Annals of the University of North Carolina Wilmington Master of Science in Computer Science and Information Systems

http://www.csb.uncw.edu/mscsis/

VISUALIZATION OF THE SPURS EXPERIMENT USING GOOGLE EARTH

Frederick M. Bingham

A Capstone Project Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Department of Information Systems and Operations Management

University of North Carolina Wilmington

2012

Approved by

Advisory Committee

Dr. Douglas Kline

Dr. Ronald Vetter

Dr. Bryan Reinicke, Chair

Accepted By

Dean, Graduate School

Abstract

Visualization of the SPURS Experiment using Google Earth. Bingham, Frederick, 2012. Capstone Paper, University of North Carolina Wilmington.

SPURS (Salinity Processes in the Upper Ocean Regional Study) is a multi-investigator oceanographic experiment to be carried out in the North Atlantic Ocean in the vicinity of (25°N, 38°W) in the year between August 2012 and September 2013. The purpose is to study the salinity maximum region of the North Atlantic and to understand the processes that maintain it. The experiment requires intercalibration of many salinity sensors on many different platforms to obtain an experiment-wide accuracy of about 0.1. Before the experimental assets go into the water there is a need to simulate the intercalibration process and make sure that there will be enough encounters between instruments to properly compare them. Accompanying this simulation is a need for visualizing the progress of the experiment. To this end, a visualization of the simulated experiment was created from historical float and drifter data, projected ship tracks, mooring locations and glider paths. These data were visualized using Google Earth. Added to the visualization were locations and times where instruments came close enough in space and time (~10 km, 1 hour) to be compared to each other. A total of 65 drifters, 25 floats, 2 ships, 3 gliders and 3 moorings were included in the visualization. These instruments came into proximity with each other about 6,000 times during the visualized experiment providing a rich dataset for understanding the intercalibration process.

Table of Contents

Chapter 1: Introduction

SPURS (Salinity Processes in the Upper Ocean Regional Study; (1)) is an oceanographic field experiment scheduled to take place in the subtropical North Atlantic in 2012-2013. The area of study is (Fig. 1; 15-30°N, 30-45°W), with an intensive study area near the central flux mooring at (25°N, 38°W). Its purpose is to understand the processes that create and maintain the subtropical horizontal salinity maximum and the surface salinity balance in general (2). SPURS will consist of a large number of elements: in situ data specifically collected for the experiment, in situ data being collected anyway but useful for the experiment, model output and satellite data. In situ data may be comprised of research vessel ancillary instruments, semi-autonomous gliders, floats, moored buoys, drifting buoys, ship-based profiles, towed profilers, microstructure measurements, volunteer observing ship data, etc.

In order to plan for this experiment, a simulation will have to be created, showing all of the elements of the experiment working together. In addition, this simulation will have to be visualized in a way that makes it clear what assets will be where and when. The need for the simulation is driven by the issue of sensor intercalibration.

Salinity is in general a relatively difficult measurement to make in seawater. In modern terms it is quantified by measuring conductivity and temperature and converting that to salinity in grams of salt per kilogram of seawater through the use of the 1978 practical salinity scale (3). As currently defined, salinity is unitless, but the pseudo-unit, the practical salinity unit is commonly seen in the literature. Important variations in ocean salinity occur on the 0.001 level, though SPURS has set an experiment-wide goal of accuracy of 0.01. Standard ocean salinity in the SPURS study region is 37. Thus there is a requirement for salinity measured during SPURS to be accurate to 4 decimal places, or

1 part in 10,000. Salinity sensors are notorious for calibration drift because the calibration is highly dependent on the geometry of the cell (4). They can be severely impacted by biofouling or physical stress, especially for sensors that spend their time continuously in the surface ocean.

In normal measurements of salinity on board a ship, instrument salinities are compared to seawater samples whose values are using an instrument called a salinometer (4). The salinometer is calibrated using standard seawater (5) which is a calibration standard accepted in the oceanographic community. For some instruments, moorings for example, this calibration step can be performed before deployment and after recovery, and assuming linear drift. However, this drift is not always linear, and any intermediate calibration will be helpful in maintaining the quality of the data. For drifting or floating instruments it is not possible to bring the instrument aboard after recovery, as the instrument is usually not recovered after its lifetime is complete. Thus, SPURS will include a large number of salinity sensors which will need to be calibrated in situ, or compared with other instruments that are. As the experiment is going on, instruments will come into proximity with each other by serendipity or design, and will need to be compared. We will need to develop a database of instrument encounters. We will also need to understand how the intercalibration process will work once the instruments are in the water.

In this SPURS simulation, tracks of ships, simulated drifters and floats, moorings, etc., are calculated. I note the times and places assets come close enough to be considered simultaneous. Each time this occurs, I record the time and date, location, identity of each sensor involved. The idea here is to make sure that there are enough encounters to ensure that the instruments are going to be properly compared with each other. It is also to determine a threshold of time and space within which measurements can be considered simultaneous. In essence, this exercise is practice for when the real assets go into the water.

The other justification has to do with OSSEs (observing system simulation experiments). In an OSSE, a model of the ocean is run which includes salinity, temperature, currents, and many other variables (see, e.g. (7)). Observing assets are sent through the model as if they were observing the real ocean. Data is collected by the observing assets within the model and analyzed in the same way it would be if the asset were moving through the real ocean. One can then go back and look at the basic scientific questions asked at the outset of the experiment and see if the observing system as designed is adequate to answer those questions. In the case of SPURS, the primary question has to do with the balance of freshwater at the surface. Can the observing system determine this balance with sufficient accuracy? If not, are there alternative designs that might work better? This project is not about running an OSSE. That is way beyond my expertise, particularly in the modeling side. However, if I can create the observing system design to go into the OSSE, I can help those who run the models. What the modelers will need is a list of what observing assets will measure what variables, when and where. With my simulation I can provide this information.

Chapter 2: Background and Related Work

Here, I will review the observing assets to be simulated in this project, and the tools that will be used to created the visualization.

2.1 The Structure of SPURS

The SPURS experiment as currently planned consists of a large number of elements. I will now go through these elements in detail and describe them in some detail.

2.1.1 Cruises

There are currently 5 or 6 cruises planned that will do work related to SPURS in the SPURS region. The first will be a mooring service cruise by the NOAA ship Ron Brown (8), dates unknown. Next is a cruise on a French ship, the Thalassa, (9) in August 2012. Third is the main cruise on the R/V Knorr (10; Fig. 2), September 6 - October 9, 2012. A Spanish cruise has been proposed for 2013 and two more cruises on smaller US vessels are also likely in 2013 for mooring recovery. Ships have the obvious advantage that they can go wherever they are needed. They have shipboard instrumentation for doing profiles, underway instrumentation for collecting data while steaming, salinometers for doing high-precision calibration and can carry multiple sensors. Ships are also used to deploy the other platforms, moorings, gliders, etc. I have obtained approximate areas of operation for the Thalassa (11) and Knorr (12) cruises, as well as departure and return ports. (Recent word has been that the Ron Brown is being moved to the Pacific and may not be available for SPURS.)

2.1.2 Gliders

Gliders can be thought of as drones in the ocean. The are powered vehicles that move where they are directed, taking measurements along the way. Because they move very slowly, they can spend many months at sea doing repeat patterns. SPURS will have three types of gliders, Seagliders (13; Fig. 3), Wavegliders (16) and Slocum gliders (17). Seagliders and Slocum gliders profile up and down to a programmed depth, probably 200 m in SPURS. Wavegliders stay at the surface and are powered by the vertical motion of the sea surface. The gliders in SPURS will cover an area around the central flux mooring of between 10-20 km and 50-100 km. I did not include the Waveglider or Slocum gliders in the visualization as I could not get a definite track from the principal investigator. Three Seagliders are included in the visualization as the proposed tracks are available (18).

As they are going to be spending most of their time in the surface ocean, calibration is an issue for salinity sensors on gliders. It is helpful that they are usually recovered at the end of their deployments, and so can be taken back to the lab for post-calibration.

2.1.3 Floats

The Argo program (19) currently consists of 3500 floats placed randomly in the global ocean. Typical float missions have them spending most of their time at a large depth, typically 1000 m. They pop up to the surface at a programmed interval, typically 9 days, and send the profile they have collected to satellites and then to receiving stations on land. Unlike gliders, floats have no ability to control their location or movement, just their depth. However, with a deep parking depth, they do not typically move very far from where they were put into the water. Currents at 1000 m are weak. Calibration procedures for salinity measurements on floats are well developed (20, 21). Floats have stable calibration mainly due to their work cycle, which takes them as deep as 2000 m. This helps to prevent fouling as most organisms that foul surface instrumentation will not survive the pressures at that depth.

