Intelligent Mission Control of Robotic Underwater Vehicles

Bradley Null*, Tim Josserand and Justin Romero
Applied Research Laboratories
The University of Texas at Austin
*Contact: bradnull@arlut.utexas.edu

***Abstract*—Deployment of robotic underwater vehicles (RUVs) in unpredictable underwater environments necessitates the development of an intelligent mission controller (IMC) that would facilitate autonomous navigation of predetermined waypoints while identifying objects, avoiding obstructions and building a model of what has been sensed in real time. The IMC should balance a preference for following a previously determined mission path and dynamically re-planning a mission in order to avoid obstacles and minimize pre-planned path deviation. This paper presents an IMC that combines an efficient three-dimensional world model representation with a frustum culling trajectory search method to create an intelligent, adaptive path planning algorithm balancing safe obstacle avoidance with minimal pre-planned path deviation. The IMC is tested in a real-world environment and results are discussed. Future improvements of this IMC will also factor in the ability to identify regions of interest off a mission path and explore them as necessary.**

***Index Terms*—Autonomous underwater vehicles, autonomous control, obstacle avoidance, path planning**

I. Problem Definition

Creation of an autonomous RUV for underwater exploration of regions with little or no *a priori* knowledge presents a number of difficulties for an intelligent mission controller. One problem is creating adaptive behaviors based on sensor input. The amount of information gathered from sensor inputs correlates to the accuracy of control. A large amount of information creates a challenge in building a generalized framework. Thus, the IMC should be scalable with the amount of information present. Another problem is that the system must be completely self-contained due to the nature of the environment (i.e. communications and external control should be avoided because the RUV is intentionally deployed away from personnel and will likely be out of communication range). The final problem is the ability to understand and complete objectives through the IMC. Prioritizing a balance between self-preservation and objective completion is necessary and should be incorporated.

Currently pre-programmed missions for the vehicle discussed in section III are limited to simple "mow-the-lawn" trajectories at altitudes above the bottom sufficient to provide side scan sensors with appropriate declination for image formation. In this case, mission objectives can only be achieved in fairly benign environments characterized by flat or gently sloped seabed topography. The objective of the work presented in this paper is to create an autonomous vehicle that can properly handle ingress and egress of harbors in more diverse environments. The IMC presented here is a step towards this objective by implementing advanced obstacle avoidance and path planning while maintaining a knowledge of mission directives. The system must be able to handle navigation and exploration through many classifications of objects such as single large obstacles, dock pilings, walls, box canyons, and cliffs. The IMC must consider all of these constraints when applying alterations to the pre-determined mission path.

II. Background

There have been many implementations of targeted and generic autonomous control systems for underwater vehicles. A survey of 12 different systems can be found in [7]. Some have succeeded in using reactive behaviors such as the artificial potential fields utilized in [11]. There have also been efforts for adaptive reactive controlling as in [3]. Reactive or behavior based control is useful in RUVs that do not maintain a reference to the explored environment. This is usually a symptom of the information being either not desired or too large to maintain such as in [5]. However, for systems that are required to keep track of the sensed environment, it seems preferable to utilize an adaptive rule schema. [10] contains the origin of STRIPS, a rule based control schema. This is the foundation that some of the work presented in this text is based on. Extending the control to artificial intelligence, some have tried using rule based systems such as UWL [6] or PROLOG [8]. These rule based controllers can use logic programming for decision making processes. Others have expanded the BDI RUV frame work for intelligent control as in [9].

The primary objective of the research presented in this paper establishes a mission controller for the purpose of performing obstacle avoidance similar to [2]. Obstacle avoidance is necessary not only to preserve vehicle integrity, but as [4] describes, to promote mission longevity by preventing the vehicle from getting stuck in local minima while avoiding obstacles. This requires that the vehicle be self-aware enough to make path trajectory decision that prevent the vehicle from getting in harms way while still remaining in the mission area. The work presented here focuses on the ability to intersect sensor data, process the data and utilize it to maintain vehicle safety that does not supersede mission goals.

978-1-4673-0831-1/12/$31.00 ©2012 IEEE

III. System Description

The system that is used for all IMC design considerations is the Hydroid REMUS underwater vehicle shown in figure 1. This vehicle has a self-contained PC104 that is running the current version (as of official testing) of the REMUS software. This software handles all low-level vehicle controls and provides a remote control (RECON) interface.

Fig. 1. REMUS underwater vehicle with obstacle avoidance sonar mounted on the front.

Attached to front of the vehicle is a BlueView sonar nosecone that contains an onboard computer running Windows XP. This sonar provides four different arrays. Two of the arrays are oriented vertically and the other two are oriented horizontally yielding a total visual field of 45 degrees by 45 degrees in front of the vehicle. The onboard computer provides a means to run self-contained software for interfacing with the REMUS RECON. The system is provided 28 volts of power through an onboard bank of batteries. All of this hardware must also not exceed a length of two meters and a diameter of 20 centimeters.

