Lab 4 - Working With Behaviors

Accelerated Marine Vehicle Autonomy,
Sensing, and Communications

May, 2017
國立台灣大學
National Taiwan University - Taiwan

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

1 Overview and Objectives

This lab will provide further experience with the IvP Helm to new users. It assumes some experience with the Helm, editing mission and behavior files, from the previous lab. It also assumes access to the corresponding lecture and lab overview lecture.

- The Loiter Behavior
- Simulating UUVs (adding depth to the simulations)
- The Timer Behavior
- The Behavior updates parameter
- Constant Depth behavior
- MaxDepth behavior
- Simulated Drift

1.1 Preliminaries

This lab assumes you have a working MOOS-IvP tree checked out and built on your computer. To verify this make sure that the following executables are built and findable in your shell path:

```
$ which MOOSDB
/Users/you/moos-ivp/bin/MOOSDB
$ which pHelmIvP
/Users/you/moos-ivp/bin/pHelmIvP
```

If unsuccessful with the above, return to the steps in the first lab and proceed until you are able to run the `s1_alpha` mission.
http://oceanai.mit.edu/ivpman/labs/machine_setup

1.2 Mission Files

A mission file is a single file, typically with a `.moos` suffix, where MOOS applications are configured.

1.3 Behavior Files

1.4 Mission Folders

2 Exercises

2.1 Exercise 1 - Changing the Alpha Mission to Use the Loiter Behavior

In this exercise the goal is make a version of the Alpha mission and swap out the Waypoint Behavior and use the Loiter Behavior instead. It should look something like:

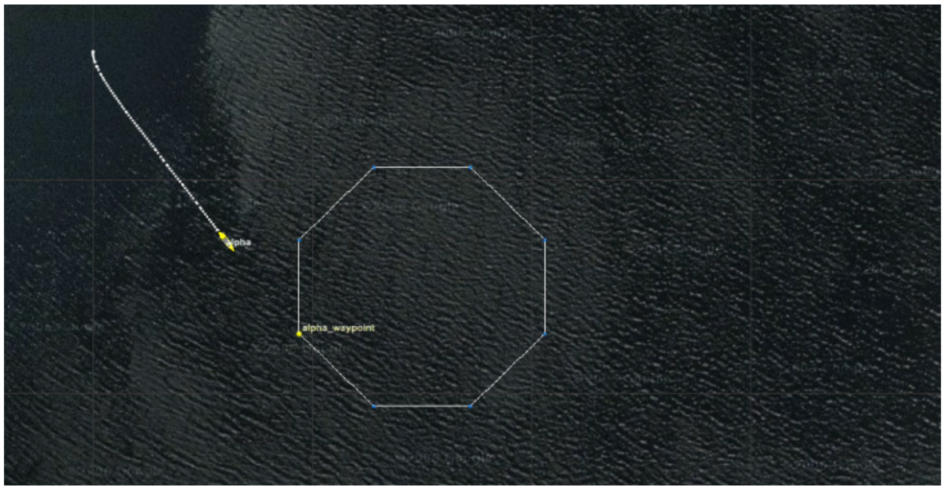


Figure 1: **s8_alpha:** Using the Loiter Behavior

You will need to do the following steps:

- Copy the `s1_alpha` mission, naming it `s8_alpha`
- Remove the Waypoint survey behavior (keep the return waypoint behavior)
- Add a Loiter Behavior

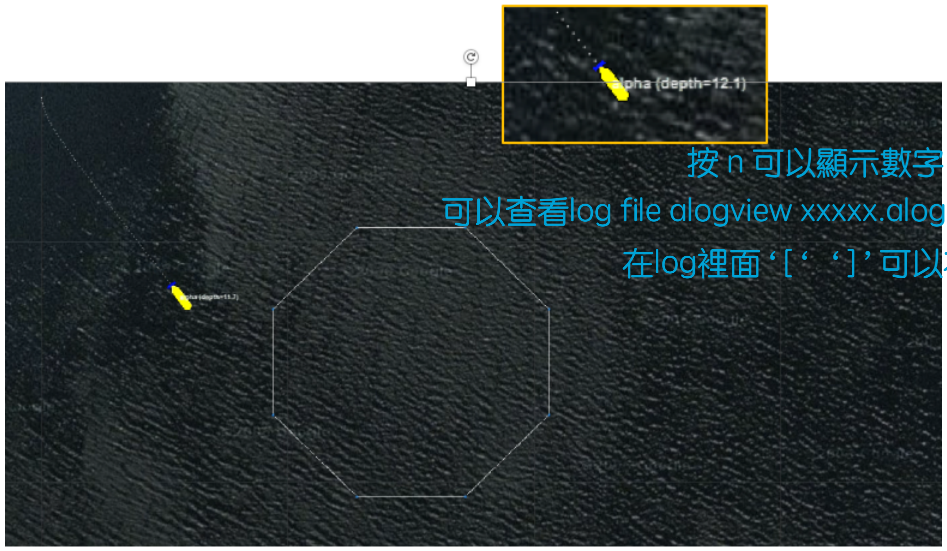
The Loiter behavior documentation is here:
`http://oceanai.mit.edu/ivpman/bhvs/Loiter`

