Extracting valuable information from a simultaneous visualisation of science and engineering data obtained with an autonomous underwater vehicle

Robert Robinson
School of Computing and Information Systems, UTAS, Hobart 7000 TAS, Australia

Andrew Davie, Paulo de Souza Jr.
Tasmanian ICT Centre, CSIRO
GPO Box 1538
CSIRO, Hobart 7000 TAS, Australia

***Abstract*- The aim of this study was to create a software solution for the visualization of scientific and engineering data from an Autonomous Underwater Vehicle (AUV) to improve analysis of performed missions. A media player style interface was created to playback and interpret AUV mission data in an interactive format while retaining a familiar and user friendly interface for researchers. This allows faster analysis and quick cross-parameter referencing without the need for hardware intensive 3D representations. This tool allows observations and inferences to be made very quickly that might otherwise be difficult to find as a consequence of the volume of the data being analyzed. This software can be applied to any time series data available in CSV format and to other autonomous machines. Being implemented in python, this tool can be used on Windows, Mac or Linux machines making it available to a wider user base.**

I. INTRODUCTION

Our main motivation was to create a software solution to effectively playback AUV mission data to support effective analysis and selection of data. The Starbug AUV is an autonomous hand-launched submarine measuring approximately 1.2m in length, used by CSIRO to investigate various marine environments [1], including the Derwent and Huon estuaries in southern Tasmania as part of the TasMAN Project [2]. It carries a variety of sensors that collect a range of scientific and engineering data. Some scientific parameters are turbidity, conductivity, temperature and salinity while some of the collected engineering parameters are depth, position, heading and roll. These data are recorded asynchronously by different instruments and are logged in a time stamped format. A typical AUV mission generates substantial amounts of data, for example, potentially 1GB/hour with images sampling at 5 Hz and other sensors at 1-20 Hz. At this rate, a set of mission data can easily become difficult to analyze and interpret simply because of the volume of data. There is currently no simple way to analyze the logged data and it can be difficult to accurately cross reference data, for example finding the conductivity at the exact point in time when a specific photo was taken. Currently paper based plots are generated and analyzed; however these plots lose the important factor of time in translation and can become useless without prior knowledge of the mission parameters. Scientific data is better understood and analyzed when context is provided. Context is given through engineering data such as depth, position, orientation and accelerations. This software to bring these data sets to life and makes it easier to cross reference points of interest. Cross referencing position with other variables, especially images, is very important as much information can be gathered from the images but without knowing where they are taken they, and reduces their value in building spatial and temporal relationships. It was also important for the software to be able to run on low level machines so that it could be accessible to many users. For this reason the interface has remained in 2D, avoiding the hardware requirements of larger 3D visualization tools.

The problem of trying to simultaneously visualize large quantities of interrelated data is common to any autonomous machine (e.g., rovers on Mars [3], spacecraft [4], ROVs, autonomous airplanes [5], to mention a few).

This study describes the development of a software tool that is able to combine synchronized engineering and scientific data into a single graphical interface quickly and coherently, making further analysis simpler.

II. OUR AUTONOMOUS UNDERWATER VEHICLE

AUV's are a sub-class of unmanned submersibles. Typically unmanned submersibles fall into several categories including tethered, untethered and even towed. An AUV is an untethered submersible that requires no communication during the execution of its prescribed mission [6].

A typical AUV mission consists of three stages; deployment, execution and retrieval. During deployment the AUV's internal systems are activated and logging begins. During execution, the AUV makes several manoeuvres and continually logs data from its sensors. One such manoeuvre is a dive operation as seen in Figure 2. The AUV resurfaces many times during a typical mission and due to this behaviour can pinpoint its position using GPS. A typical mission position graph can be seen overlayed onto a satellite image of the AUV's mission area near Apollo Bay in Tasmania in Figure 3. During retrieval the AUV is deactivated and the log files are downloaded from the onboard computer.