There will be 25 floats deployed as part of SPURS (22) on a 5X5 grid with about 0.5° spacing. The SPURS floats will have additional sensors on them beyond standard Argo floats, extra conductivity cells for short vertical scale measurements near the surface, an irridium satellite connection for two-way communication and hydrophones for measuring rainfall.

Float data for the Atlantic basin were obtained from the Global Argo Data Repository at the National Oceanographic Data Center (23). The data are stored in a compressed form as NetCDF files (24). NetCDF is a self-describing binary format used for array-based scientific data. It is a standard way of distributing data in the oceanographic and atmospheric sciences. MATLAB (see Section 2.2.3) has a set of routines to read and manipulate NetCDF files.

2.1.4 Drifters

As opposed to floats, drifters (25; Fig. 5) stay at the surface. The drifters deployed during SPURS will be part of the global drifter program, which currently has 875 instruments in the water globally (27). Drifters have a structure called a drogue, which acts as a sea anchor. It hangs down at a typical 15 m depth and allows the drifter to follow the currents at the drogue depth. There will be close to 200 drifters deployed during SPURS, 100 of which will have salinity sensors on them. This is in addition to another

100 or so that populate the North Atlantic at any one time. The simulated deployment of the drifters was done at positions given by one of the scientists (28).

It is not typical for drifters to have salinity sensors on them. The sensors can go quickly out of calibration due to fouling or ingestion of inorganic material into the cell. Thus, SPURS is a very unusual in having such a large deployment of salinity drifters. Thus, these instruments will be a major challenge in terms of keeping them calibrated.

Drifter data for the Atlantic basin were obtained from the Canadian Fisheries and Oceans Surface Velocity Program data archive (29). The drifter data there are in an awkward comma separated value (csv) format (see Appendix C) that was especially difficult to deal with, mainly because they included non-numeric characters in large files. I had to learn and use the unix stream editor (sed) command to to remove the non-numeric characters to get the data files into a form that I could read into MATLAB.

2.1.5 Moorings

There are 3 moorings that will be put in place solely for SPURS plus one nearby that will continue to collect data during the experiment. The main one, the central flux mooring (30; Fig. 6), will be collecting meteorological data at the surface, and ocean data below the surface. The flux mooring will be in place for about one year. The other two are experimental version of ATLAS (Autonomous Temperature Line Acquisition System; 32) moorings called prawlers (33). These have much more limited subsurface measurements, but similar surface measurements to the flux mooring and will be placed 25 km east and 25 km south of the central flux mooring. The other mooring is a PIRATA (PIlot Regional Array in the Tropical Atlantic; 34) mooring ~500 miles to the south of the flux mooring at (20°N, 38°W). This was not considered in this project.

2.1.6 Other Observing Assets

SPURS will have a number of other observing assets, including volunteer observing ships, microstructure floats, surface salinity sleds, etc. None of these was considered for this visualization, mainly because they will be deployed from a ship and thus easily intercalibrated.

2.2 Visualization of the Data

I will now describe some of the tools that I used to visualize the data.

2.2.1 Google Earth

The visualization was done using Google Earth (GE). GE has become a de facto standard in the area of geospatial visualization. The useful aspect to GE is that, once data are put into GE, they can be animated, zoomed in and out, and subsets turned on and off at the user's command. This makes GE easy to manipulate for the user and highly interactive. There are other similar programs (33), but none with the universal use of GE.

2.2.2 Keyhole Markup Language

The input to GE are .kml files, KML standing for Keyhole Markup Language. KML is a form of XML (eXtensible Markup Language) specially designed to describe geospatial data. It has become a standard for describing earth-bound objects and data (34, 35). It's schema has been published (36).

For objects described in KML, the location, time duration, altitude and other information are recorded. KML has a series of hierarchical tags that are used to describe geospatial objects as shown on the KML developer's page (37). These tags can describe an object's shape, location, time duration, motion, how it should be represented (e.g. what kind of icon or color line), viewed, etc. KML includes a standard series of tags, and a set of extensions. These extensions use the “gx” prefix and require using the namespace URI: `xmlns:gx="http://www.google.com/kml/ext/2.2" (38)`. I found that I needed to use some of these extensions to properly visualize my data, particularly the “gx:track” tag.

Data represented in GE can be easily distributed as KML files (or compressed as KMZ files). GE can read KML flies directly and they can be sent from one user to another.

2.2.3 MATLAB

Most of the programming for this project has been done in MATLAB. MATLAB (39) is a procedural language that is used commonly in science and engineering and in the oceanographic and geospatial communities in particular. There are substantial areas where discussion and help is available on the Mathworks (MATLAB's distributor) website. MATLAB was developed in the 1970's and 80's by Cleve Moler as a way to allow students to manipulate matrices without having to learn FORTRAN (40). UNCW recently obtained a campus-wide site license. MATLAB was the language of choice for this project because 1) the author is familiar with it and required no learning time, 2) there are toolboxes (extensions) to output KML (see below) and 3) the data used in this project, particularly the float data, come in a form that is most easily read in using routines available in MATLAB. MATLAB leaves something to be desired in terms of speed and the ability to do object-oriented programming, but that is more than made up in the fast development time resulting from its straightforward interface.

2.2.4 GE Toolbox

MATLAB can have extensions on the basic functionality, which are commonly called toolboxes. For example there is a statistics toolbox, a signal processing toolbox, a mapping toolbox. Most relevant for this project, I found and downloaded a GE toolbox (41). This includes a variety of functions to plot, draw, organize, and output eospatial information to KML. For example, the function “ge_output(F,S)” takes a string S, and writes it as KML to a filename F. F can then be read by GE. S can be created with functions like “S = ge_plot(X,Y)”, where X and Y are longitude and latitude data in a pair of matrices. Having functionality and an interface this easy made creation of KML files a straightforward operation.

Unfortunately, I found that the GE toolbox was only marginally useful in the end. I did use it to understand and get a feel for how KML works and how to output into KML files. The KML that I produced required the KML extensions (see red functions at (37); Section 2.2.2), which the GE toolbox did not support. I found the most convenient use of KML was to take a KML file that a colleague sent me (42), strip out the latitude, longitude and time information and insert my own information. The file show in Appendix A is one where I show how that is done. I did use the GE toolbox to help organize my KML files into folders (ge_folder function) and to output to files (ge_output).

Chapter 3: Procedure

The simulation was taken to be two years long, Jan. 1, 2012 – Dec. 31, 2013. The actual SPURS experiment is a little over one year, from August 1, 2012 – October 31, 2013. The steps taken to create the visualization and calibration file are as follows.

1. Download historical float and drifter data from the respective data assembly centers (sections 2.2.3 and 2.2.4). Floats and drifters deployed within the SPURS region (15-30°N, 30-45°W) were considered to be "SPURS" floats and drifters. Other floats and drifters were considered to be "background". 250 background floats and 100 background drifters were deployed at their given locations at some random time during the simulation period. The background floats and drifters were not used for intercalibration, just for visualization. 65 SPURS drifters and 25 SPURS floats were deployed in the SPURS region at times and locations specified by SPURS principal investigators. The drifters and floats used were chosen randomly among the ones downloaded from the respective archives.
2. Develop simulated ship and glider tracks for two ships, the Knorr and the Thalassa, and three gliders. These were created based on input from the scientists directing these efforts. Three moorings were also included, though of course these did not need detailed track information. The moorings and gliders were placed in the field for one year, Sept. 16, 2012 – Sept. 15, 2013.
3. Creation of the simulation
    1. Interpolate tracks of ships, gliders, moorings, drifters and floats to one hour intervals. Floats were considered to stay at the surface for 1 hour every 10 days.
    2. For each asset, create a track file, which includes time, latitude and longitude.
    3. Bundle those track files into folders.
    4. Write the folders to a simulation KML file using the GE toolbox.
    5. Put the interpolated track information into MATLAB structure arrays. Output these cell arrays to a MATLAB .mat file.
4. Development of the intercalibration file
    1. Load the interpolated track information (3.5 above).
    2. Go though the experiment hour by hour. At each hourly interval:
        1. Gather all assets that are present at that hour. Compare positions of assets. For all assets whose positions differ by less than 0.1 degrees (~10 mile):
            1. Note time, locations of both assets and sensor IDs.
    3. The full list of asset positions and sensor IDs are known as the calibration file. Later, during the actual intercalibration, this file will include sensor readings. Write the calibration file to disk as a MATLAB cell array.
5. Animate the calibration file
    1. Input the calibration file, step 4.3.
    2. For each entry in the calibration file, create a KML placemark.
    3. Gather all placemarks and write to disk using the GE toolbox.

