Dynamic AUV Mission and Data Visualization with Google Earth

Alhayat Ali Mekonnen, VIBOT student – Heriot Watt University, Universite de Girona, and Universite de Bourgogne

Mentors: Kanna Rajan
Summer 2009

Keywords: AI, Artificial Intelligence, Visualization, TREX, Google Earth, AUV

ABSTRACT

AUVs are currently being used for doing scientific observations of deep ocean water where it is difficult for humans to reach. As these vehicles are autonomous and moving underwater, it is difficult for operators and Engineers know what they are doing. This report describes an implementation of a dynamic AUV mission and data visualization tool that interacts with TREX, an intelligent AUV control SW developed at MBARI, and uses Google Earth to show observed data in a geospatial context to aid the understanding and analysis of scientists and to help AUV programmers understand decisions and activities taken by the autonomous agent.

INTRODUCTION

The ocean plays a fundamental role in our planet’s ecosystem and yet it has proven to be difficult to study. Visibility issues, pressure, and inaccessibility have made ocean study difficult for humans. Throughout the years, scientists and Engineers have come up with different inventions to study the world’s oceans in an efficient way. One of these inventions is the Autonomous Underwater Vehicle (AUV). AUVs are untethered robots that travel in water. By adding different sensors on these robots, they can be made to collect required data about oceans, from shallow to deep depth ranges, in an autonomous way. Figure 1 shows MBARI’s AUV, Dorado, about to embark on a mission to map oceanfront in the Monterey Bay area.

Making an AUV intelligent enough to cope with an unstructured and dynamic environment like the ocean autonomously is a very complicated and challenging task. At MBARI, the Autonomous Systems group uses Artificial Intelligence to make an AUV perform required science missions in an autonomous way. The team has successfully developed and deployed T-REX, a constraint-based planning system, on board an AUV enabling the vehicle to respond to unseen conditions and observations in the ocean in an appropriate manner for the required science mission.
T-REX, short for Teleo Reactive Executive, is a goal-oriented system that controls and drives the vehicle breaking down high-level science missions into low-level time interleaved control for an AUV based on constraint-based reasoning using EUROPA, a proven planning system developed by NASA. The system encapsulates the long-standing notion of sense-deliberate-act cycle at the heart of a control loop and reflects the goal-oriented nature of control and provision of response to exogenous events [Conor et al 2008]. Figure 2 shows a conceptual view of T-REX with 4 Sense-deliberate-act cycles (reactors). All these make T-REX a very complex software and difficult to debug. T-REX is run under several simulation schemes before being deployed on a real mission out on the sea. The way things work right now, T-REX is run and the programmer has to keep an eye on the log files to see what is happening and how things are going. That means, understanding all the text files being displayed which is difficult for all but the programmer. Also on some missions, the vehicle is made to make a survey then send the data to shore and wait for further commands. The scientist on shore analyses the data and sends a command back to the vehicle telling it what to do next. All these entail a visual interface with T-REX that is capable of showing what is happening inside the vehicle, real or simulated, and the data being collected in a geospatial context. The work discussed in this document is an attempt to address this need for an AUV data and mission visualization tool.
This report first discusses the objectives of the work, then followed by a discussion of the implemented system. Conclusion and recommendations follow concluding the document.

OBJECTIVES

At MBARI, AUVs are utilized to aid scientists to study the ocean in different aspects, seafloor mapping, water sampling, oceanfront mapping, INL surveying [Conor et al, 2009] to name a few. In all of these it would be very nice to have a visualization tool that would show the data of interest in a geospatial context. Also to the Software Engineer programming the AUVs, it would be very interesting to see what is happening inside the vehicle for problem understanding and debugging purposes. So the main goal, in the long run, would be to develop a visualization tool that would show the observations made by the vehicle and also depict what is happening inside the vehicle in a geospatial context with the possibility of having a two-way interaction.

As a first step towards this goal, the objectives during my internship were:

- To choose a suitable 3rd party geographical visualization software.
- To try and embed the selected geographical visualization software to show AUV observed data in a geospatial context.
- To try and interface the selected geographical visualization software with T-REX.

The work that was carried out to fulfill the above outlined objective is discussed in this report.