978-1-4244-5222-4/10/$26.00 ©2010 IEEE

Starbug uses DDX (Dynamic Data eXchange), running under Linux, as the basis for its onboard computer. DDX is a software platform for building distributed robot controllers [7]. The core components of a DDX system are known as the store and the catalogue. The store provides an efficient data storage medium for client programs (e.g. navigation and manoeuvring programs and instrument interfaces). A distributed robot controller can have many stores, it is the catalogue which allows all clients access to any data recorded in the store. The catalogue does this by keeping a list of all connected stores and a list of all current values that are being kept by any store. The catalogue also manages inter store communications, allowing a client connected to one store, to access values that are only known by another store.

III. Engineering Data

Engineering data are collected by Starbug to give the collected scientific data context. For example, a conductivity measurement is of limited use without at least a location and a depth. Engineering data collected by Starbug includes depth, distance to sea bed, location, orientation, heading and time. See Table I for a list of the engineering data collected by Starbug, the instrument which collects it, the reason for collecting this parameter and the format and frequency of the data. All engineering and scientific data is stored in the DDX store.

TABLE I
SUMMARY OF RELEVANT ENGINEERING DATA BEING COLLECTED BY STARBUG.

Parameter	**Sensor**	**Application**	**Format and frequency**
Depth	SSI 100psi pressure probe SSI 500psi pressure probe	Determine the depth of Starbug under water.	Double precision floating point recorded at 20 Hz.
Orientation	OceanServer OS5000 Digital Compass	Determine the heading, roll and pitch of Starbug for navigational purposes.	3 x floating point values recorded at 20 Hz for roll pitch and heading.
Position	GlobalSat em-408 Locosys mc-1513 SanJose fv-m8	For pin pointing Starbugs location while surfaced.	Floating point values combined using an algorithm and recorded at 20 Hz as a single floating point number for both easting's and northing's.
IMU		Monitoring Starbug's acceleration	

IV. Scientific Data

A variety of scientific data are collected by Starbug. These data are used for assessing the health of the Derwent and Huon estuaries as part of CSIRO's TasMAN project. As a consequence of Starbug's ability to be configured quickly for different mission types (such as water quality surveys and benthic mapping), it is ideally suited to rapid data collection and analysis of new hypotheses related to various aquatic environments. Because Starbug is a mobile platform and carries different types of optical measurement equipment, it is well suited to benthic mapping applications. See Table II for a list of the scientific data collected by Starbug, the instrument which collects it, the reason for collecting this parameter and the format and frequency of the data.

TABLE II
SUMMARY OF RELEVANT SCIENTIFIC DATA BEING COLLECTED BY STARBUG.

Parameter	**Sensor**	**Application**	**Format and frequency**
Image	4 x Logitech USB web cameras	Benthic mapping, dead-reckoning	JPG files recorded between 1 to 5 Hz
Optical Emission Spectrometry	Ocean Optics Red Tide USB650, 350-1000 nm with 650 active pixels Ocean Optics USB2000, 200-1100 nm with 2048 active pixels	Measure the optical properties of the water and to measure the spectrographic signatures of the benthic environment	
Temperature, Turbidity, Conductivity, Dissolved Oxygen, pH	Turo-T613	Measuring various water quality parameters for the purposes of assessing waterway health	Floating point values recorded every 1.5 seconds
Fluorescence	Wet Labs ECO Fluorometer	Measure of active phytoplankton biomass and chlorophyll concentrations in water.	

V. Combining Engineering and Scientific Data

There are a few challenges in combining engineering and scientific data. The first problem is that of data synchronization. Starbug carries a variety of instruments, which usually have different logging regimes to each other. A single instrument may collect many parameters. For example the Turo T613 (see Table 2) instrument collects turbidity,

conductivity, ph, DOx and temperature. The data from some instruments can be logged as often as 20Hz while others may be much slower, such as once a second. This introduces the problem that each value from one parameter effectively refers to 20 values from another parameter. This issue could be handled by interpolation of the lower resolution data or averaging the higher resolution data. (This is not relevant here – but where we need to get to is an environment, where we don't have to interpolate, but we accept the data that we have, understand the fact that some is collected at 1 Hz and some is collected once a day – but we build models and visualizations that cope with mixed data sources) These approaches would change the raw data and potentially introduce errors due to data manipulation. For the purposes of accuracy it was decided to leave the data alone by default and instead try to match the closest values to each other during playback.