The code used to develop the intercalibration file (part 4 above) is included in Appendix B.

The interpolated positions were output as a set of structure arrays (step 3.5). Structure arrays are a MATLAB data structure that resembles a database. Data are stored in a set of fields, which can contain data of any type or size. For example, a patient structure array might contain fields such as ID#, name, test results, etc. (Fig. 7). In my case, a float structure would contain entries for each float including an ID, longitudes, latitudes and date/time.

The intercalibration file was output (step 4.3) as a different MATLAB data structure, a cell array. Cell arrays are simply containers for other matrices (43).

A screenshot of the entire animation is shown in Fig. 8. A zoomed in view of the central area of the experiment is shown in Fig. 9. Two KML files are posted at [URL]. These should be opened with GE.

Though the procedure described above appears together, the visualization was done in parts. I visualized the ship tracks and buoys first as they were easiest. Next I did the floats and drifters. I checked on the veracity of the visualization by examining a few individual float and drifter tracks to see that they agreed with the raw data. For the intercalibration points, the purple and black markers in Fig. 9, I examined at some sample encounters in detail, comparing the tracks of the two instruments that met, the times and dates output to the calibration file, and the placement of the purple symbols on the visualization. The correctness of a few of these convinced me that they are all correct.

Chapter 4: Analysis of the Intercalibration File

There were approximately 5900 points in the intercalibration file. For example, Fig. 9 shows a screenshot where a number of assets were in close enough proximity to generate entries.

Examination of the calibration file reveals that of 98 observing assets, (65 drifters + 25 floats + 3 gliders + 3 moorings + 2 ships), 87 showed up in the calibration file. All floats had at least one proximity, with an average of 1.9. All but 10 drifters had at least one proximity with an average of 72. One ship did not appear, the Thalassa, but that was a result of careless placement of the simulated track. When assets met, the mean distance between them was 5.7 km with a distribution as shown in Fig. 10.

The calibration scheme appears to work relatively well. The most problematic of the assets, the drifters, had the most encounters, though there were 10 drifters with no encounters at all. (This is not exactly true as the drifters when deployed will have at least one calibration point.) Nevertheless, care should be taken during the experiment, if possible, to recover drifters or generate proximities with assets that can be controlled like gliders or ships. It appears from the animation that the gliders generate a lot of encounters with other assets. Thus, the gliders are crucial to tying the experimental calibration together. It would thus be highly recommended that during the year-long experiment the gliders be gathered together at one point on a regular basis, say monthly, in proximity to one of the moorings. This way, the gliders can be tied to each other more directly to greatly enhance the connectivity of the experimental assets to each other.

Chapter 5: Conclusions and Future Work

The most frustrating part of the this whole experience has been the functionality of the “Time Slider” in GE. I created a 2-year long animation, but most of the action occurs during about a 3-month period, August 1 – November 1, 2012. I tried to slow down the animation during that time period and focus in on it. There is a time zoom function, and bracket bars to determine the beginning and end of the data to be displayed. However, I was not able to reliably display exactly the time period I wanted at exactly the right animation speed. The bracket bars would jump around unpredictably, the time zoom buttons would gray out for no reason, the animation speed would be very difficult to control and go either too fast or too slow. Despite much trial and error and many attempts to find answers on the numerous GE forums, I was never able to figure out how to do this exactly right. This is not something that is controlled within the KML file. Maybe someday Google will come out with a version of the time slider that is more transparently easy to use.

Except for this issue, GE is a great tool to work with. KML files are very straightforward and easy to manipulate. There is a wealth of resources out there on the internet for working with them, sample files, icons, Javascript scripts, etc. It is easy to embed GE files into web pages using the GE plugin. KML files can be distributed as KMZ files that are compressed versions of the KML. For example, the main simulation file I created was 11 MB in size uncompressed, but only 933 kB when compressed, a compression factor of more than 10.

I felt I learned a lot from this exercise, how to use and manipulate KML files, how to put data into structure and cell arrays in MATLAB, how to read in text into MATLAB and replace parts of it, how to use the “sed” function in unix. I also got a good idea of how the experiment is going to go. The cruise on the Knorr will be extremely busy with mooring, glider and float deployments, and a planned suite of microstructure measurements. This exercise will help the chief scientist to plan the activities on the cruise.

The criterion for proximity was 0.1° and 1 hour. I did not try any different criteria to see how the number of encounters depends on them. The main reason for not doing this was because I was more interested in the process of visualization than experimenting with the method. Also, the process of creating the visualization and calibration file was somewhat time-consuming. Doing this multiple times would have been cumbersome in this regard. It would be straightforward to streamline the process with a goal of analyzing the proximities and how they depend on different variables like distance. This would be an easy extension of the present work.

This project was a simulation of the SPURS experiment in the sense that it was "a model designed to imitate the time-evolution of a real system" (44). However, most such simulations involve multiple runs with varying conditions (called jitter) with the output analyzed in a statistical framework. To better understand how the experiment and the intercalibration process will work, it would be better to rerun the simulation in this way. In doing this, we will need to develop specific metrics to determine the quality of the intercalibration. Examples of such metrics would be total number of proximities, total number of proximities per sensor, number of sensors with zero or one proximity. An interesting study would be to see how such metrics vary with things such as the total number of sensors deployed, the defined distance and time for a proximity, the mixture of vehicle types and the tracks taken by vehicles that are under human control. These are variables that could be changed in multiple-run simulations to compare with experimental metrics. The visualization part of the project, where the observing assets and intercalibration points were seen (Fig. 9) helps to understand how the assets are going to interact, but more quantitative information is needed. This is possible future direction for this work.

Ultimately when the experiment does occur, and the data are available, it may be possible to use the “web of trust” idea (45) to help with the instrumental intercalibration. In a web of trust, trust is distributed among a set of participants through a series of introductions. There are some good analogies between the SPURS experiment and the web of trust. There will be an authoritative source of calibration, the salinometer on board the ship. Connected to this source are the various other instruments which will be connected to each other by proximities. Some proximities will be better than others, closer in space or in actual measured salinity. The precision of a particular measurement will be determined by the mean trust distance between it and the authoritative source. Again, this is another possible future direction for this work.

Acknowledgments

Special thanks to my advisor Dr. Bryan Reinicke and my committee members Drs. Kline and Vetter. This work was funded by NASA under grant #NNX11AE83G.

References

(1) http://spurs.jpl.nasa.gov. Accessed April 23, 2012.

(2) Salinity Processes in the Upper-Ocean Regional Study (SPURS). SPURS Workshop Report, January 2010. Unpublished white paper available at http://spurs.jpl.nasa.gov. Click on “Meetings”, “Pasadena Workshop Dec 2009”, “Workshop Report”.

(2) http://upload.wikimedia.org/wikipedia/commons/3/3e/Wiki_plot_04.png. Accessed April 23, 2012.

(3) UNESCO, Technical Papers in Marine Science, 36, 1981

(4) http://www.seabird.com/technical_references/condpaper.htm. Accessed April 23, 2012.

(5)http://www.osil.co.uk/Products/OtherMarineInstruments/tabid/56/agentType/View/PropertyID/67/Default.aspx. Accessed April 23, 2012.

(6)http://www.osil.co.uk/Resources/SeawaterTechNotes/tabid/104/articleType/ArticleView/articleId/219/IAPSO-Standard-Seawater-and-the-Practical-Salinity-Scale.aspx. Accessed April 23, 2012.

(7) Ballabrera-Poy, J. et al., An Observing System Simulation Experiment for an Optimal Moored Instrument Array in the Tropical Indian Ocean. J. Climate, 20, 3284. DOI: 10.1175/JCLI4149.1

(8) http://www.moc.noaa.gov/rb/. Accessed April 23, 2012.

(9) http://flotte.ifremer.fr/fleet/Presentation-of-the-fleet/Vessels/Deep-sea-vessels/Thalassa/. Accessed April 23, 2012.