REQUIREMENTS AND PRELIMINARY ANALYSIS

The needs for visualization are clear. The AUV running TREX is autonomous and thus shares the main challenge with autonomous systems in general, we do not know how they are going to do the mission they are given. Added with this fact, the AUV is moving underwater and one cannot see its actions. Hence the requirements for a visualizer would be such that it would:

- Depict what the vehicle is doing in detail, position, behavior, decision, goals and intentions for the Software Engineer thus by easing understanding and debugging.
- Show the data being observed in a geo-spatial context for the scientist for better understanding and analysis.
- Be easily perceivable by all audience.

With the above set requirements, the list of suitable 3rd party geographical visualization software to use was narrowed down to two namely: COVE and Google Earth. The main reason for this is that these were the only softwares that show ocean bathymetry and thus can be used for accurate localization for the AUV in latitude, longitude, and depth. COVE supports python scripts for serious data manipulation and can also read netcdf data files. Google Earth on the other hand supports a markup language, Keyhole Markup Language (KML)-an XML based language for expressing geographical annotation and
visualization, and it also has a browser plug-in that can be manipulated with JavaScript api calls.

After careful evaluation, though COVE seems to be very promising, Google Earth was chosen at this point. Currently there is no documentation available for COVE and it is unstable, as it crashes some times. The only available means for understanding features and guides for manipulating it for the purpose at hand was by direct communication with Keith Grouchow, the person who developed it. On the contrary, Google Earth is stable, has abundant documentation along with sample codes and various forums are available. Hence all visualization work was carried using Google Earth with both kml and api calls.

IMPLEMENTATION

In the following sections a description of the implementation along with the necessary user guide to use the implemented system is discussed.

Off-board Visualization

The implemented off-board visualization tool system is capable of parsing through the log files generated by T-REX and showing the required data in multiple forms in Google Earth. The communication with Google Earth is done through a kml file. A Matlab script does the required task and generates the appropriate kml file, which is then loaded by Google Earth. Figure 3 shows the GUI of this implementation.

![Matlab GUI of the off-board visualizer](image)

This implementation has four options. ‘Show path’ shows the path of the AUV taken during the mission; ‘Show Behavior’ shows where each behaviors of the vehicle were active during the mission. The last two buttons are for data visualization. ‘Line Plot’ shows the actual data collected by the AUV for example temperature, conductivity, density, etc ‘Surface Plot’ makes an attempt to show a surface profile of the required
data, interpolated between min and max depth range. The surface plot generated is not smooth and requires further work either by increasing the number of interpolated data points or using extra features that might be available for Google Earth in the future. Figure 4 shows sample visualizations of temperature observations.

![Sample Temperature visualization along AUV path. Data taken from July 1st, 2009 ocean front following mission.](image)

**Online Visualization**

The Online Visualizer interacts directly with TREX to get the status of the vehicle and what is going on inside TREX. On a high level, a deprecated TREX component was modified so that it publishes the status of the vehicle and any other component of interest though a TCP port. On the other end, the visualizer connects to TREX through the web and shows what is happening in Google Earth. The html page contains two instance of the Google Earth plugin. One of them shows a global view of what is happening where as the second shows a first person view of the moving AUV. A java applet client is used on the visualizer to interact with Vitre, the modified component of TREX, reading the transmitted data and passing them to a javascript script that calls Google Earth apis to depict the required visualizations. Figure 5 shows a block diagram of the implemented system and Figure 6 shows how the page looks when running a simulated mission. The implemented system currently works with the simulator as there is no communication between the vehicle and base station when it is underwater.
Figure 5 – High-level block diagram of the implemented visualizer with TREX

Figure 6 – A snapshot of the Online visualizer showing the status of a path survey simulated mission. This mission is part of the front following mission.
USER GUIDE

The offboard visualization tool is fully developed using Matlab. Table 1 below gives a description of all the m-files necessary for its proper execution and a brief description of each.

