Md. Emazuddin Alif

Model Based Assurance for Autonomous Vehicles

Mentor: Dr. Gabor Karsai, Institute for Software Integrated Systems, Vanderbilt University

1. General Problem and Context

Use of underwater pipelines is deemed one of the most convenient means for long-distance transport of oil and gas and power transmission, and likewise submarine cables are important for telecommunications. Although their use has vastly facilitated progress in their respective domains, they face risks from various sources, including damage from corrosion, tectonic activity, faulty materials, construction flaws, and damage from ships' anchors. There have been documented cases of mass fish mortality caused by natural gas leaks from underwater pipelines due to drilling accident. [1] In order to avoid such unfortunate situations, constant monitoring of the pipelines is required. Because round-the-clock human monitoring in such cases is too expensive, a comparatively cheaper and safe alternative is needed. Hence, researchers have examined the use of Unmanned Underwater Vehicles (UUV). [2]

UUVs provide an easier option for pipeline monitoring because of their autonomous capabilities. Remotely operated vehicles communicate with the master control using radio frequency (RF) signals. Water poses significant resistance to RF signals and have been shown to absorb signals upto 2.4GHz (standard wi-fi signal). [3] This makes autonomy of the vehicle a big advantage in monitoring the pipeline, as the vehicle's decision-making capabilities cut down the cost and margin of error that may have been caused by radio communications between ground and the remotely operated vehicle. Due to the vague nature of the policies around Connected

and Automated Vehicles (CAVs), the safety certification required for a fully functioning UUV falls short of the benchmark provided by other vehicular technologies. Rigorous testing and evaluation are required to ensure proper vehicle safety for UUVs. This can be conducted in two ways, using physical real-life experimentation and using various simulation platforms. Real-life experiments can provide very accurate data, but they are often too expensive (for example, the starting price of the UUV that can serve the necessary function costs fifty thousand dollars each). [4] In contrast, simulation platforms often approximate different real-life scenarios, but their impractical data are not always practical. For proper analysis, a balance between these two methods must be reached.

2. Description of specific human cyber-physical systems problem

Since CAVs operate in a highly uncertain environment, and algorithms cannot be designed for every situation, researchers have started using software components built with machine learning techniques. [5] Here data is collected from a simulated environment and a general system (typically in the form of a neural net). Then the system is trained to perform various functions using the said techniques. Since training data does not cover every possible scenario, it is a challenge to properly build a case to assure safety of the autonomous vehicles. In case of UUVs human interaction is minimal, and it is comparatively easier to design a system and argue for its safety than for other types of automated vehicles. For our project, the UUV is intended to track and follow the pipeline in the seabed while maintaining a specified distance and avoiding obstacles. The human cyber-physical system problem addressed in this project was to graphically

show the safety of the software system through logic-based argument maps called Goal Structured Notations (GSN). [6]

3. Challenges reaching a functional system

UUVs usually use a combination of sonar and thermal imaging technologies to sense the pipelines on the seabed. [7] The sonar tracks the elevation of the seabed, and it senses the pipe as a line. It deploys the line-following algorithm built in its software system to follow the pipe while keeping a distance. As the UUV follows the pipe, it needs to avoid obstacles like rocks, shoals of fish, and other marine creatures. During its journey, the UUV needs to adjust its trajectory according to pipe bend, obstacles, ocean current, water temperature, and many other environmental factors. When the seabed is very uneven and the pipe is covered by sand, the UUV often loses track of the pipe. On such instances the UUV keeps moving along the last recorded vector. If the pipe bends or changes direction significantly while covered by sand, the UUV loses the track of the pipe permanently. If the UUV cannot find the pipe after losing its track in thirty seconds, it comes back to the point where it lost the pipe and starts looking for it in increasingly larger circles. A new algorithm is in the works right now to add a timeout (N=30s) for the UUV.

A part of the student project involved simulating the UUV in the simulation platform Gazebo and testing out one of its pipe-following algorithms. Gazebo can be described as a 3D simulator which can rapidly and accurately test algorithms and design robots in a realistic (outdoor and indoor) environment. [8] One of the biggest advantages of Gazebo is it can be easily integrated with Robotic Operating System (ROS). ROS is a collection of libraries, drivers and tools that are used for effective development of robot systems. The Gazebo architecture consists of

two executables – the *gzserver* and *gzclient*. *Gzserver* is the core of Gazebo and can run independently from *gzclient*. *Gzclient* is a graphical user interface that visualizes the model running on *gzserver* and provides some controls to actuate over the model. It cannot run independent of *gzserver*. Gazebo boasts a robust physics library with the following physics engines: Open Dynamics Engine (ODE), Bullet, Simbody, and Dynamic Animation and Robotics Toolkit (DART). The physics library allows the simulation to be realistic and the model to act coherently with the environment according to the laws of physics. Gazebo plugins are used to provide direct control over the models in simulation. They are complicated chunks of code written in C++ language. Gazebo's built-in model library did not have any shape resembling the UUV. Unfortunately, there were also no existing Gazebo plugins that could be used to provide complete autonomy to the simulated UUV, so a combination of the existing plugins was required to be modified to work together.