(10) http://www.whoi.edu/page.do?pid=8157. Accessed April 23, 2012.

(11) Reverdin, G., personal comm.

(12) Schmitt, R. personal comm.

(13) http://www.apl.washington.edu/projects/seaglider/summary.html. Accessed April 23, 2012.

(14) http://www.awi.de/en/news/press_releases/detail/item/successful_series_of_measurements_in_arctic_sea_ice_rv_polarstern_completes_work_in_the_fram_str/?cHash=350480fc5ff53b38f74b8ddca39e3a84. Accessed April 23, 2012.

(15) http://www.apl.washington.edu/projects/seaglider/images/profiling.jpg. Accessed April 23, 2012.

(16) http://liquidr.com/technology/wave-glider-concept/. Accessed April 23, 2012.

(17) http://www.webbresearch.com/slocumglider.aspx. Accessed April 23, 2012.

(18) Lee, C. et al., 2012. Seaglider Science Objectives. Presented at the SPURS Science Team Meeting in Seattle, WA, Jan. 2012. Presentation available at http://spurs.jpl.nasa.gov, click on “Meetings”, “Seattle Jan. 2012”. Scroll down to PI-9.

(19) http://www.argo.ucsd.edu/ . Accessed April 23, 2012.

(20) Riser, S. et al., 2008. Salinity in Argo: A Modern View of a Changing Ocean. Oceanography, 21, 56.

(21) Wong, A. et al., 2003. Delayed-mode Calibration of Autonomous Profiling Float Salinity Data by theta-S Climatology. J. Atmos. Ocean. Tech., 20, 308-318.

(22) Riser, S. et al., 2012. The Use of Profiling Floats in SPURS: T, S, Wind and Rain. Presented at the SPURS Science Team Meeting in Seattle, WA, Jan. 2012. Presentation available at spurs.jpl.nasa.gov, click on “Meetings”, “Seattle Jan. 2012”. Scroll down to PI-8.

(23) http://www.nodc.noaa.gov/argo/latest_data.html. Accessed April 23, 2012.

(24) http://www.unidata.ucar.edu/software/netcdf/. Accessed April 23, 2012.

(25) http://www.aoml.noaa.gov/phod/dac/gdp_drifter.php. Accessed April 23, 2012.

(26) http://www.adp.noaa.gov/images/drifterschematic_hr.jpg. Accessed April 23, 2012.

(27) http://www.aoml.noaa.gov/phod/dac/dacdata.php. Accessed April 23, 2012.

(28) L. Centurioni, pers. comm.

(29) http://www.meds-sdmm.dfo-mpo.gc.ca/isdm-gdsi/drib-bder/svp-vcs/index-eng.asp. Accessed April 23, 2012.

(30) http://uop.whoi.edu/buoyschematic.html. Accessed April 23, 2012.

(31) http://uop.whoi.edu/projects/whots/whots.html. Accessed April 23, 2012.

(32) http://www.pmel.noaa.gov/tao/proj_over/mooring.shtml. Accessed April 23, 2012.

(33) Meinig, C., 2007. Development of a Next Generation Platform and Instrumentation for Continuous Ocean Observations (PICO). Unpublished manuscript, Pacific Marine Environmental Laboratory. Accessed at http://www.oco.noaa.gov/docs/arsooosc07/chapterII/03-b-Meinig_FY07_PICO.pdf on April 23, 2012.

(34) http://www.pmel.noaa.gov/tao/doc/PIRATA_BAMS_2008.pdf. Accessed April 23, 2012.

(33) http://www.brighthub.com/internet/google/articles/61335.aspx. Accessed April 23, 2012.

(34) http://www.opengeospatial.org/. Accessed April 23, 2012.

(35) Open Geospatial Consortium, 2008. OGC KML. Report OGC 07-147r2.

(36) http://schemas.opengis.net/kml/. Accessed April 23, 2012.

(37) https://developers.google.com/kml/documentation/kmlreference. Accessed April 23, 2012.

(38) https://developers.google.com/kml/documentation/kmlreference#kmlextensions. Accessed April 23, 2012.

(39) http:www.mathworks.com. Accessed April 23, 2012.

(40) http://www.mathworks.com/company/newsletters/news_notes/clevescorner/dec04.html?s_cid=wiki_matlab_3. Accessed April 23, 2012.

(41) http://www.mathworks.com/matlabcentral/fileexchange/12954. Accessed April 23, 2012.

(42) D'Asaro, personal comm.

(43) http://www.mathworks.com/help/techdoc/ref/cell.html. Accessed April 24, 2012.

(44) Hartmann, S., 1996. Simulation and Modeling in the Social Sciences from a

Philosophy of Science Point of View. R. Hegelsmann et al. eds., Theory and Decision Library. Kluwer: Dordrecht, pp. 77-100.

(45) Golbeck, J., et al., 2003. Trust Networks on the Semantic Web. Lecture Notes in Computer Science, 2782, 238-249, DOI: 10.1007/978-3-540-45217-1_18.

Appendix A

A Sample KML File

The text “coordinatesAndTimesGoHere”, “beginTimeGoesHere” and “endTimeGoesHere” are all replaced with appropriate values for my own dataset. The icon “icon_circle.png” can be replaced with another if desired.

Note, this file uses the kml extensions shown in red on this page (36).

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2."
xmlns:gx="http://www.google.com/kml/ext/2.2">
<Document>
<name>
File_Name_Goes_Here.kml
</name>
  <Placemark>
   <name>Drifter</name>
   <Style>
    <LineStyle>
     <color>FF0033FF</color>
    </LineStyle>
    <IconStyle>
     <color>FF0033FF</color>
     <Icon>
      <href>icon_circle.png</href>
     </Icon>
     <scale>0.2</scale>
    </IconStyle>
    <LabelStyle>
     <color>00000000</color>
     <scale>0.5</scale>
    </LabelStyle>
    <ListStyle>
     <ItemIcon>
      <state>closed</state>
      <href>icon_circle.png</href>
     </ItemIcon>
     <bgColor>FF0033FF</bgColor>
    </ListStyle>
   </Style>
   <TimeSpan>
    <begin>beginTimeGoesHere</begin>
    <end>endTimeGoesHere</end>
   </TimeSpan>
   <gx:Track>
     coordinatesAndTimesGoHere
   </gx:Track>
  </Placemark>
</Document>
</kml>
```

Appendix B

MATLAB Code for producing the intercalibration file

This is the MATLAB code used to generate the calibration file. The input is the SPURS observing asset positions interpolated to hourly intervals. Output is the calibration file, which is a list of locations, times and sensor IDs associated with proximities between sensors. Note the proximity condition is expressed as

```
   distArray = sqrt( (latSav(i1)-latSav(i2))^2 + (lonSav(i1)-
lonSav(i2))^2 );
   if (distArray < 0.1)
…
```

The distance is expressed as the Euclidean distance in latitude and longitude terms. This is a very crude way of expressing distance on the earth, but a more accurate way would be slower and more cumbersome. Note, 0.1° of latitude or longitude on the equator is about 6 nautical miles or 10 km.

```
% Create calibration file and examine

clear
load '/Users/bigkahuna/Documents/SPURS DMS/Data/SPURS Asset Structure
Arrays.mat'
iFlux = 1;
iPrawler = ones(1,2);
iKnorr = 1;
iThalassa = 1;
iDrifter = ones(1,length(drifterStruct));
iFloat = ones(1,length(floatStruct));
iGlider = ones(1,3);

