Lab 16 - Autonomous Rescue Challenge - Part 4
Initial In-Water Competition

2.680 Unmanned Marine Vehicle Autonomy, Sensing and Communications

April 30th 2024

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

1 Overview and Objectives

In the fourth part of the Autonomous Rescue Challenge lab we return to the water and will run our missions on the Herons. The goal in this lab is to select two teams, four students, where each team fields a Heron running their own autonomy with their own version of pGenRescue. We will work with the same mission structure as in the previous lab, `rescue_baseline`. A TA will run the shoreside mission.

A summary of topics:

- Revisiting the PABLO operation of the Heron robot.
- Demonstration of student-provided path planning module, pGenRescue, in an adversarial situation.

2 Preliminaries

2.1 Preliminaries: Documentation Conventions

To help distinguish between MOOS variables, MOOS App configuration parameters, MOOS App names, behavior parameters, we will use the following conventions:

- MOOS variables are rendered in green, such as IVPHELM_STATE, as well as postings to the MOOSDB, such as DEPLOY=true.
- MOOS configuration parameters are rendered in blue, such as AppTick=10 and verbose=true.
- Behavior parameters are rendered in brown, such as priority=100 and endflag=RETURN=true.
- MOOS-IvP applications are rendered in magenta, such as pShare, or pHelmIvP.
- General GNU/Linux commands are represented in dark purple, such as wget, mkdir, or cd.

When distinguishing between command-line actions on a PABLO, and on your laptop, the below convention is used:

```
      $ cd moos-ivp    (a command executed on your local laptop)
PABLO $ cd moos-ivp    (a command executed on the PABLO computer)
```

2.2 Preliminaries: Pull Latest Code and Mission Files

At each stage of the lab we need to be mindful of the code on both the laptop and the PABLO. Changes may be made to (a) course-provided code, (b) the course baseline mission, and (c) your own `moos-ivp-extend` code. Use this section as a reminder to update frequently.

To recap, here are the relevant trees:

- `moos-ivp`: All the core autonomy code, and MOOS middleware
- `moos-ivp-2680`: uFldRescueMgr, gen_swimmers, and the baseline `BHV_Scout` behavior in `lib_bhv_scout`.
- `moos-ivp-extend`: Your pGenRescue and your `BHV_Scout` behavior.

- `moos-ivp-pavlab`: (PABLO only) When running on the Heron, we need the `iM300` app. This should already be on your provided PABLO.
- `pablo-common`: (PABLO only) Enviroment settings for the Heron. This should already be on your provided PABLO.

The `pablo-common` and `moos-ivp-pavlab` trees are only on your PABLO. The PABLO boot script is configured to automatically update the `pablo-common` tree when the PABLO boots and finds the Internet. If the `moos-ivp-pavlab` tree should happen to need an update, this would need to be done by entering the tree and performing an update and build. This is unlikely and only done if a change to `iM300` needs a change during the course of the lab.

Ensure the Latest Code and Updates *On your Laptop*

Updates will be needed on your laptop to run the initial simulations. You may also be using your laptop to act as the shoreside computer during field tests. You will need to update and possibly rebuild: the `moos-ivp`, `moos-ivp-2680`, and `moos-ivp-extend` trees:

```
$ cd ~/moos-ivp; svn update; ./build-ivp.sh
$ cd ~/moos-ivp-2680; svn update; ./build.sh
$ cd ~/moos-ivp-extend; svn update (or git pull); ./build.sh
```

In the first rescue lab you may not already have the `moos-ivp-2680` tree. It can be obtained as below, placing it alongside your `moos-ivp` and `moos-ivp-extend` trees. You will need to augment your shell path accordingly.

```
$ svn co https://oceanai.mit.edu/svn/moos-ivp-2680-aro/trunk moos-ivp-2680
```

Ensure the Latest Code and Updates *On your PABLO*

Updates will also be needed on your PABLO. Re-connect your PABLO to your laptop with Internet Sharing enabled, and update the following trees:

- The `moos-ivp` tree, and build
- The `moos-ivp-2680` tree, and build
- The `moos-ivp-pavlab` tree, and build
- The `pablo-common` tree (build not needed)
- The `moos-ivp-extend` tree, and build