4. Technical Problem and Research Setting

Last summer I performed my research under Dr. Gabor Karsai at the Institute for Software Integrated Systems (ISIS) on simulating and evaluating UUVs and building assurance cases for them. A structured assurance case can be defined as a documented body of evidence that provides a convincing and valid argument that a specified set of critical claims regarding a system's properties are adequately justified for a given application in a given environment. [9] For example, if a system is designed that has three independent sensors and a single, highly reliable voting circuit that always picks the 'mid-value' from the three sensor readings, then the logical argument is that the system is highly reliable because (1) if one sensor fails and the other two

show matching values, the voting circuit's output is correct, and (2) the probability of two or three sensors failing exactly the same way is very low. An integrated toolchain was used for architectural modeling of the UUV with Learning Enable Components (LECs), LEC training and training data collection, LEC evaluation and verification, system software deployment, and modeling and analyzing safety cases. [10] It was used to train the LECs in the UUV to run in various ranges of temperature, ocean current, pipe bend, and other environmental variables.

The UUV selected for experimentation and evaluation was IVER3. IVER3 was 74-85 inches in length, 5.8 inches in diameter and 59-85 lbs. in weight. It had a speed rank from 1-4 knots (0.5 -2 m/s), 800 W-hrs of rechargeable Li-ion batteries, and endurance of 8-14 hours at 2.5 knots based on configurations. It had standard Ethernet and wireless 802.11n connection, 48V Servo DC motor with four control planes, Intel Dual-Core 1.6 GHz N2600 processor with MS Windows embedded, and up to 512 SSD for data storage.

For the student project, I wrote a ROS node that uses the existing Gazebo plugins to provide line-following capabilities to the model. A ROS node can be simply described as a program in ROS that allows the user to use the Gazebo plugins and control the robot in simulation. For example, to give a robot object avoidance capability, a ROS node can be written which will subscribe to messages from the depth sensor of the robot and give it velocity commands to move a specific distance away from the obstacle. A ROS node consists of two files, a launch file that launches the node and a source file that contains code specifying the function of the node. The source file of the node can be written in C++ and Python. In this project, I used Python 2.7 to write the ROS node. Subsequently I integrated it with the model in Gazebo to subscribe to the messages from the camera and cliff sensors in the simulation and detect the

change in color and elevation of the ground. The ROS node then published velocity commands to the model to follow the line accordingly. This implementation of the ROS node was able to control successfully the model into following the line.

5. Future Research

Although the pipe-following algorithms were working correctly with UUV in both the simulation and real-life experiments, there is always more room for improvement. The sonar and the light sensor can be improved by using more sophisticated signal processing algorithms to generate better and clearer picture of the seabed. New search patterns and algorithms can also be implemented to the UUV to recover the location of the pipe after losing it. Hardware modifications can be made to the UUV to strengthen it in event of collision with obstacles. A big part of this project was to observe the behavior of the UUV in a congested environment and how it responds to other elements. More research can be done in an environment with more than one autonomous element. How they interact with each other and with other human controlled machines can be observed as well. Goal Structured Notation (GSN), the logic map used to build the assurance case for the UUV here can also be used for building similar safety cases of other automated vehicles. More robust certification of UUVs may lead to better regulations and policies, which would contribute to commercialization and research of automated vehicles as a whole.

References

1. Ellen Karm. Environment and Energy: The Baltic Sea Gas Pipeline, Journal of Baltic

Studies, 2008.

2. Button RW, Kamp J, Curtin TB, Dryden J. A Survey of Missions for Unmanned Undersea Vehicles, RAND National Defense Research Institute, 2008

3. Butler L. Underwater Radio Communication, Amateur Radio, 1987.

4. Kirk J. US Navy Purchases 4 Iver3 AUVs and Modernizes 3 Iver2 Systems, Autonomous Undersea Vehicle Application Center, 2014.

5. Gupta A. Machine Learning Algorithms in Autonomous Driving, Industrial IoT World, 2016.

6. Ge, X et al. Introducing Goal Structuring Notation to Explain Decisions in Clinical Practice, Procedia Technology, 2012.

7. Zhao, S. Automatic Underwater Multiple Objects Detection and Tracking Using Sonar Imaging, The University of Adelaide, 2014

8. Miro, JL. Control Simulation of a Line Tracker Vehicle Using Gazebo, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, 2017.

9. Scott AT, Krombolz AH. Structured Assurance Cases: Three Common Standards, High-Assurance Systems Engineering, 2005.

10. Hartsell C, Mahadevan N, Ramakrishna S, Dubey A, Bapty T. Model-Based Design for CPS with Learning Enabled Components, DESTION '19, 2019.