idateSPURS = [datenum(2012,1,1):(1/24):datenum(2014,1,1)];
nTime = length(idateSPURS);
ical = 1;
calFile=cell(1,5);
calFile{4} = cell(1,1);
calFile{5} = cell(1,1);
for iTime = 1:nTime
    iSav = 1;
    lonSav = []; latSav = []; IDSav = {};
    if ( (iFlux<=length(fluxStruct(1).idate)) &&
((fluxStruct(1).idate(iFlux)-idateSPURS(iTime)) < (1/48)) )
        lonSav(iSav) = fluxStruct(1).xlon(iFlux);
        latSav(iSav) = fluxStruct(1).ylat(iFlux);
        IDSav{iSav} = fluxStruct(1).ID;
        iSav = iSav + 1;
        iFlux = iFlux + 1;
    end
    for ip=1:length(prawlerStruct)
        if ( (iPrawler(ip)<=length(prawlerStruct(ip).idate)) &&
((prawlerStruct(ip).idate(iPrawler(ip))-idateSPURS(iTime)) < (1/48)) )
            lonSav(iSav) = prawlerStruct(ip).xlon(iPrawler(ip));
            latSav(iSav) = prawlerStruct(ip).ylat(iPrawler(ip));
            IDSav{iSav} = prawlerStruct(ip).ID;
            iSav = iSav + 1;
            iPrawler(ip) = iPrawler(ip) + 1;
        end
    end
    if ( (iKnorr<=length(knorrStruct(1).idate)) &&
((knorrStruct(1).idate(iKnorr)-idateSPURS(iTime)) < (1/48)) )
        lonSav(iSav) = knorrStruct(1).xlon(iKnorr);
        latSav(iSav) = knorrStruct(1).ylat(iKnorr);
        IDSav{iSav} = knorrStruct(1).ID;
        iSav = iSav + 1;
        iKnorr = iKnorr + 1;
    end
    if ( (iThalassa<=length(thalassaStruct(1).idate)) &&
((fluxStruct(1).idate(iThalassa)-idateSPURS(iTime)) < (1/48)) )
        lonSav(iSav) = thalassaStruct(1).xlon(iThalassa);
        latSav(iSav) = thalassaStruct(1).ylat(iThalassa);
        IDSav{iSav} = thalassaStruct(1).ID;
        iSav = iSav + 1;
        iThalassa = iThalassa + 1;
    end
    for ip=1:length(drifterStruct)
        if ((iDrifter(ip)<=length(drifterStruct(ip).idate)) &&
((drifterStruct(ip).idate(iDrifter(ip))-idateSPURS(iTime)) < (1/48)) )
            lonSav(iSav) = drifterStruct(ip).xlon(iDrifter(ip));
            latSav(iSav) = drifterStruct(ip).ylat(iDrifter(ip));
            IDSav{iSav} = drifterStruct(ip).ID;
            iSav = iSav + 1;
            iDrifter(ip) = iDrifter(ip) + 1;
        end
    end
    for ip=1:length(floatStruct)
        if ( (iFloat(ip)<=length(floatStruct(ip).idate)) &&
((floatStruct(ip).idate(iFloat(ip))-idateSPURS(iTime)) < (1/48)) )
            lonSav(iSav) = floatStruct(ip).xlon(iFloat(ip));
            latSav(iSav) = floatStruct(ip).ylat(iFloat(ip));
            IDSav{iSav} = floatStruct(ip).ID;
            iSav = iSav + 1;
            iFloat(ip) = iFloat(ip) + 1;
        end
    end
    for ip=1:length(gliderStruct)
        if ( (iGlider(ip)<=length(gliderStruct(ip).idate)) &&
((gliderStruct(ip).idate(iGlider(ip))-idateSPURS(iTime)) < (1/48)) )
            lonSav(iSav) = gliderStruct(ip).xlon(iGlider(ip));
            latSav(iSav) = gliderStruct(ip).ylat(iGlider(ip));
            IDSav{iSav} = gliderStruct(ip).ID;
            iSav = iSav + 1;
            iGlider(ip) = iGlider(ip) + 1;
        end
    end
    for i1=1:length(latSav)
        for i2=i1+1:length(latSav)
            distArray = sqrt( (latSav(i1)-latSav(i2))^2 + (lonSav(i1)-
lonSav(i2))^2 );
            if (distArray < 0.1)
                calFile{1} = [calFile{1} ; [lonSav(i1) latSav(i1)]];
                calFile{2} = [calFile{2} ; [lonSav(i2) latSav(i2)]];
                calFile{3} = [calFile{3} ; idateSPURS(iTime)];
                if (isnumeric(IDSav{i1}));
                    s1 = int2str(IDSav{i1});
                else
                    s1 = IDSav{i1};
                end
                calFile{4} = [calFile{4} ; s1];
                if (isnumeric(IDSav{i2}));
                    s1 = int2str(IDSav{i2});
                else
                    s1 = IDSav{i2};
                end
                calFile{5} = [calFile{5} ; s1];
            end
        end
    end
end
```

```
junk = calFile{4};
junk = junk(2:end);
calFile{4} = junk;
junk = calFile{5};
junk = junk(2:end);
calFile{5} = junk;

clearvars -except calFile
```

Appendix C

Sample Drifter Data

This the header and the top 4 lines of a sample drifter data .csv file. The columns are self-explanatory, but the “/” and “:” characters included in the dates could not be easily read into MATLAB and had to be replaced using the sed utility.