The first four trees above are on your PABLO when you received it, but may need updating if changes were made during the course of the lab. The `moos-ivp-extend` tree was not on your PABLO when you received it (since it is your individual work), but should be on your PABLO as part of Lab 11 Introduction to the PABLO.

```
$ cd ~/moos-ivp; svn update; ./build-ivp.sh
$ cd ~/moos-ivp-2680; svn update; ./build.sh
$ cd ~/moos-ivp-pavlab; svn update; ./build.sh
$ cd ~/moos-ivp-extend; svn update (or git pull); ./build.sh
$ cd ~/pablo-common; svn update
```

Note: In a pinch, you can do any of the above steps while your PABLO is connected to a Heron, but the more you can do *before* you work with the Heron, the better. This will free up Heron time and TA time for other users.

2.3 Preliminaries: Pavilion Coordinates

The MIT Sailing Pavilion is the center of operations for the Pavlab, or Marine Autonomy Lab. The `(0,0)` coordinates, or datum, is located at: `42.358456, -71.087589`. These datum coordinates are declared in the top of every `.moos` file in this lab, and set in the plug file `plug_origin_warp.moos`:

```
// MIT Sailing Pavilion
LatOrigin  = 42.358456
LongOrigin = -71.087589
```

All other coordinates are usually configured in local coordinates relative to the above datum.

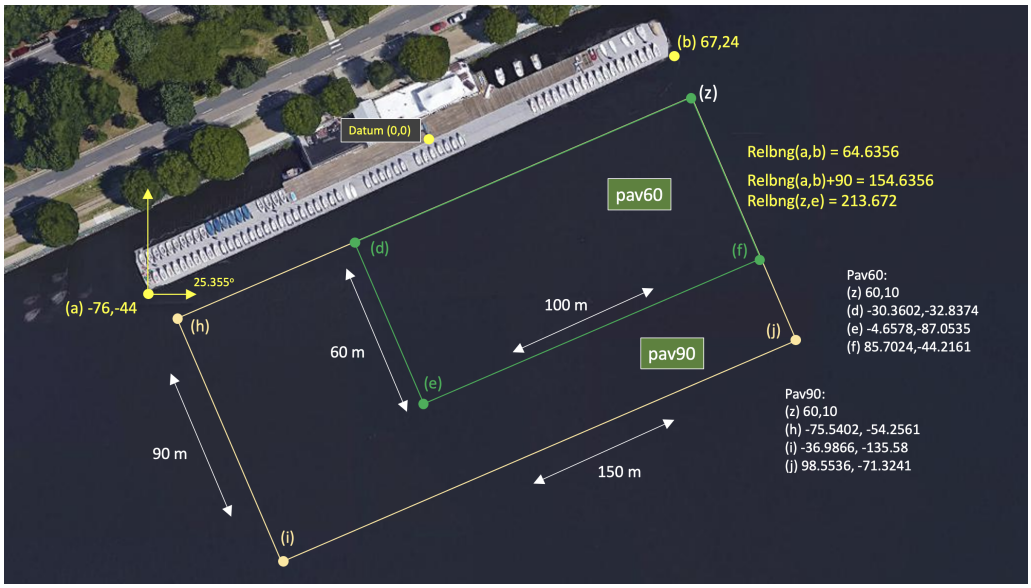

Figure 1: Useful coordinates related to the rescue lab. There are two playing field rectangles, a large one (`"pav90"`) roughly the extent of the docks, and a smaller region (`"pav60"`) closer to the Pavlab doors on the East end of the dock.

Note there are two rectangles representing a small and large operation area. The smaller area (`pav60`) is typically used for in-water tests involving one or two vehicles. The larger area (`pav90`) is typically used for in-water tests involving two-on-two competitions.

These two regions can be passed to the gen_swimmers app:

```
$ gen_swimmers = --pav60 --swimmers=15
$ gen_swimmers = --pav90 --swimmers=17 --unreg=11
```

Likewise, in the baseline mission, a new set of random swimmer locations can be generated at launch time with either of the two MIT regions:

```
$ ./launch.sh --pav60 --rescue 10
$ ./launch.sh --pav90 --rescue-rescue --swimmers=17 10
```

The boundaries of the playing field are read in by uFldRescueMgr from the swim file. The swim file is a key config parameter for this app, and it contains the location of all the swimmers. The region will will be communicated to the vehicles, through the MOOS variable RESCUE_REGION. For example, for the larger region above:

```
RESCUE_REGION = pts={60,10:99,-71:-37,-136:-76,-54}
```

The region message is received by the vehicle and is ingested by the `OpRegion` behavior in the helm, so the vehicle will know the playing field.