<table>
<thead>
<tr>
<th>M-file name</th>
<th>Input</th>
<th>Output</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main_GUI.m</td>
<td>TREX log directory (String)</td>
<td>None</td>
<td>Main script that creates the GUI.</td>
</tr>
<tr>
<td>plotNav.m</td>
<td>TREX log directory (String)</td>
<td>A structure with all the data properly organized.</td>
<td>Extracts relevant data from Trex log files and returns them in a structured data type.</td>
</tr>
<tr>
<td>processMission.m</td>
<td>- Data structure return by plotNav</td>
<td>A data structure containing 2D interpolated data of the selected data.</td>
<td>Generates a 2D interpolated representation for vertical profiling of the selected data.</td>
</tr>
<tr>
<td></td>
<td>- String which data to process.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Resolution of the interpolation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>genLinePlot.m</td>
<td>- Data structure returned by processMission.m</td>
<td>None</td>
<td>Generates a Google Earth kml file for displaying a given data in a geospatial context with a line.</td>
</tr>
<tr>
<td></td>
<td>- File name for the generated kml</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Title of the path on Google Earth</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Description to be displayed on Google Earth ‘places panel’.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>behavior_processing.m</td>
<td>- File name for the generated kml</td>
<td>None</td>
<td>Generates a Google Earth kml file for displaying rectangular polygon showing the start and end of the active behaviors</td>
</tr>
<tr>
<td>genSurfPlot.m</td>
<td>- Data structure returned by processMission.m</td>
<td>None</td>
<td>Generates a Google Earth kml file for displaying a 2D vertical section of a data in a geospatial context with a line.</td>
</tr>
<tr>
<td></td>
<td>- File name for the generated kml</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Title of the path on Google Earth</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Description to be displayed on Google Earth ‘places panel’.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NorEast2latlong.m</td>
<td>Northing</td>
<td>[latitude; longitude]</td>
<td>Converts a UTM locations measurement in northing and easting, into a geographic location measurement with lat and long.</td>
</tr>
<tr>
<td></td>
<td>Easting</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Southern hemisphere? (True/false)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>UTM zone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WriteScript2.m</td>
<td>Coordinates [lat, long, depth]</td>
<td>None</td>
<td>Writes a section of kml file for making a line with specified style</td>
</tr>
<tr>
<td></td>
<td>Hexadecimal color string specifying color type [bbgrr]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kml file name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unique serial number to identify the style</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1 Description of all Matlab scripts making up the off-board visualizer.
Other files necessary for proper execution are:
- jetplus.mat
- main_GUI.fig

Prototype of the above described scripts:

```matlab
function varargout = main_GUI(varargin)
function plan = plotNav(path)
function gr = processMission(nav, fieldName, resolution)
function genLinePlot(gr, file_name, title, description)
function behavior_processing(file_name)
function genSurfPlot(gr, kmlFileName, title, description)
function lat_long = NorEast2latlong(northing, easting, southern, zone)
function WriteScript2(Coordinate, color_type, file_name, SnNumber)
```

To run the GUI, on the Matlab command prompt do
```matlab
>> main_GUI ‘TREX_LOG_DIR/latest’
```
Before running this visualizer, the post-processing script ‘volGet.sh’ should have been executed. One should also make sure that the path containing these scripts has been added to Matlab’s path.

The Online visualizer consists of two main developments and one modification. The modification made to Vitre involved modifying VitreReactor.cc and VitreReactor.hh. Each time the notify() method of this reactor is called, it sends the incoming observation in string form through the established socket and waits for an acknowledgement. Since all the new implementation went into this method, execution of TREX is paused until the visualizer connects to it.

On the other end, the component that reads the data sent from Vitre is implemented as one class extending the JApplet class. It contains necessary attributes and methods to establish connection and read data from the socket. The implemented applet is packaged as a jar file and signed by a certificate generated. The certificate generated and is valid for six months only. It has to be resigned after this period has passed for proper functioning. Signing the applet is the only means to ask and get proper permission on a client machine wanting to load the html for visualization. The prototype of the implemented applet class is as shown below.

```java
public class ClientDatastreamer extends JApplet {
    private String serverAddress; // Server IP Address
    private int port_number; // Port for comm
    private Socket socket;
    private BufferedReader input;
    public DataOutputStream output;
```
public void init( );
public int setPortNumber(int);
public void setServerAddress(String);
public int connect2Server();
public String readServer();
public void sendAcknowldege();