```
   Contents:
 Col  1 = Platform identifier (ARGOS #)
 Col  2 = Observation year/month/day hour:minute (UTC)
 Col  3 = Latitude of observation (+ve North)
 Col  4 = Longitude of observation (+/- 180 deg +ve West of Greenwich)
 Col  5 = SSTP - Sea surface temperature (deg. C)
 Col  6 = EWCT - East component of current (+ve East)
 Col  7 = NSCT - North component of current (+ve North)
 Col  8 = ELA$ - The error in latitude (degrees)
 Col  9 = ELO$ - The error in longitude (degrees)
 Col 10 = EXP$ - The originator's experiment number
 Col 11 = WMO$ - WMO platform identifier number
 Col 12 = Drogue on/off - 1 = attached; 0 = not
 Note: Missing value indicated by 999.9999
 DATA
0220177,1979/02/15 06:00, -3.621, 85.708, 26.3250,    -0.0823,     0.1734, 0.0006, 0.0002, 40, 0,1
0220177,1979/02/15 12:00, -3.585, 85.726, 26.3430,    -0.1096,     0.1955, 0.0006, 0.0002, 40, 0,1
0220177,1979/02/15 18:00, -3.545, 85.750, 26.5910,    -0.1601,     0.2162, 0.0001, 0.0000, 40, 0,1
0220177,1979/02/16 00:00, -3.500, 85.788, 26.5420,    -0.2264,     0.1335, 0.0001, 0.0000, 40, 0,1
```

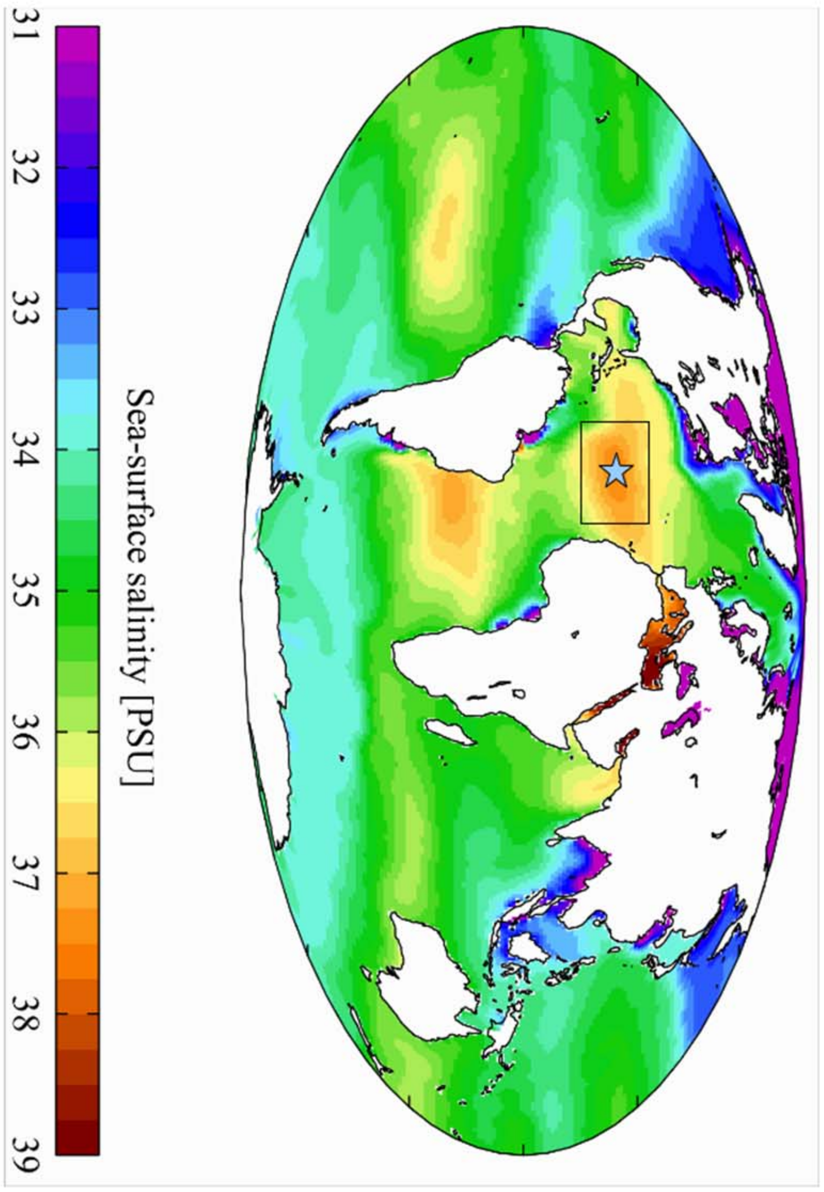
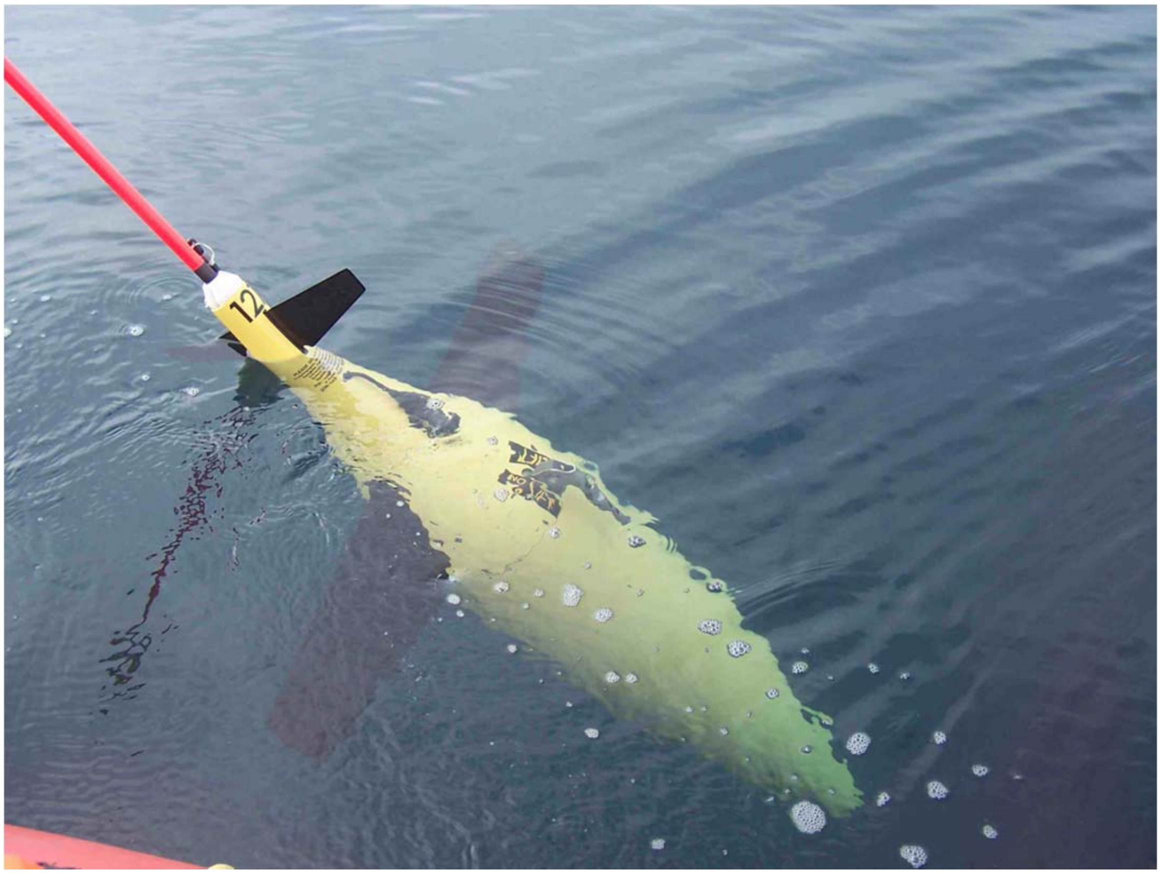