The behavior block should look like that below. You can copy it into your behavior file, but try to understand the configuration parameters by reading the Loiter documentation above.

```
//----------------------------------------------
Behavior = BHV_Loiter
{
   name         = loiter
   priority     = 100
   condition    = RETURN = false
   condition    = DEPLOY = true
   updates      = UP_LOITER
   speed        = 1.3
   clockwise    = false
   radius       = 6.0
   slip_radius  = 25.0
   polygon      = format=radial, x=75, y=-75, radius=30, pts=8, snap=1
   visual_hints = nextpt_color=yellow, nextpt_lcolor=khaki
   visual_hints = edge_color=white, vertex_color=dodger_blue
   visual_hints = edge_size=1, vertex_size=3, label=LOITER_POLYGON
   visual_hints = nextpt_vertex_size=5
}
```

2.2 Exercise 2 - Adding Depth to our Missions, Constant Depth Behavior

In this exercise the goal is to modify the previous `s8_alpha` mission to have a vehicle that operates underwater, reasoning about depth in addition to heading and speed. It should look something like:



按 n 可以顯示數字

可以查看log file alogview xxxxx.alog

在log裡面 ‘[‘ ‘]’ 可以左右動

Figure 2: **s9_alpha:** A simulated UUV showing its current depth.

You will need to do the following steps:

- Copy the `s8_alpha` mission, naming it `s9_alpha`
- Configure the Helm to reason about depth
- Configure the simulator to reason about depth
- Configure the PID controller to reason about depth
- Add the ConstantDepth Behavior to the behavior file

2.2.1 Configure the Helm to Reason about Depth

The Helm documentation is at the link below. In particular, Section 6.4.6 discusses the helm decision space.

`http://oceanai.mit.edu/ivpman`

The helm configuration block should look like that below. You can copy it into your mission file, but try to understand the configuration parameters by reading the documentation.

```
ProcessConfig = pHelmIvP
{
  AppTick    = 4
  CommsTick  = 4

  bhv_dir_not_found_ok = true

  behaviors  = alpha.bhv
  domain     = course:0:359:360
  domain     = speed:0:4:41
  domain     = depth:0:100:101     <-- Add this line
}
```

新增深度域

2.2.2 Configure the Simulator to Reason about Depth

The Simulator documentation is at the link below.
http://oceanai.mit.edu/ivpman/apps/uSimMarine

The uSimMarine configuration block should look like that below. You can copy it into your mission file, but try to understand the configuration parameters by reading the documentation. The last four lines are needed with simulating depth.

```
ProcessConfig = uSimMarine
{
  AppTick   = 4
  CommsTick = 4

  start_x        = 0
  start_y        = -20
  start_heading  = 180
  start_speed    = 0

  prefix         = NAV

  turn_rate      = 40
  thrust_map     = 0:0, 20:1, 40:2, 60:3, 80:4, 100:5
  thrust_reflect = true

  buoyancy_rate        = 0.025
  max_depth_rate       = 5
  max_depth_rate_speed = 2.0
  default_water_depth  = 400
}
```

新增深度的動態

沒偵測到東西的時候default是400

2.2.3 Configure the PID Controller to Reason about Depth

You will need to add the below lines to the pMarinePID configuration block in your mission file to reason about depth.

```
  // Depth control configuration
  depth_control        = true    // or {false}
  z_to_pitch_pid_kp    = 0.12
  z_to_pitch_pid_kd    = 1.0
  z_to_pitch_pid_ki    = 0.004
  z_to_pitch_pid_integral_limit = 0.05
  maxpitch             = 15

  // Depth control configuration
  pitch_pid_kp         = 0.5
  pitch_pid_kd         = 1.0
  pitch_pid_ki         = 0
  pitch_pid_integral_limit = 0
  maxelevator          = 13
}
```

新增深度PID

2.2.4 Add the ConstantDepth Behavior to the behavior file

The ConstantDepth behavior documentation is at the below URL.
http://oceanai.mit.edu/ivpman/bhvs/ConstantDepth

