

# VELOCITY ESTIMATION ALGORITHM

For the last part of this project, a velocity estimation algorithm was created in MATLAB to filter the raw sensor data and perform numerical integration. The first step was to look at the power spectrum density (PSD) of the sensor signal and identify the 13

dominant frequencies within that signal. The Welch method was used to obtain an estimate on the PSD and implemented a Hamming window with 50% overlap and FFTs that had individual widths of 8,192 bins. This information was then compared to the PSD for the simulated acceleration signal in order to differentiate the true signal from noise. Before calculating the PSD for each axis, any linear trends and offsets were removed by applying the MATLAB *detrend()* function to the raw sensor data. The resulting PSD for the Y axis measurements are shown in figure 16. The portion of the signal from 0.01- 2.5 Hz was identified as the actual acceleration signal and the remaining frequencies corresponded to noise from the MEMS sensor or the mechanical linear stage vibrations.



Figure 16: Estimation of power spectral density from Y axis sensor data.

The velocity estimation program implemented two types of filters:

A. Total variation denoising filter – referenced from image processing literature

B. Moving average filter – basic digital infinite impulse response filter A brief overview of each filter is given below, however it is important to note that these two filters are simply examples of signal processing methods that might be suitable for this project. More testing is still needed to determine the best filter and filter settings for estimating CPF velocities given a wide range of acceleration measurements.

The total variation denoise filter, hence forth referenced as *denoise*, is heavily used in digital image processing since it has the benefit of removing background noise while maintaining sharp edges and contrast. The denoise function implemented in this project was written by Mr. Daniel R. Pipa who had referenced the work published by Chan S. et al. (2011). This version of the denoise filter had two parameters,  $\mu$  and  $\rho$ , which had to be adjusted based on the noise amplitude and characteristics of the signal being filtered. Through trial and error, it was found the denoise filter was effective at removing the noise from all five of the experimental runs which ranged from 1 - 150mm/sec<sup>2</sup>. Below is a parameter study, showing the affect of changing the  $\mu$  parameter from 0.01 to 0.005, with the white plot representing the simulated acceleration profile. The benefit of this filter is clearly seen in the image below, with sharp edge detection for the majority of the profiles.



Figure 17: Denoise filter performance for range of  $\mu$  values.

An infinite impulse response (IIR) filter was also implemented to provide a comparison to the denoise filter. The IIR filter is a type of digital filter that a recursive process, using information about the previous output value to update the filter, which results in robust digital filters that are computationally efficient [Steven S. (1997)]. The MATLAB *filter()* function is used to create custom digital filters by passing A and B matrices that represent the numerator and denominator of the filter transfer function, respectfully. To implement a simple moving average filter, the A matrix is given a default value of one, and the B matrix is determined by the number of data points that will be average together. Once values have been determined for the A and B matrices, the digital filter is created and applied to the raw accelerometer data. While simple in its implementation compared to other IIR filters, moving average filters have been found to

be very effective at reducing additive Gaussian white noise and are computationally efficient making them suitable as a first attempt when trying to process sensor outputs [Steven S. (1997)]. Just as with the denoise filter, a series of simulations were run to observe the affect of altering the number of points averaged together. The image below shows five simulations where the number of averaged data points ranged from 1,000 - 5,000.



Figure 18: Test to determine the effect of changing the number of points used in moving average filter.

With both filter settings selected, the raw accelerometer data was then processed. The plot below shows the filtered profiles along with the ideal simulated acceleration profile. There is an obvious phase delay between the experimental and simulated data, which is corrected within the program by specifying the time delay for each filtered profile. The phase delay is primarily caused by misalignment between the start of the simulated and experimental data and is not a serious concern. A small portion of the phase delay is caused by the filtering methods, with the moving average filter producing the larger phase delay. However, a phase delay of 1 - 5 seconds is not significant for this project, since the CPF is profiling upwards at a velocity of 10 cm/sec and the variation between subsurface currents at various depths is on the order of meters not centimeters.



Figure 19: Comparison of the filtered accelerometer data with the simulated acceleration profile.