Figure 1. A map of global sea surface salinity in PSU (practical salinity units). (2). Superimposed is a box showing the SPURS region in the North Atlantic, and a * indicating the location of the flux mooring at the center. Note, PSU is the unitless measure of salinity in common use.

Figure 2. The research vessel, R/V Knorr (10).



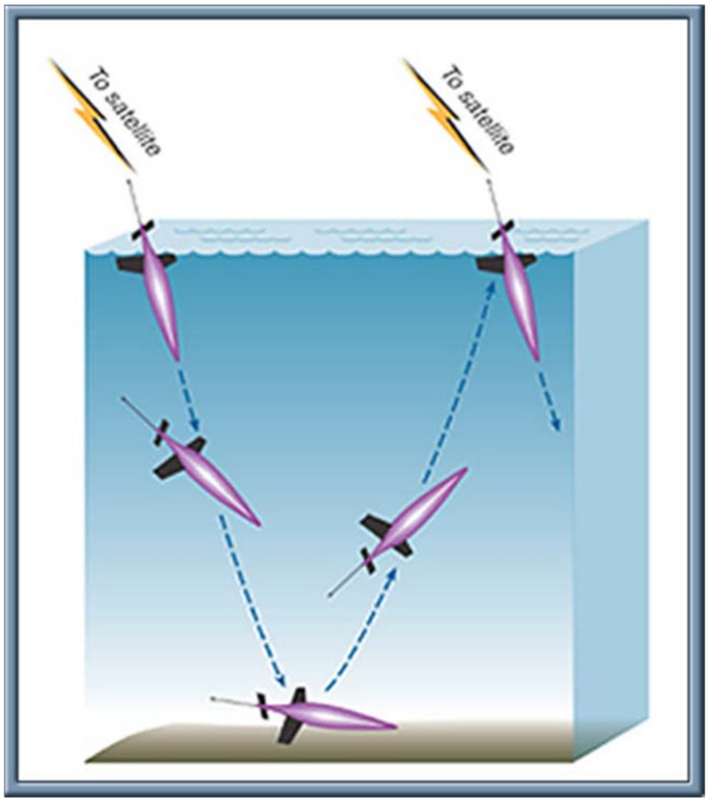


Fig. 3 Top. A Seaglider in the water. (14) Bottom, a schematic showing the undulating path that a Seaglider takes as it moves around in the ocean (15).

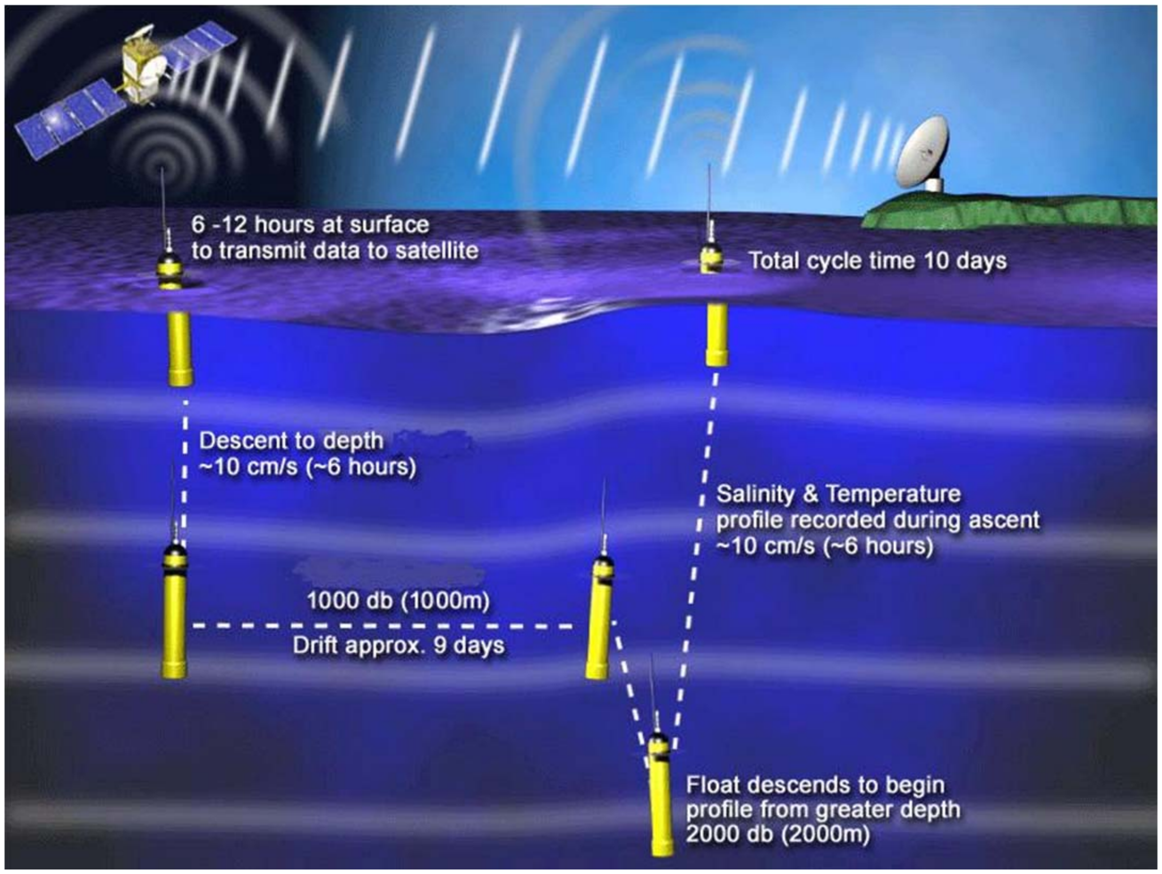


Figure 4. The work cycle of an Argo float. (19)

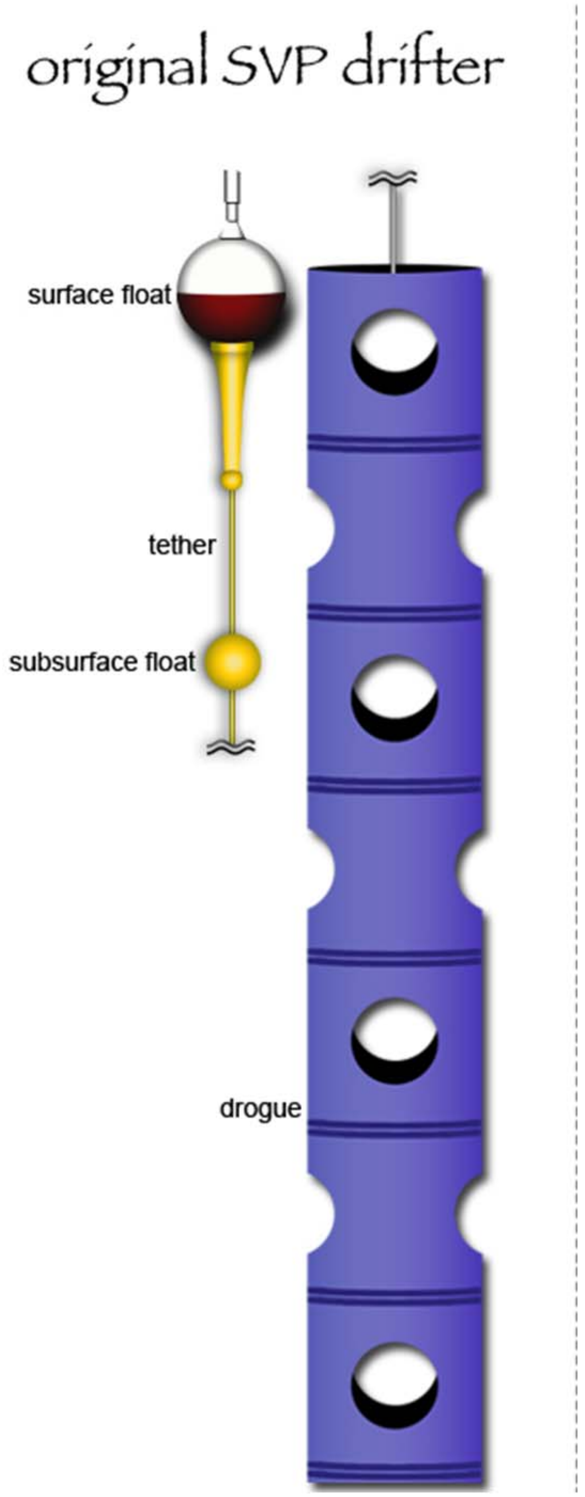


Figure 5. A schematic picture of a surface drifter and its drogue (26).

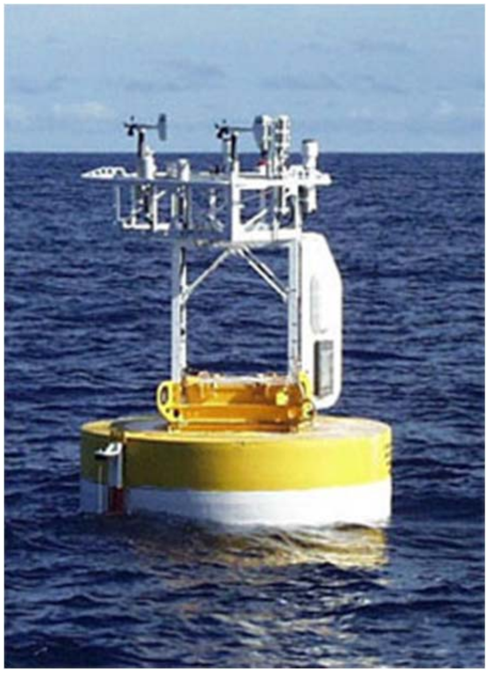


Figure 6. A version of the flux mooring to be deployed at (25°N, 38°W). Note the meteorological sensors. There are more sensors under the surface not visible in this photo. (30)

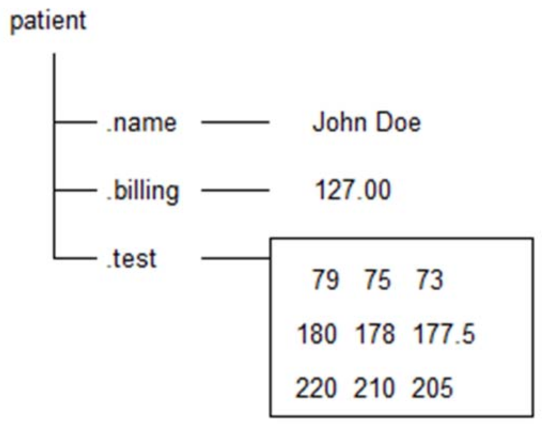


Figure 7. This shows an example of a structure array called patient, which contains the name, billing ID and test results (43). The variables are assigned with the statements