2.4 Preliminaries: The Shell Path

This lab sequence will use applications in the following locations:

- `moos-ivp/bin`
- `moos-ivp/scripts`
- `moos-ivp-2680/bin`
- `moos-ivp-extend/bin` (your code)

The `moos-ivp-2680` tree will hold a few apps specific to this lab, e.g., uFldRescueMgr, and the `moos-ivp-extend` tree will hold your app, pGenRescue. To augment your path, edit your `.bashrc` file as below:

```
PATH+=:~/moos-ivp/bin             (this was likely already there)
PATH+=:~/moos-ivp/scripts         (this was likely already there)
PATH+=:~/moos-ivp-extend/bin
PATH+=:~/moos-ivp-2680/bin
export PATH                       (this was likely already there)
```

You can confirm this by examining the contents of the `$PATH` bash environment variable:

```
$ echo $PATH
```

Confirmation can also be done with the GNU/Linux `which` function:

```
$ which pHelmIvP
/Users/you/moos-ivp/bin/pHelmIvP              (or /home/you/... in GNU/Linux)
$ which uFldRescueMgr
/Users/you/moos-ivp-2680/bin/uFldRescueMgr    (or /home/you/... in GNU/Linux)
```

There exists a convenience function in `moos-ivp/scripts` called `path_shell.sh`. This command will list your `$PATH` environment variable in a more human readable format. Normally the output is one long line with each directory separated by a colon:

```
$ echo $PATH
/usr/bin:/usr/local/bin:/bin:/Users/janedoe/bin:/Users/janedoe/project-pavlab/utils/bin:
/Users/janedoe/moos-ivp-janedoe/bin:/Users/janedoe/moos-ivp-pavlab/bin:/Users/janedoe/mo
os-ivp-extend/bin:/Users/janedoe/moos-ivp-2680/trunk/bin:/Users/janedoe/moos-ivp/trunk/b
in:/Users/janedoe/pablo-common/bin
```

The `path_shell.sh` utility will separate each directory out to a single line:

```
$ path_shell.sh
1: /usr/bin
2: /usr/local/bin
3: /bin
4: /Users/janedoe/bin
5: /Users/janedoe/project-pavlab/utils/bin
6: /Users/janedoe/moos-ivp-janedoe/bin
7: /Users/janedoe/moos-ivp-pavlab/bin
8: /Users/janedoe/moos-ivp-extend/bin
9: /Users/janedoe/moos-ivp-2680/trunk/bin
10: /Users/janedoe/moos-ivp/trunk/bin
11: /Users/janedoe/pablo-common/bin
```

2.5 Preliminaries: The Helm Behavior Path

Two behavior library folders need to be in your behavior path. They are:

- `moos-ivp-extend/lib`
- `moos-ivp-2680/lib`

The `moos-ivp-extend` tree will hold your `Scout` behavior when needed in the final part of this lab sequence. The `2680` library folder contains the staw-man version of the `Scout` behavior used for testing until replaced with your behavior. **Important:** When you are ready to use *your* Scout behavior, comment out the `2680` library from your behavior path.

If not done already, on your laptop and PABLO, augment your `.bashrc` file to augment the `IVP_BEHAVIOR_DIRS` to the following:

```
IVP_BEHAVIOR_DIRS+=~/moos-ivp-2680/lib      (comment out when you have your own)
IVP_BEHAVIOR_DIRS+=:~/moos-ivp-extend/lib
export IVP_BEHAVIOR_DIRS
```

Confirm your path afterwards by examining the `$IVP_BEHAVIOR_DIRS` environment variable:

```
$ echo $IVP_BEHAVIOR_DIRS
```

Note: The helm will not complete its startup process if it detects a behavior of the same name in two locations in the `IVP_BEHAVIOR_DIRS`. That may happen in this lab since the `BHV_Scout` behavior exists in the `moos-ivp-2680` tree, and then you will make your version in your `moos-ivp-extend` tree. Using the same name has its benefits since it allows us to have a common mission `meta_vehicle.bhv` file.

There exists a convenience function in `moos-ivp/scripts` called `path_bhv.sh`. This command will list your `$IVP_BEHAVIOR_DIRS` environment variable in a more human readable format. Normally the output is one long line with each directory separated by a colon:

```
$ echo $IVP_BEHAVIOR_DIRS
/home/janedoe/moos-ivp-foo/trunk/lib:/home/janedoe/moos-ivp-bar/lib:/home/jane\
doe/moos-ivp-2680/lib:/home/janedoe/moos-ivp-foobar/trunk/lib
```

The `path_bhv.sh` utility will separate each directory out to a single line:

```
$ path_bhv.sh
1: /home/janedoe/moos-ivp-foo/trunk/lib
2: /home/janedoe/moos-ivp-bar/lib
3: /home/janedoe/moos-ivp-2680/lib
4: /home/janedoe/moos-ivp-foobar/trunk/lib
```

3 The Baseline Mission for Today's Lab

We will continue to use the baseline mission from the previous labs, `moos-ivp-2680/missions/rescue_baseline`. This mission folder is updated as part of the `moos-ivp-2680` tree.

3.1 Launching the Vehicle Mission

To launch a mission, you will need to be logged into the PABLO on your robot, and know one piece of information:

- The IP Address of the computer running the shoreside software

Assuming your shoreside IP address is say 192.168.1.241, you will launch with:

```
PABLO  $ cd moos-ivp-2680/missions/rescue_baseline
PABLO  $ ./launch_vehicle.sh --shore=192.168.1.241 -v
```

It's always a good idea to run the above command with the `--verbose` or `-v` flag. This will present the user with the configuration summary, after which the launch can be continued by hitting the Enter key. You can also launch with the `--just_make`, or `-j` flag. This will just make the targ files, `targ_abe.moos` and `tar_abe.bhv` for example. If you see any error messages in this process, address them before continuing. Otherwise you should be ready to proceed.

3.2 A Note on Return Points

All of the below is FYI regarding how robot return points are set.

In the simulation version of this lab, the vehicle starting positions were chosen randomly in a manner that ensures they were unique and separated by a minimum distance. And the return waypoint behavior was configured to return to its starting position. The return waypoint is handled a bit differently in the in-water version of the lab.

In the in-water experiments, it is also helpful if the vehicles are not returning to the same point. And they don't have a "starting position" as they do in simulation. The unique points are assigned to each vehicle by their Heron name. For example, `eve` returns to position `(4,-11)`. The vehicle `ned` returns to `(12, -8)`. These are set in the `meta_vehicle.bhv` file in a conditional block that switches base on the detected vehicle name.

```
#ifdef VNAME abe
        point = 52,9
#elseifdef VNAME ben
        point = 39,4
#elseifdef VNAME cal
        point = 29,0
#elseifdef VNAME deb
        point = 16,-6
#elseifdef VNAME eve
        point = 4,-11
#elseifdef VNAME fin
        point = 2,-15
#elseifdef VNAME max
        point = 26,-2
#elseifdef VNAME ned
        point = 12,-8
#elseifdef VNAME ned
        point = 14,-10
#else
        point = $(START_POS)
#endif
```

All Herons will have one of these names. The `launch_vehicle.sh` script, when not running in simulation mode, will automatically detect which Heron you are in (based on the network configuration which is unique to each Heron).

3.3 Assignment 1 (check off) - Run the Vehicle on the Water

To complete this assignment, you will need another ready team to compete against.

If there is no second team at the moment, you can use your team's second PABLO box and run a competition between lab partners. In this case, presumably it will be a completely even 50-50 chance of win if you are both using the same code.

Save your alog files. We would like to have the alog files from each robot that ran a successful mission. When you are done with the robot, please save your alog file in your `moos-ivp-extend` tree per the instructions in Section 4.1.

4 Instructions for Handing In Assignments

4.1 Requested File Structure

The assignment for this lab is to (a) demonstrate a working mission on the Heron in a two-vehicle competition. (b) save your alog files.

```
moos-ivp-extend/
   src/
      pGenRescue
   missions/
      may0224/
        (your alog files)
```

4.2 Due Date

This lab should be completed by the end of day Thursday, May 2nd, 2024, but make up days may be available the following week.