Numerical integration is performed after signal processing. A cumulative trapezoid rule was implemented by calling the MATLAB function *cumtrapz()*. After numerical integration there was a dominant periodic waveform superimposed with the velocity estimates which was due to low frequency noise spectra that was not filtered out from either the denoise or moving average filters. To obtain an accurate estimate of the velocity profile, this low frequency periodic signal had to be removed. This was accomplished by fitting a 12<sup>th</sup> order polynomial to each velocity profile and then subtracting the polynomial from the data set. The results are shown in figure 20.



Figure 20: Final velocity estimates after subtracting 12<sup>th</sup> order polynomials from velocity profiles.

Alternatively, a band pass filter may be implemented before the numerical integration step. If tuned properly, the band pass filter would avoid the need to use this polynomial fit method. While the polynomial fit approach works with this data set, the experimental data is periodic and only approximates the types of signals that would be expected from the ocean. Is not desirable to use this polynomial fit approach on data collected from ocean measurements since fitting high polynomials to stochastic data will not be computationally efficient nor is it expected to be effective.

The last step in the velocity estimation program is to compare the experimental data to the simulated acceleration and velocity profiles and compute error estimates by calculating mean and standard deviation values for the segments of the profiles where there should be constant velocity. A plot of the final velocity estimates and the ideal simulated velocity profile are shown in figure 21.



Figure 21: Comparison of final velocity estimates with simulated profile.

## **RESULTS/DISCUSSION**

A total of five experimental runs were conducted and results from the two most extreme cases are shown below. Run 1 had the largest velocity and acceleration settings, velocity = 200 mm/sec and acceleration =  $150 \text{ mm/sec}^2$ . This set of data was used to verify that the velocity estimation algorithm was functioning properly. Surprisingly, the

estimation program performed very well when the denoise filter was tuned to  $\mu = 0.002$ and  $\rho = 10$ , while the moving average used 1000 average points.

Next, the filters were tuned to process the data from Run 5 which had velocity = 16 mm/sec and acceleration = 1 mm/sec<sup>2</sup>. These values were more similar to conditions that the CPF is expected to experience while floating in subsurface currents. The denoise filter parameters were set to  $\mu = 0.0035$  and  $\rho = 50$ , while the moving average filter used 4,600 averaging points. The average velocity values for each filter are plotted along with error bars representing one standard deviation.



Figure 22: Performance of velocity estimation algorithm for Run 5 data, red line represents the true constant velocity value of 16 mm/sec.

The algorithm perform modestly well, with both filtering methods producing results that first underestimate, then overestimate, and finally underestimate the true value. While nether filtering method achieved consistent estimates of 16 mm/sec, the program is considered a success because the velocity estimates are within an order of magnitude of the true programmed velocity and this is considering that fact that the noise of the acceleration signal was  $\pm 200 \text{ mm/sec}^2$  for a true signal of only  $\pm 1 \text{ mm/sec}^2$ .

Next, the filter settings (which were optimized for Run 5 with small accelerations) were applied to the Run 1 data that had the larger acceleration and velocity values. This was done as a way to see if the filter settings could be used for a wide range of

acceleration measurements, or if the filtering methods were only effective for a limited range of sensor values. The results are shown in figure 23, with the velocity = 200 mm/sec.



Figure 23: Performance of velocity estimation algorithm for Run 1 data, red line represents the true constant velocity value of 200 mm/sec.

In this situation, the moving average filter severely underestimated the true velocity value. The denoise filter performed modestly well, also underestimating the true velocity with an average value that ranged between 140 - 170 mm/sec. Near the end of the 10 minutes, the average velocity estimates from the denoise filter become smaller and the standard deviation grows larger as the error in the estimate continues and the variability of the estimate increases. This is all visualized in figure 24, which shows the first four minutes of data from the velocity estimates for Run 1 using the filter settings established for the Run 5 data set.



Figure 24: Velocity estimates for V = 200 mm/sec, using filter settings optimized for small accelerations.

# CONCLUSIONS/RECOMMENDATIONS

The goal of the internship was not to determine the best filtering method or the optimal filter settings. Rather, a systematic procedure was created for evaluating MEMS based IMU sensors using both simulation and experimental data. This organized approach will allow other researchers to experiment with new filtering methods and quantify the improvements that are made. Furthermore, the five experimental data sets collected with the BNO055 and linear translation stage can be used as a benchmark, allowing direct comparison of the new filtering methods without the need to conduct additional experiments in the short term. Once a more effective filtering method is determined, experiments using new sensors having smaller acceleration resolution can conducted and the results should be compared to the BNO055 results.