```
patient(1).name = 'John Doe';
patient(1).billing = 127.00;
patient(1).test = [79 75 73; 180 178 177.5; 220 210 205];
```

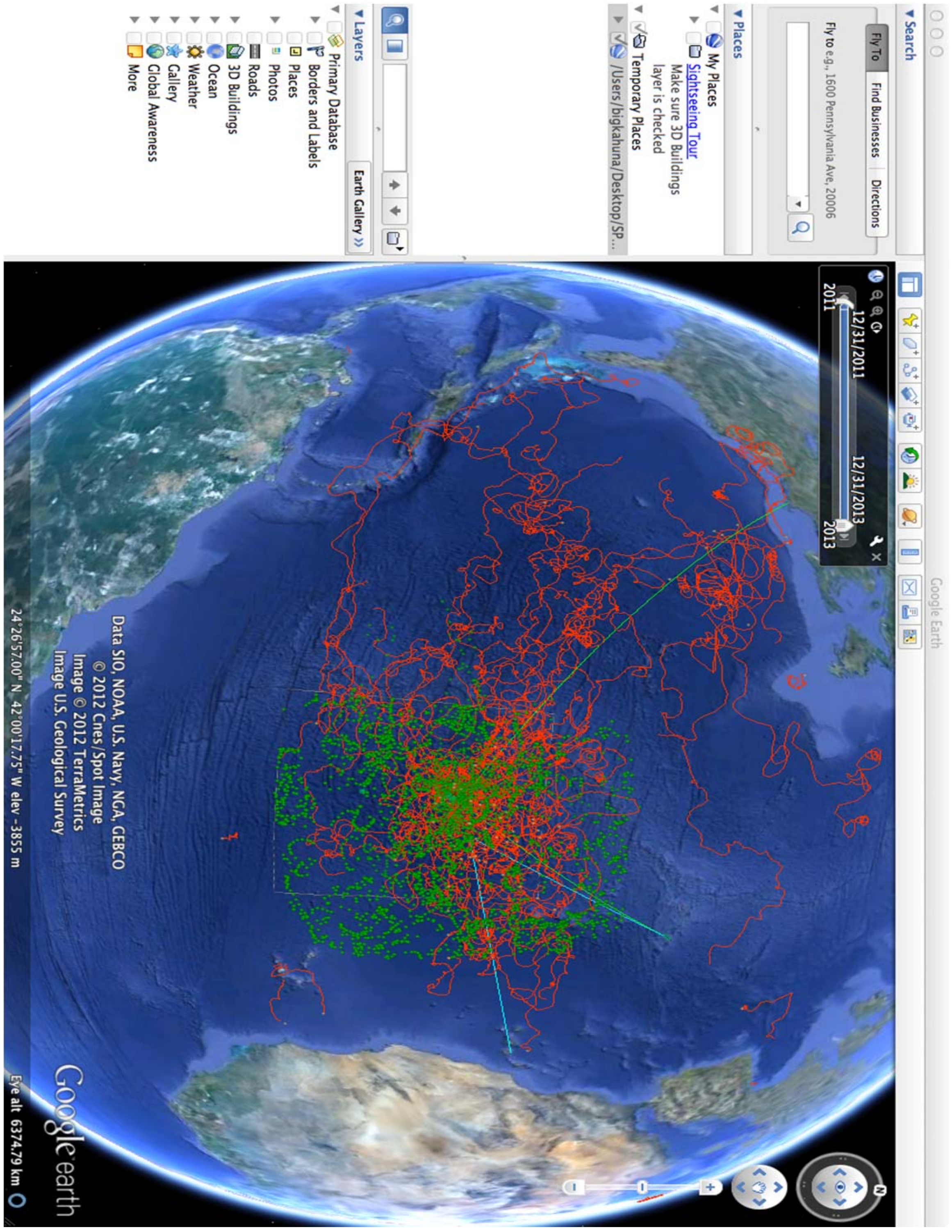


Figure 8. A screenshot of the final animation (without the intercalibration points) in GE. Green symbols are float positions. Red tracks are drifters.

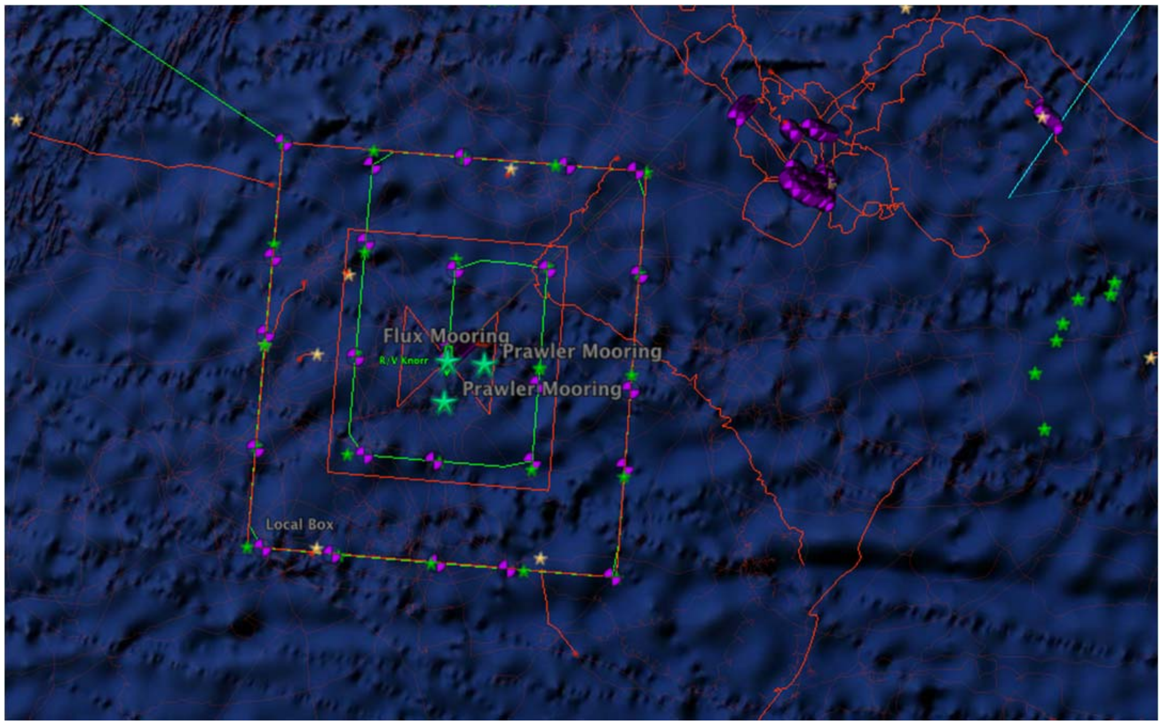


Figure 9. A screenshot of the visualization during the period August 14 – September 16, 2012. The flux mooring and prawler moorings are shown as green *'s. Purple and black circles are locations where assets came together close enough to be compared, i.e. entries in the intercalibration file. The SPURS floats have just been deployed by the Knorr in a 5X5 grid. Each one was intercalibrated with the Knorr when it was put into the water. The red boxes are tracks that the seagliders will occupy, though they still have not been deployed. The R/V Knorr is in the center of the array. The track of the Thalassa, which has just departed the area, is seen at the top right in light blue.

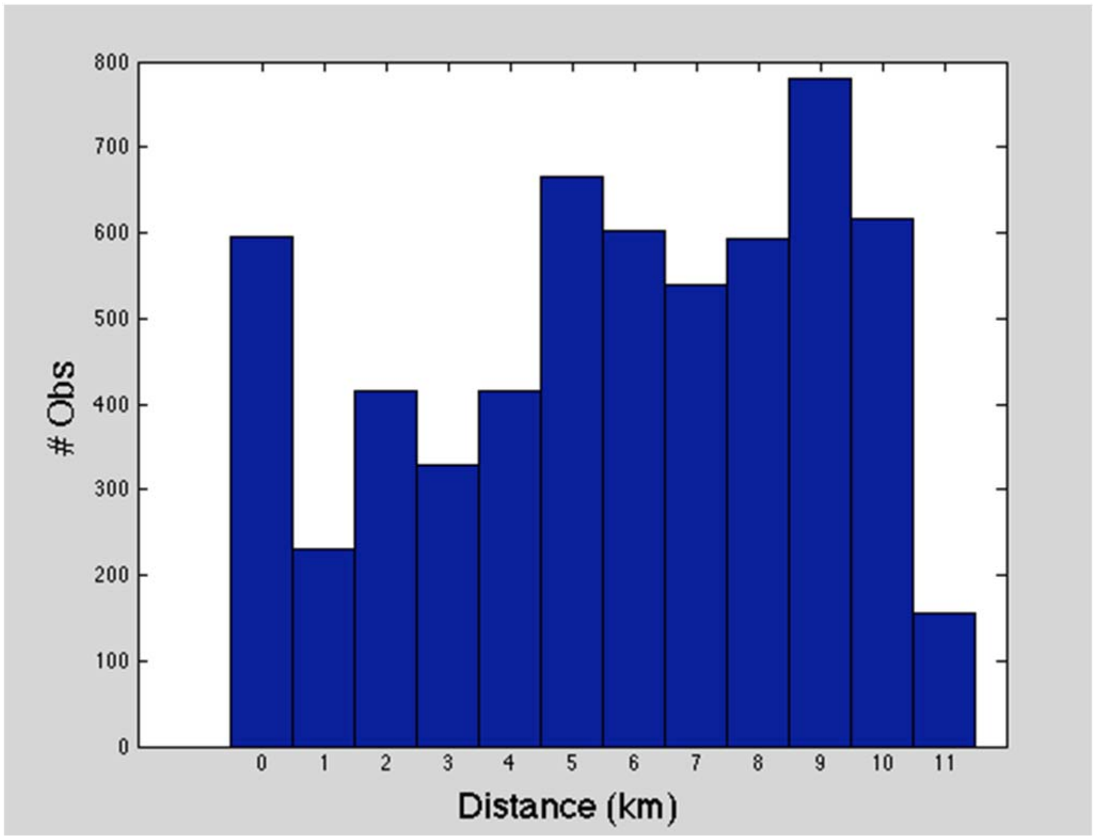


Figure 10. A histogram of the number of asset proximities (y-axis) vs. the distance in km. The mean distance was 5.7 km.