The javascript component is responsible for calling the java applet methods, parse the incoming messages, and direct them to the Google Earth plug-in, with an api call, appropriately for the required depiction. The Google Earth plug-in requires a map api key that is generated at http://code.google.com/apis/maps/signup.html. This key is unique and depends on the URL of the page containing the plug-in. Hence it should always be generated if a different URL is going to be used for the page. Replacing the key, shown in green below, in the first <script> tag of the index.html would suffice

<script
src="http://www.google.com/jsapi?key=ABQIAAAAWGo6skSoVfEpHThmXWyOXRTeZU8Kbqlby81LeA1UL4a3BuJohQi3vskJBKCwIAjSRnGy_YQLQJ_Sw" type="text/javascript">
</script>

The files that are required for the html to run properly are:
- index.html
- ClientDataStream.jar

The html page should be hosted on a computer that can act as a webserver and should be pointed so as to serve index.html.

To run a mission with the visualizer:
1. Make sure the amc.cfg file of TREX contains Vitre reactor

```xml
<Config>
<!-- Use this component when running the sudo sim -->
<TeleoReactor log="1" name="sim" component="DeliberativeReactor"
lookAhead="1" latency="0" solverConfig="sim.solver.cfg"/>

<TeleoReactor name="exec" component="DeliberativeReactor"
lookAhead="5" latency="1" solverConfig="exec.solver.cfg"/>

<!-- Skipper has 10 Hour lookahead. -->
<TeleoReactor name="skipper" component="DeliberativeReactor"
lookAhead="36000" latency="60" solverConfig="skipper.solver.cfg"/>
<TeleoReactor name="webviz" component="Vitre" lookAhead="0"
latency="0" config="webviz.cfg"/>
</Config>
```

<Config host="localhost" port="32000">
  <External name="missionGoals"/>
  <External name="vehicleState"/>
  <External name="waypoint"/>
  <External name="navCommandState"/>
  <External name="setpoint"/>
  <External name="descend"/>
  <External name="waypoint_yoyo"/>
  <External name="getgps"/>
  <External name="FireTheGulperAMC"/>
  <External name="vehicleActivity"/>
</Config>

3. Run TREX

crd2007 $ ./amc_o_rt <mission-name>.cfg -sim

4. Open an Internet browser (preferably Firefox). Type the URL where the index.html file is being hosted. When the page loads, it will ask for confirmation to allow the applet to run and should be confirmed. If the Google Earth key is correctly set then two instances of Google Earth plug-in will appear in the page.

5. Click on Start Mission; input the port number and IP of the computer running TREX. The port number should be the same as that provided in the webviz.cfg file. Now the visualization should start and show the status of the vehicle and mission

CONCLUSIONS AND RECOMMENDATIONS

To conclude, a dynamic AUV mission and data visualization tool has been developed. This tool has two components: an off-board Visualizer that parses the log files generated by TREX and visualizes data and vehicle information in Google Earth, and an online visualizer that directly communicates with TREX and shows the trajectory, AUV mission, and currently active vehicle behavior. The visualizer is of great help to the SW Engineer and Scientist, showing vehicle status and data in a geospatial context.

As a recommendation, there are further modifications that can be made to the visualizer. These include first improving the data depiction in Google Earth so that the scientist can make all the analysis directly on it. Next it should also be enhanced so that it can show the plan TREX is generating and/or modifying to accomplish the mission. This would help get a very good understanding of the automated planning TREX is using. Furthermore, if a communication scheme can be established between the vehicle and shore when it is underwater, then the visualizer can be made to work with this showing Real Time data and vehicle status.
ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my mentor Kanna Rajan and Frederic Py who has been guiding me and helping me in doing what I was doing. Next in line I would like to express my appreciation for Thom Maughan, Angel Olaya, and Tom O’rielly for all the help and support they have given me. I would also like to thank John Ryan, Mike McCann, and Michael Godin for their time in listening to me and giving me suggestions. Last but not least I want to say thank you for George Matsumuto and Linda Kuhnz, for organizing this great internship program and their support, advice, and encouragement through out the summer.

References:
