MOOS-IvP Utility Applications

June 2018

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

1 Overview

The MOOS-IvP utilities are applications typically run either as part of an overall autonomy system on a marine vehicle, as part of a marine vehicle simulation, or used for post-mission off-line analysis. They are each MOOS applications, meaning they are running and communicating with a MOOSDB. The Alog Toolbox described here contains a number of off-line tools for analyzing alog files produced by the `pLogger` application.

2 Mission Monitoring Modules

Mission monitoring modules aid the user in either keeping a high-level tab on the mission as it unfolds, or help the user analyze and debug a mission. In release 13.2 this includes two powerful new tools for appcast monitoring, `uMAC` and `uMACView`. The `pMarineViewer` has also been substantially augmented to support appcast viewing.

- `pMarineViewer`: GUI tool for rendering events in an area of vehicle operation. It repeatedly updates vehicle positions from incoming node reports, and will render several geometric types published from other MOOS apps. The viewer may also post messages to the MOOSDB based on user-configured keyboard or mouse events.
- `uHelmScope`: A terminal-based (non-GUI) scope onto a running IvP Helm process, and key MOOS variables. It provides behavior summaries, activity states, and recent behavior postings to the MOOSDB. A very useful tool for debugging helm anomalies.
- `uXMS`: A terminal-based (non GUI) tool for scoping a MOOSDB Users may precisely configure the set of variables they wish to scope on by naming them explicitly on the command line or in the MOOS configuration block. The variable set may also be configured by naming one

or more MOOS processes on which all variables published by those processes will be scoped. Users may also scope on the *history* of a single variable.

- uProcessWatch: This application monitors the presence of MOOS apps on a watch-list. If one or more are noted to be absent, it will be so noted on the MOOS variable PROC_WATCH_SUMMARY. uProcessWatch is appcast-enabled and will produce a succinct table summary of watched processes and the CPU load reported by the processes themselves. The items on the watch list may be named explicitly in the config file or inferred from the Antler block or from list of DB_CLIENTS. An application may be excluded from the watch list if desired.
- uMAC: The uMAC application is a utility for Monitoring AppCasts. It is launched and run in a terminal window and will parse appcasts generated within its own MOOS community or those from other MOOS communities bridged or shared to the local MOOSDB. The primary advantage of uMAC versus other appcast monitoring tools is that a user can remotely log into a vehicle via ssh and launch uMAC locally in a terminal.
- uMACView: A GUI tool for visually monitoring appcasts. It will parse appcasts generated within its own MOOS community or those from other MOOS communities bridged or shared to the local MOOSDB. Its capability is nearly identical to the appcast viewing capability built into pMarineViewer. It was intended to be an appcast viewer for non-pMarineViewer users.

3 Mission Execution Modules

Mission execution modules participate directly in the proper execution of the mission rather than simply helping to monitor, plan or analyze the mission.

- pNodeReporter: A tool for collecting node information such as present vehicle position, trajectory and type, and posting it in a single report for sharing between vehicles or sending to a shoreside display.
- pBasicContactMgr: The contact manager deals with other known vehicles in its vicinity. It handles incoming reports perhaps received via a sensor application or over a communications link. Minimally it posts summary reports to the MOOSDB, but may also be configured to post alerts with user-configured content about one or more of the contacts. May be used in conjunction with the helm to spawn contact-related behaviors for collision avoidance, tracking, etc.
- pEchoVar: A tool for subscribing for a variable and re-publishing it under a different name. It also may be used to pull out certain fields in string publications consisting of comma-separated parameter=value pairs, publishing the new string using different parameters.
- pSearchGrid: An application for storing a history of vehicle positions in a 2D grid defined over a region of operation.

4 Mission Simulation Modules

Mission simulation modules are used only in simulation. Many of the applications in the uField Toolbox may also be considered simulation modules, but they also have a use case involving simulated sensors on actual physical vehicles. The two modules below are purely for simulated vehicles.

- uSimMarine: A simple 3D vehicle simulator that updates vehicle state, position and trajectory, based on the present actuator values and prior vehicle state. Typical usage scenario has a single instance of uSimMarine associated with each simulated vehicle.
- uSimCurrent: A simple application for simulating the effects of water current. Based on local current information from a given file, it repeatedly reads the vehicle's present position and publishes a drift vector, presumably consumed by uSimMarine.

5 Modules for Poking the MOOSDB

Poking the MOOSDB is a common and often essential part of mission execution and/or command and control. The pMarineViewer tool also contains several methods for poking the MOOSDB on user command.

- uPokeDB: A command-line tool for poking a MOOSDB with variable-value pairs provided on the command line. It finds the MOOSDB via mission file provided on the command line, or the IP address and port number given on the command line. It will connect to the DB, show the value prior to poking, poke the DB, and wait for mail from the DB to confirm the result of the poke.
- uTimerScript: Allows the user to script a set of pre-configured pokes to a MOOSDB with each entry in the script happening after a specified amount of time. Script may be paused or fast-forwarded. Events may also be configured with random values and happen randomly in a chosen window of time.
- uTermCommand: A terminal application for poking the MOOSDB with pre-defined variable-value pairs. A unique key may be associated with each poke.

6 The Alog Toolbox

The Alog Toolbox is set of offline tools for analyzing and manipulating alog files produces by the pLogger application distributed with the Oxford MOOS code base.

- alogview: A GUI tool for analyzing a vehicle mission by plotting one or more vehicle trajectories on the operation area, while viewing a plot of any of the numerical values in the alog file(s).
- alogscan: A command line tool for generating a summary report on contents of a given .alog file. The report lists each logged MOOS variable, which app(s) publish it, min/max publish time and total number of character and lines for the variable. The original alog file is not altered.
- alogclip: A command line tool that will create a new MOOS .alog file from a given .alog file by removing entries outside a given time window. The original alog file is not altered.
- aloggrep: A command line tool that will create a new MOOS .alog file by retaining only the given MOOS variables or sources from a given .alog file. The original alog file is not altered.
- alogpare: A command line tool that will create a new MOOS .alog file by paring back the given alog file in a two-pass manner. First pass detects events defined by given mark vars. The second pass removes lines with vars on the pare list if they are not within pare_window seconds of an event line. It also removes lines with vars on the hitlist unconditionally. Latter could also be done with alogrm. The original alog file is not altered.

- alogiter: A command line tool that will scan the given alog file and analyze the ITER_GAP and ITER_LEN information provided by applications. These variables record information related to the duration of the iterate loop for each application. This information helps indicate whether the CPU allowed the application to keep up with the requested application frequency. In higher time warps, apps are less able to keep up with the requested frequency.
- aloghelm: A command line tool that will scan the given alog file and produce one of several possible helm reports. One report will show the changes of behavior state. Another will show the changes in helm modes. Another will show helm life events, i.e., when behaviors are spawned or die.
- alogcheck: A command line tool that will scan the given alog file and determine if certain logic conditions have been satisfied. The tool can check for multiple conditions, and can be configured to check conditions before or after other conditions have been satisfied. Useful for automated benchtesting.
- alogcd: A command line tool that will scan the given alog file for collision detection reports. It will tally the totals and averages, and optionally create a file holding all the timestamps of events. By default it scans for events defined by postings to the MOOS variables COLLISION, NEAR_MISS, and ENCOUNTER.