You will need to add a configuration block like that below. You can copy it into your behavior file, but try to understand the configuration parameters by reading the documentation.

```
//-----------------------------------------------
Behavior=BHV_ConstantDepth
{
  name       = const_depth
  pwt        = 100
  condition  = DEPLOY = true
  duration   = no-time-limit

       depth = 20
}
```

2.3 Exercise 3 - Allow the User to Dynamically Change Depth

In this exercise the goal is to modify the previous `s9_alpha` mission to accept user specified changes to the commanded UUV depth, during the mission. The user should be able to toggle between commanded depths of 20 and 40 meters. Two buttons will be added to `pMarineViewer` to initiate the depth changes. It should look something like:

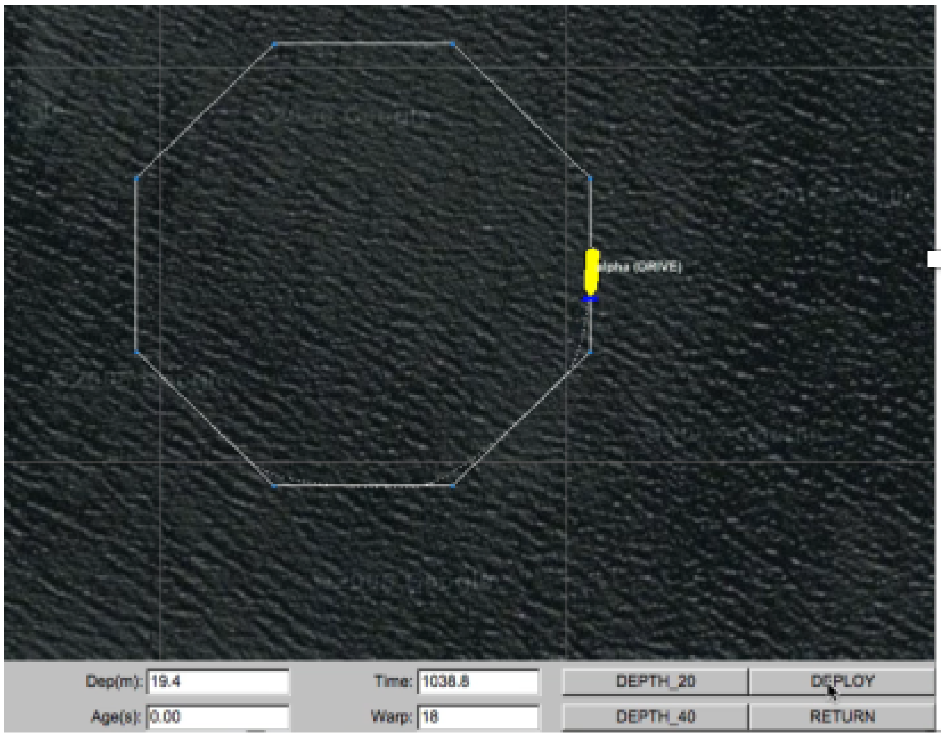


Figure 3: **s10_alpha:** A simulated UUV with two new operator buttons for changing the depth between 20 and 40 meters.

You will need to do the following steps:

- Copy the `s9_alpha` mission, naming it `s10_alpha`
- Add the updates parameter to the ConstantDepth behavior
- Add the appropriate buttons to the pMarineViewer configuration

2.3.1 Add the updates ConstantDepth Behavior

The `updates` parameter is defined for all IvP Behaviors. It specifies a MOOS variable from which updates to prior behavior configurations can be applied. Read more here:
`http://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php?n=Helm.HelmBehaviors#sec_updates`

You will need to add one line to the existing configuration block below:

```
//----------------------------------------------
Behavior=BHV_ConstantDepth
{
  name       = const_depth
  pwt        = 100
  condition  = DEPLOY = true
  duration   = no-time-limit
  updates    = DEPTH_UPDATE

       depth = 20
}
```

2.3.2 Add Buttons to pMarineViewer

The next step is to add a pair of buttons to the pMarineViewer configuration block that allow us to command different depths:

The pMarineViewer documentation is here (see Sec 47.1.5):
http://oceanai.mit.edu/ivpman/apps/pMarineViewer

You will need to add the below line to the existing configuration block below:

```
  button_three = DEPTH_20 # DEPTH_UPDATE=depth=20
  button_four  = DEPTH_40 # DEPTH_UPDATE=depth=40
```

2.4 Exercise 4 - Use a Script to Randomly Change Depth

In this exercise the goal is to copy the previous s10_alpha, to a new mission s11_alpha, to use a timer script to randomly change the vehicle depth between 20 and 80 meters.

You will need to do the following steps:

- Copy the s10_alpha mission, naming it s11_alpha
- Add the uTimerScript to the Antler block in the mission file
- