Syllabus: Introduction to Adaptive and Collaborative Marine Autonomy with MOOS-IvP

Athens, June 2026

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

1 About This Course

This short intensive course provides a introduction to the emerging field of marine autonomy, using the MIT open-source MOOS-IvP software framework. Participants will learn the fundamentals of decision making for autonomous control of a single marine platform and teams of platforms. This course is conducted with examples and exercises in both simulation and in-water experiments using multiple small autonomous surface vehicles.. We will present variations of marine autonomy problems and missions often found on marine robots, and discuss why certain approaches to autonomy are more appropriate than others depending on the mission. The concept of robot middleware software is introduced along with basic and advanced methods for engaging with the software at run time through low-level injection and scoping methods as well as through a command-and-control (C2) interface. Decision-making with a behavior-based architecture is presented, along with methods for model predictive action selection for behavior reconciliation, including the use of multi-objective optimization used in the MIT software stack. Students will then be introduced to methods for constructing autonomous capabilities using the behavior building blocks for switching modes based on mission milestones, sensed events or C2 messages. Finally, collaborative multi-vehicle autonomy is introduced through the core topics of inter-vehicle messaging, decentralized decision-making and the use of multi-vehicle protocol-based autonomy. The class will include instruction and labs to guide the construction of new applications and applying an application to an autonomy mission. The culmination of the class will be three days of in-water deployments comprised of head-to-head competitions on ASVs running student-generated algorithms.

2 Learning Objectives

2.1 High-level Learning Objectives

Understand:

- `Core Decision Layers`: The difference between: Planning, Autonomy and Control.
- `Collaboration paradigms`: The types of multi-vehicle collaboration: co-fielded, protocol-based, consensus-based, and the assumptions about inter-vehicle messaging in each.
- `Core architectures`: (a) field control, (b) robot middleware, (c) behavior-based decision-making, (d) front-seat/backseat or payload architecture.
- `Full life-cycle mission phases`: The major phases of marine robot operation: (a) mission planning, (b) mission launching, (c) mission operation, (d) post-mission analysis, and the tools associated with each.

- Mission structures: The difference between time-linear or sequentially scripted missions and open-ended mode based autonomy missions.
- Robot Field Operations: The full preparation and operation of an un-crewed marine vehicle in the outdoor environment, including vehicle recovery and data archiving and post-mission analysis.
- Automated Monte Carlo Mission Simulations: Construction of missions with performance metric utilities runable in headless unsupervised mode Monte Carlo tests.

2.2 Component Level Learning Objectives

Understand:

- Primary robot system components: Be able to identify and understand the jobs and roles of core software modules comprising an autonomous marine vehicle including, external sensing, sensor fusion, on-board health sensing, path planning, autonomy, control, C2 communications, inter-vehicle communications
- Robot middleware: the construction of complex software systems through asynchronous distinct software modules. Inter-process communication, serialization and deserialization of messages.
- Autonomy software community: Understand how a MOOS community, a MOOSDB and set of connected MOOS applications, comprises the distinct components of an autonomous marine vehicle software stack.
- Simulation: Understand (a) the difference between the rendering of vehicles and simulation of physical motion, (b) which modules replace a simulator on a physical vehicles, (c) how to launch multiple simulated vehicles on a single computer, over a collection of computers if running a distributed simulation.
- Data injecting and scoping: Understand (a) the software tools and options for injecting (poking or writing) data into a live software community, (b) scoping (reading or visualizing) data values and changes in a live community.
- Scripting: Understand how shell scripts automate many components of the launch pipeline, especially in multi-vehicle simulations.
- Basic C2 (command and control): Learn how to interact with a deployed vehicle or vehicles through a GUI-based module, and how to configure C2 to suit a new set of mission objectives.
- Mission planning and launching: Understand the core parts of mission specification, launch-time mission file configuration, and the role of shell scripting in complex mission launches.
- Behavior-Architectures: How decision-making can be comprised of the influence of distinct behaviors. The advantages of a behavior-based architecture in terms of decoupled software development. A historical account of behavior-based approaches.
- Action Selection: The role of action-selection for behavior reconciliation and the pros and cons of the several available options.
- Multi-objective Optimization: Understand the key concepts of multi-objective decision making, including the role of value functions, pareto optimality, and the application of multi-objective optimization to behavior-based architectures through the use of interval programming (IvP).
- Mission modes and structures: Understand how mission modes are used for identifying rele-

vant behaviors to accomplish the goals of the mission mode.