Another hurdle that this software solution must overcome is that of data conversion and the sheer volume of data that might need to be displayed. An AUV can collect large quantities of data over the course of a mission and this data is stored in binary format; a format which essentially requires conversion before it becomes useful for analysis. The reason the data is stored in a binary format is due to the compactness of this kind of data and the limited storage resources on an AUV. Once the data has been extracted from the AUV there is no real benefit in keeping it in binary format.

The sheer quantity of data is also a problem in that it can be quite challenging to display so much data to a user coherently. Figure 1 tries to illustrate the different types of data that Starbug collects, which need to be visualized, and just how varied the data can be, making visualization challenging.

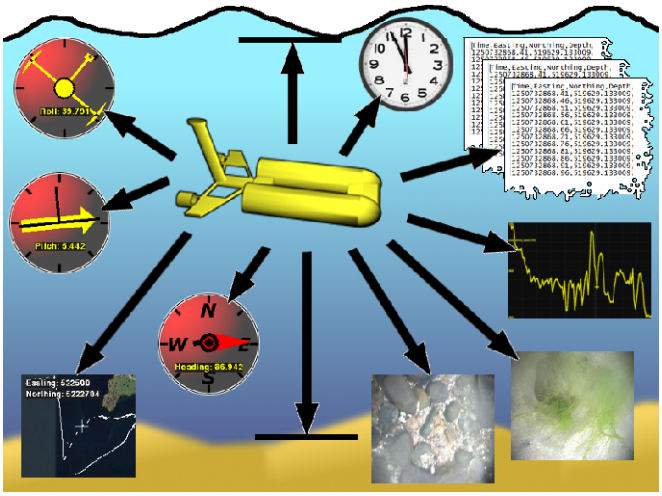

Figure 1. A visual representation of the diversified data collected by Starbug

Another problem is the visualization platform itself, there are many 3D visualization tools available however many of them require extra 3D graphics hardware that may not be available on a field laptop. The solution needs to be user friendly, easy to navigate and must effectively display both engineering data and scientific data so that the user can quickly cross reference data streams without lengthy interpretation. Various techniques were used to ensure that plotted data can be interpreted easily, such as time based zoom and arbitrary y-axis rescaling of datasets. The solution should be able to be run on field laptops so that if needed, a field officer could check data quickly to ensure that a mission went according to plan. The platform should be able to be reconfigured quickly so that data sets could be changed or plotted in different ways so that the user can optimize the view of the data, making interpretation easier.

Figure 2. CSIRO's AUV 'Starbug' during the diving operation. The tail section is visible above water, which is approximately 0.5m in height.

Figure 3. Typical Mission: AUV path traced in white from its mission start; bottom left, to end point; right center. The map shown covers an area of 3.76km horizontally and 1.84km vertically.

VI. Software Implementation

The software was developed on a Windows XP pro machine with Python 2.4 installed. The only external Python libraries required are Pygame and WxPython. The number of external library dependencies was kept to a minimum to reduce installation issues.

The software has also been tested at various stages during development in a Linux environment to ensure compatibility. Some of the main differences which impact on cross platform

operation are mainly associated with the underlying operating system. These differences include file name case sensitivity and directory path separators.

Pygame offered all of the extra requirements such as managing the display and creating a simple graphical user interface. The performance of new, average desktop computers eliminates the need to convert the images into a movie file format for playback. Images are instead loaded and displayed on the fly. Pygame is able to manipulate the images that are needed to be presented and do so at up to 30 frames per second on an average desktop machine or laptop. The images are logged in stereoscopic pairs, in vertical format. There are two sets of images to consider, from the forward looking and downward facing cameras. By using a preprocessing script, these images are cut, rearranged and merged so that the resulting image shows four images, forward left to right and bottom left to right. The resultant images are then optimized and reduced in size for the efficiency of playback.