IV. Control Schema

The control for this IMC can be seen in figure 2. The sensor framework layer reads and interprets all the information from the vehicle sensors and passes the unprocessed data to the sonar processor. The sonar processor then rectifies the sonar data given by the sensor framework into a format that can then be passed to the mission processor layer. The mission processor is where all of the dynamic control through data interpretation occurs.

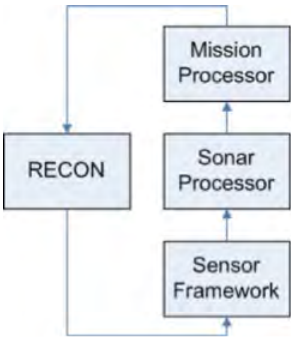

Fig. 2. Control flow for the autonomous IMC.

The essential low level path following algorithms are contained in the RECON software and not of concern to the mission processor layer. The mission processor will subsume the RECON software only when it has determined that a region of interest (ROI) has been identified. This trigger will give complete control of the vehicle to the mission processor for high level path planning.

V. World Model Representation

To maintain a world model for the mission processor layer to reference, a sparse occupancy grid is utilized. By dividing the explored regions into discrete cells in a three-dimensional grid, also known as a voxel grid, redundant data storage is eliminated. This is also beneficial because the resolution of the discretization can be bound properly for memory management purposes. To populate the world model, each region in the grid can be classified as occupied or empty. If a data point from the sonar processor falls within a specific voxel, then the voxel becomes classified as occupied. Otherwise, all voxels are empty at instantiation. The occupancy grid structure creates enough knowledge to derive a safe path with respect to the ROI.

In the case of the voxel grid, we store each item in the occupancy grid as an axis-aligned bounding box. This is a common datatype in three-dimensional computations as the axis-aligned bounding boxes allow for fast computation of intersection. Bounding boxes are used in computer graphics in order to determine proper collision detection between graphical entities.

To define the trajectory of the proper path, a method known as frustum culling is used. An excellent tutorial of implementing view frustum culling can be found in [1]. The frustum culling method casts frustums, a three-dimensional viewing box, from an origin position to a goal position. Each point of intersection between the frustum and an axis-aligned bounding box is recorded. In the case of a voxel grid, the only concern would be if the frustum intersected a voxel that was classified as occupied. By casting a frustum from the vehicle's current position to the desired ROI plan's position, the voxel grid representation provides a means to know if the trajectory would intersect an occupied region. In the case that a direct trajectory could not be found, the plan would be modified to only travel through voxels that are classified as empty. The initial directions for the search frustums are shown in figure 3. The frustum culling method begins by searching these regions for occupied voxels and iteratively searches radially outward with specific input bounds. If the search reaches the bounds, the current objective is aborted.

Fig. 3. Initial frustum culling search pattern to find a new trajectory.

VI. Mission Processing

The goals of the mission processor for our REMUS vehicle focused on preserving vehicle health while maintaining the pre-programmed mission objectives. A diagram of the three major goals can be seen in figure 4. The Obstacle Avoidance maneuver is intended to divert the vehicle off the current trajectory in order to circumvent excessive encroachment on a potentially threatening obstacle. The Next-Leg maneuver should occur when the frustum search pattern has failed to identify a safe path on the current mission leg, but can still continue onto other legs of the mission. The Mission Abort maneuver should occur when there is no unobstructed path available.

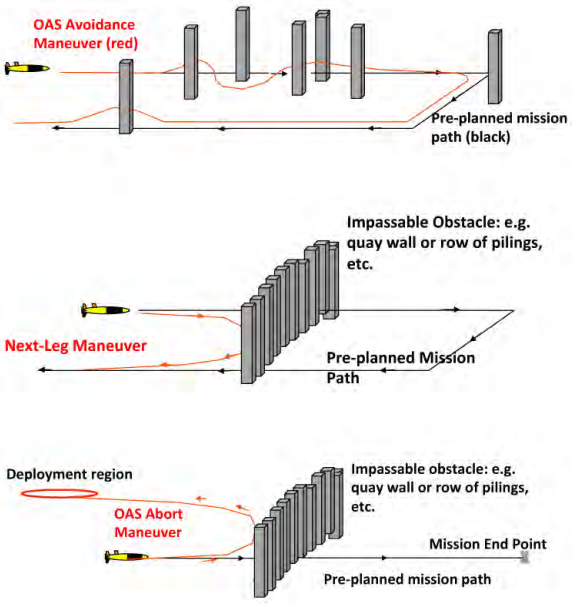

Fig. 4. Mission controller vehicle path modification maneuvers. (Top) Obstacle Avoidance Maneuver. Obstacle avoidance by diverting off-path. (Middle) Mission Leg Abort Maneuver. Obstacle avoidance by skipping pre-planned mission legs. (Bottom) Mission Abort Maneuver. Maneuver to safely abort mission when no path can be found.

Once the mission processor layer has subsumed control using the RECON interface, the IMC will begin to interpret the data by following the state machine shown in figure 5. The data is first given to an identification algorithm. The current implementation of the of obstacle identification algorithm searches for different classifications of obstacles including single independently floating obstacles, walls, dock pilings, box canyons, and cliffs. After the ROI has been identified, a plan can be devised to either navigate around the obstacle or explore the ROI as described in section V using the frustum search algorithms.

If for any reason a path cannot be derived, the state machine will enter an abort sequence. This will return the vehicle to a known safe position and allow for its retrieval. By implementing the state machine, the mission processor layer has the control required to dynamically alter a mission based on the sonar input.

To accomplish the mission processor logic, an adaptive planning system is implemented in for the form a push-down automaton. A pushdown automaton maintains a stack of the current states and executes the rules for the state that is at the top of the stack. Everything on the stack is a valid state which has a specific operation and a transition function to a next state. The state machine adapts to specific situations via the transition function between states. The adaptive automaton is shown in figure 5. In the case of the REMUS IMC, the goal of the state machine was to keep the "Mission" state at the top, and begin pushing down obstacle avoidance states as necessary.

VII. Results

The IMC was implemented and tested on REMUS vehicles in the Applied Research Laboratories' Lake Travis Test Station. Test targets were deployed at known locations and various missions were devised to test the behaviors of the IMC. The IMC was able to successfully execute the three maneuvers shown in figure 4. Tests were set up as shown in figure 6. The test shown on the top of figure 6 placed targets directly in the vehicle's path in order to test coordinated obstacle avoidance. The second test, shown on the bottom of figure 6 intentionally attempted to beach the vehicle in order to allow the shore to act as a wall for testing the skipping of mission legs and abort maneuvers.

Upon mission review, it was apparent that the vehicle was able to perform coordinated avoidance when possible, skip mission legs when unavoidable obstacle were present and abort a mission if an unavoidable obstacle blocked the path to all legs of a mission. In the future, testing can be performed to better quantify the success/failure rate of the IMC in various environments as well as perform more advanced avoidance maneuvers.

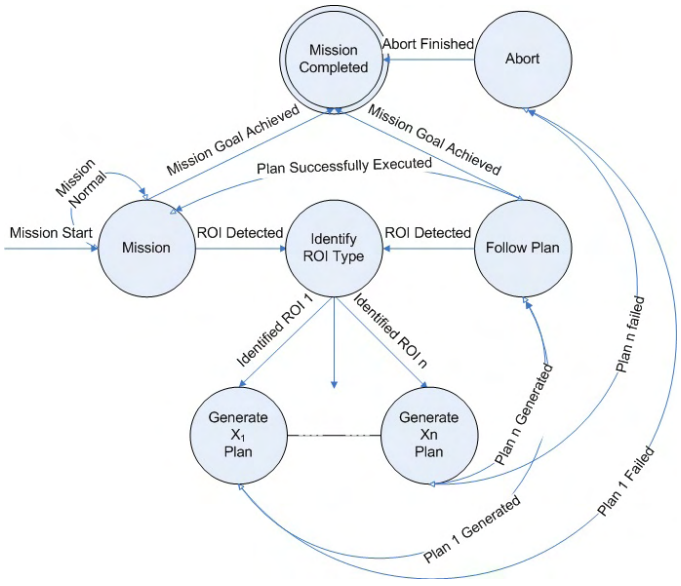

Fig. 5. State diagram of the Mission Processor layer.

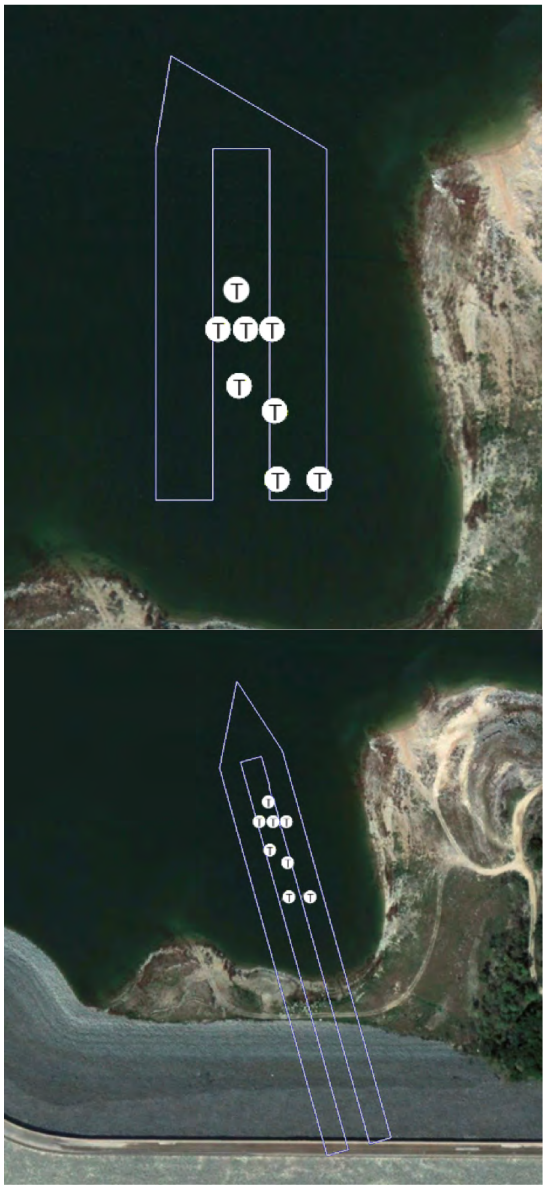

Fig. 6. Test missions run at in Lake Travis to validate proper IMC behaviors. (Top) A mission path, in white, through several test targets marked with T. This mission demonstrates the IMC's ability to avoid multiple obstacles. (Bottom) A mission path onto the shore that forces the IMC to skip mission legs and abort the mission.

VIII. Future Work

The current implementation of the IMC has been completed in C++. This language interfaces well with the low-level processors of the RECON software and still provides enough high level abstraction to implement the mission processor layer. However, in the future it may be desired to move the mission processor away from an object oriented language for a rule-based language that better suits the desired automaton. The area of logic programming is rapidly developing and new satisfiability solvers and declarative languages provide high level mission control capabilities. However, there are many problems with keeping the current system real-time. Many logic solvers can take exponential time to perform resolution refutation making them difficult to adapt to embedded real-time systems. Further exploration into this issue is necessary.

As was stated previously, the quantity and quality of data impacts the effectiveness of the mission processor. In the future it would be desired to install new equipment that would allow for more sonar data to be acquired from the environment. A more accurate controller should entail more accurate sonar systems. However, it would be necessary to explore if the opposite case is accurate. If there is limited data, will the controller fail completely or can it overcome the limitations data through refinement of a probabilistic model? These questions and many others are yet to be explored as the IMC is further developed.

References

[1] Lighthouse 3D. View frustum culling. http://www.lighthouse3d.com/tutorials/view-frustum-culling/.

[2] G. Antonelli, S. Chiaverini, R. Finotello, and R. Schiavon. Real-time path planning and obstacle avoidance for rais: an autonomous underwater vehicle. *Oceanic Engineering, IEEE Journal of*, 26(2):216 –227, apr 2001.

[3] G. Antonelli, S. Chiaverini, N. Sarkar, and M. West. Adaptive control of an autonomous underwater vehicle: experimental results on odin. *Control Systems Technology, IEEE Transactions on*, 9(5):756 –765, sep 2001.

[4] P. Calado, R. Gomes, M.B. Nogueira, J. Cardoso, P. Teixeira, P.B. Sujit, and J.B. Sousa. Obstacle avoidance using echo sounder sonar. In *OCEANS, 2011 IEEE - Spain*, pages 1 –6, june 2011.

[5] C.C. Eriksen, T.J. Osse, R.D. Light, T. Wen, T.W. Lehman, P.L. Sabin, J.W. Ballard, and A.M. Chiodi. Seaglider: a long-range autonomous underwater vehicle for oceanographic research. *Oceanic Engineering, IEEE Journal of*, 26(4):424 –436, oct 2001.

[6] Oren Etzioni, Steve Hanks, Daniel Weld, Denise Draper, Neal Lesh, and Mike Williamson. An approach to planning with incomplete information. In *In Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 115–125. Morgan Kaufmann, 1992.

[7] W.D. Hall and M.B. Adams. Autonomous vehicle software taxonomy. In *Autonomous Underwater Vehicle Technology, 1992. AUV '92., Proceedings of the 1992 Symposium on*, pages 49 –64, jun 1992.

[8] A.J. Healey, D.B. Marco, and R.B. McGhee. Autonomous underwater vehicle control coordination using a tri-level hybrid software architecture. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 3, pages 2149 –2159 vol.3, apr 1996.

[9] David Morley and Karen Myers. The spark agent framework. *Autonomous Agents and Multiagent Systems, International Joint Conference on*, 2:714–721, 2004.

[10] Nils J. Nilsson and Richard E. Fikes. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, oct 1970.

[11] C.W. Warren. A technique for autonomous underwater vehicle route planning. *Oceanic Engineering, IEEE Journal of*, 15(3):199 –204, jul 1990.