- `Dynamic behavior modification`: How and why one can dynamically modify a behavior at run-time. Understand the difference and similarity between mission-planning behavior configuration and run-time behavior configuration.
- `Behavior events`: Understand how behaviors maintain their own unique internal state representation of the world and how they can regard certain state changes as events. How behavior can be configured to generate/post information upon these events that affect other aspects of the autonomy system. Understand certain states an events that are available to all behaviors, and discern them from states and events that unique to a particular behavior.
- `Core behaviors`: Be familiar with certain core behaviors commonly found in most missions, such as waypoint following, obstacle and collision avoidance etc.
- `Mission Planning`: Understand the methods and role of mission planning, in terms of application and behavior configuration as well as specification of spatial coordinates. How to look for and address mistakes in mission configuration in the pre-launch and launch phases.
- `Multi-vehicle autonomy`: How the operation of a single vehicle generalizes to two or many vehicles. How mission launching and C2 generalizes to handle multiple vehicles, and how information is passed between vehicle MOOS communities and to/from vehicles to C2.
- `Inter-vehicle messaging`: Understand how inter-vehicle message are composed, routed and received between vehicles. The inherent limits of inter-vehicle messages in terms of range, frequency and bandwidth limits. Understand the different communication modalities available for platform types (surface or underwater) and operation area (near-shore or open water).
- `Collision and obstacle avoidance`: Understand the differences between avoiding stationary obstacles and moving contacts, the information that is needed for each and how this is generally obtained, and how this is handled in a behavior-based architecture.
- `Behavior spawning`: Extend the notion of a behavior-based architecture with a fixed set of behaviors created at the time of launch, to the operation of the architecture where behaviors are spawned upon events, to handled an ephemeral event, and subsequently destroyed. The contact manager and obstacle managers will also be introduced.
- `Core contact-behaviors`: Be familiar with certain core behaviors commonly found in all contact missions, e.g., collision avoidance, convoying, intercepting, formation keeping etc.
- `Voronoi-based missions`: Understand the concept of a protocol-based collaborative autonomy method, where coordination can be achieved with only limited, local and periodic inter-vehicle position communications.
- `Consensus-based missions`: Understand how richer inter-vehicle messaging, beyond sharing or position information, can be used by groups of vehicles to achieve a consensus of action. Basic inter-vehicle auctions and simple missions using auctions.
- `Post-mission analysis tools`: Learn how to use generated mission log files to post-process, replay and analyze prior missions. Learn the GUI based methods and CLI methods, and which situations are suitable for each tool.
- `Extending MOOS-IvP`: Obtain a template software tree from Github for user creation of MOOS apps and helm behaviors. Understand the basic structure of the software and build environment and how to augment your system shell environment to use new modules.
- `MOOS app development`: Learn how to develop a new MOOS application from scratch, using an app template. Understand the basics of mail handling, data processing and publishing of

results.

- `Payload Autonomy Hardware`: Assemble a payload autonomy computer from basic components, wiring and water-tight connections. Use of ssh keys and Git deploy keys for obtaining public autonomy code augmented with student developed code.
- `Basic robot field deployments`: Preparing robotic equipement for field operation, checking for systems health, integration of the autonomy computer for operation, launching and supervising a robot deployment, vehicle recovery, data offload and securing equipment for storage.
- `Automated mission testing`: How to modify a field-ready robot mission to be run in simulatin in a headless (no GUI) unsupervised mode suitable for randomizing starting conditions, environmental events, module configuration parameters or any combination of the three.
- `Automated post-mission analysis`: How to gather the results from several unsupervised automated simulations to convey trends in performance or detect outlier performance cases.

3 Class Mechanics

This course will be taught over ten days over two weeks. The first week consists of primarily simulation. During the initial week participants will learn to use existing autonomy software to create unique mission capabilities. Participants will also learn to augment the autonomy system with their own simple C++ applications. During the second week, participants will be provided their own payload autonomy computer suitable for embedding on the course robotic platforms. Field operations will consist of deployments of small autonomous surface vehicles (ASVs), deployed in initially as single-vehicle missions and then multi-vehicle 1-1 or 2-2 competitions.

Each day in the first week will be comprised of two half-day sessions, roughly 3-3.5 hours each. Each session begins by introducing new lecture material for about an 45 minutes, followed by a lab preview discussion, followed by a two hour self-contained lab. During the second week, in-water operations will be held in the mornings followed by in-lab instruction and simulation in the afternoons.

The lab exercises require either a Linux or MacOS system provided by each participant. Guidance will be provided on how to download and install the MOOS-IvP autonomy software and autonomy tools on participant laptops. External network connectivity is strongly preferred to allow for downloading of course material, labs and lab solutions.