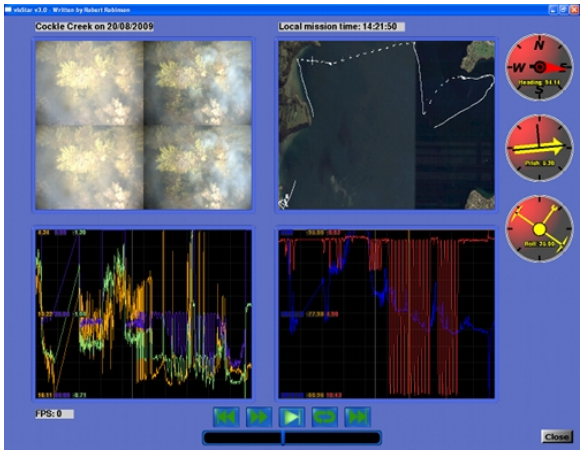

Figure 3. The interface showing all controls, indicators and display areas during playback. Images are being displayed in the top left, position is shown in the top right and time series graphs are in the bottom two windows.

By displaying the images on the fly as opposed to playing a pre-rendered animation, it was easier to implement the combination of the image animation with the mission timeline and simultaneously displaying other variables. Using a GUI plug-in or library was deliberately avoided for the visualiser because many GUI libraries utilize their own main loop and this would interfere with the very busy runtime of the main loop of the visualiser. The interface was designed in a simple style with images for control elements and a combination of Pygame's "Rect" objects and the associated "collide point" method to monitor mouse clicks on elements. This proved to be very effective in keeping the GUI code to a minimum and also has the added benefit of being easy to re-skin later if needed by changing image files and their associated coordinates.

Each display area is a custom object which can contain a variety of datasets. Each dataset is also a custom object which not only holds the time and value pairs of data, but other important information such as the colour of the plot and the metadata required to convert to and from the screen coordinates. The class structure is essentially hierarchical. A controlling class (Visualiser) manages all of the underlying elements, including the GUI and the configuration.

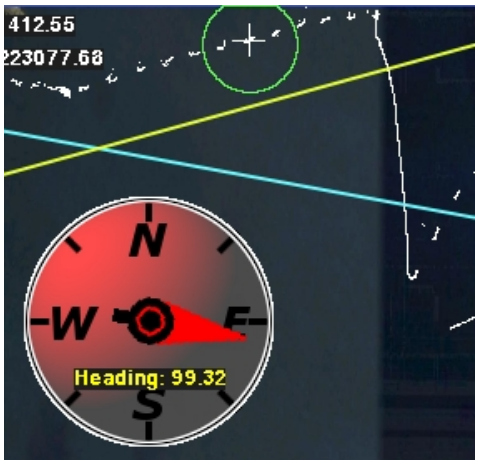

Figure 4. Example of inferred data with this tool. The green circle shows the AUV's position, the yellow line is parallel to the AUV's actual direction from GPS, the blue line is parallel to the AUV's heading from its onboard compass. This deviation was caused by southerly water currents in the mission area.

Once the program got to the stage where it was using a configuration file to define its behavior, it became apparent that a configuration editor would make using the visualiser easier. The Configurator was written using the WxPython library. WxPython is a Python wrapper around the popular WxWidgets library, commonly used in other languages such as C and C++ for GUI implementations. Using a GUI library for this section was considered far more appropriate as the Configurator would not need a busy main loop.

VII. Discussion

To be easily accessed programmatically, the logged data must be converted to a format that is easily read such as XML or CSV, in this case CSV was chosen. The DDX read program is used for this process and is currently available in the Linux build of the DDX framework. Once the log files are in CSV format, they can be read by the visualiser. Log files in CSV format will be referred to from here on as CSV log files. The CSV log files are expected to have a header line which offers a description of each field or column in the file. Fields include entries such as time, salinity or depth, among others. It is important to ensure that each CSV log file's first field is time.

It is also important that the time stamps are in Unix Epoch, or seconds since the Epoch format.

The first step in using the software effectively is to ensure that the config file (config.cfg) is correct and exists in the same directory as the visualiser program. The visualiser loads all its datasets from references in this file. A configuration manager program (called the Configurator) has been implemented to streamline and make this process less prone to user error.