While the filtering methods discussed in this report did produce adequate results, they have not been fully optimized nor do they work over a wide range of acceleration measurements. It is believed that more effective filter methods can be implemented that will result in more accurate velocity estimates. For example, the implementation of a band pass filter would fix two important issues: the DC offsets could be removed without the use of the MATLAB *detrend()* function, and a carefully tuned band pass filter would remove the need for the polynomial fit routine.

The programs developed for this project have been transferred to engineer Gene Massion and saved on the MBARI CPF work directory so that other researchers will be able to use this software to continue this work. Based on the results presented in this paper, it seems highly feasible that given a more sensitive sensor and optimized filtering methods it will be possible to estimate the underwater velocity of the CPF for limited periods of time using the MEMS IMU sensors.

Future work should continue to explore alternative digital filtering methods, and begin to incorporate the magnetometer sensor readings into the project so that NED orientation values can be collected along with the acceleration data. This will provide the researchers with information about drifting speed and the direction. Lastly, this analysis assumed that the CPF was perfectly vertical as it drifted upward to the surface of the water. Since this may not be the case during ocean deployment with turbulent currents, the actual orientation of the CPF must be used to correct the accelerometer measurements before filtering and integration. One significant benefit of the BNO055 is that an onboard microcontroller processes accelerometer and gyroscope measurements to produces orientation estimates in real time. This information can also be read by the DAQ with minimal modification to the firmware so that the velocity estimation program can use the orientation values. All of these efforts will aid in producing more reliable and accurate velocity estimates for the CPF so that the larger scientific research object can be realized.

#### ACKNOWLEDGEMENTS

First and foremost I would like to thank George Matsumoto and Linda Kuhnz for being such amazing intern coordinators and for helping me navigate the inner workings of MBARI and the internship. I would also like to thank the Chemical Sensor lab staff: Josh Plant and Hans Jannasch for letting me tag along during the inspection of Lobo mooring and for their helpful afternoon talks, Carole Sakamoto for patiently explaining how the nitrate sensor calibration process works, and Luke Coletti for sharing his past experiences with electrical engineering and discussing the development of the ISUS hardware. I would also like to give special thanks to Ken Johnson for allowing me to borrow a large portion of Chemical Sensor laboratory space during the summer internship and for being a positive supporter of the IMU sensor project. And most importantly, I am eternally grateful to my amazing mentor Gene Massion for always making time to share his insights and wisdom with me and for teaching me the importance of developing a systematic approach for solving hard engineering problems. Oh, and yes he does enjoy telling lots of jokes.

#### **References:**

Access to quality-controlled MBARI OASIS Data http://dods.mbari.org/lasOASIS/main.pl

Bosch Sensortec (2016). BNO055 Intelligent 9-axis absolute orientation sensor. Document number: BST-BNO055-DS000-14.

Chan S., Khoshabeh R., Gibson K., Gill P., Nguyen T., (2011). An Augmented Lagrangian Method for Total Variation Video Restoration. IEEE Transactions on Image Processing IEEE Trans. on Image Process.

Daniel P.R., (2012). Stack Exchange Forum - Signal Processing Question http://dsp.stackexchange.com/questions/3026/picking-the-correct-filter-foraccelerometer-data

Johnson, K., Berelson, W., Boss, E., Chase, Z., Claustre, H., Emerson, S., Gruber N., Körtzinger A., Perry M., Riser, S. (2009). Observing biogeochemical cycles at global scales with profiling floats and gliders: prospects for a global Array. Oceanography, 22(3), 216–225. http://doi.org/10.5670/oceanog.2009.81

Steven S. (1997). *The scientist and engineer's guide to digital signal processing* California Technical Publishing, ISBN-10: 0966017633

Titterton, D. H., Weston, J. L. (2004). *Strapdown inertial navigation technology*. Reston, VA: American Institute of Aeronautics and Astronautics.