The interface uses four display areas which can be individually configured, thanks to the configuration tool, to show images, time series data in graph form, or Global Positioning System (GPS) data. The interface also has GUI controls for playback functionality and timeline navigation, see Figure 4. Custom aircraft-style indicators were also created for other data types such as heading, roll and pitch making these variables easier to understand during mission playback. The speed and direction of playback can be adjusted as well as pausing and moving through the timeline frame by frame. Analysis aids have also been implemented. These aids include being able to rescale X and Y axes on any plotted time series data, allowing high density data to be visually analyzed more easily. Crosshairs show the current values during playback and actual text values are displayed when the visualiser is paused. An example of using the software to gain information from the mission playback can be seen in Figure 5. In this example during a WSW to ENE transect you can see the difference between actual direction (from the positional graph, GPS) and the compass indicator (from the AUV's internal compass). This indicates that Starbug was possibly driving against a reasonable northerly current when outside the influence of the bay. This particular conclusion demonstrates that despite the fact that local currents were not being monitored by the AUV, this information was incidentally recorded and discovered quickly by playing the data back in an interactive visual format. Trends or events like this tend to stand out when viewed in this way. Another example of this was found in two other sections of the same data set where it was noticed that Starbug's speed was slightly quicker and that it had an offset of about 90 degrees to its heading. It was deduced that Starbug had been loaded into a boat (lying transversely) and was then manually transported to a new location before being lowered back into the water. After reviewing the field notes, this is indeed exactly what had happened.

When using the interface, the ability to click anywhere on a graph and instantly move to that point in time was intuitive and required little explanation to users. Thanks to the vertical line on each time series plot which describes where in the time line the visualisation is at, many plots in different display areas could easily be cross-referenced. The start of an event on one plot can instantly be correlated to events in other plots. Because of these two features, traversing a mission log file and finding and analysing points of interest is quick and easy.

VIII. CONCLUSION

The software solution described here successfully visualizes and effectively displays many types of engineering and scientific data simultaneously, including images for benthic mapping, position from GPS and many time series data types thus improving the analysis of AUV missions. The media player style for the interface proved to be user friendly and simple to use, allowing the user to dive in quickly without much explanation and start getting results from the software quickly. The software made cross-referencing different data streams very fast and performed well on low specification hardware such as field laptops. The software examines CSV format data and still requires a more streamlined approach for converting the DDX based log files to CSV format. The software is able to quickly display very large data sets in a coherent way and the included analysis tools make examining highly compressed data much simpler. This tool has made the process of analyzing Starbug mission data much faster and simpler. This tool has also been successfully used to visualize mission data from an autonomous boat, proving that it can be used in conjunction with other autonomous machines.

ACKNOWLEDGMENT

The Tasmanian ICT Centre is jointly funded by the Australian Government through the Intelligent Island Program and CSIRO. The Intelligent Island Program is administered by the Tasmanian Department of Economic Development, Tourism and the Arts.

This research was conducted as part of the CSIRO Wealth from Oceans National Research Flagship.

REFERENCES

[1] Dunbabin, M, Roberts, J, Usher, K, Corke, P, "A New Robot for Environmental Monitoring on the Great Barrier Reef", national Conference on Robotics & Automation, pages. 1771–1776, April 2004.

[2] Davie A., de Souza P., Giugni S., Howell B., Hugo D., McCarthy P. and Timms G., "Tasmanian Marine Analysis Network (TasMAN Project)", Derwent Estuary Program Science & Management Symposium, Hobart, Tasmania. Nov. 2009.

[3] Maestro program: http://marsrover.nasa.gov/relatedsites/

[4] E. Nathan Harris a and George W. Morgenthaler, Planning, implementation and optimization of future space missions using an immersive visualization environment (IVE) machine, Acta Astronautica, Volume 55, Issue 1, July 2004, Pages 69-78.

[5] Conway, A., R., 1995. Autonomous control of an unstable model helicopter using carrier phase GPS only. Dissertation on the department of Electrical Engineering. Stanford University; and references therein.

[6] Blidberg, D. R., 2001, "The Development of Autonomous Underwater Vehicles (AUV); A Brief Summary", IEEE ICRA.

[7] Corke, P, Sikka, P, Roberts, J, Duff, E, "DDX: A distributed software architecture for robotic systems" Proc. Australian Conf. Robotics and Automation, 2004.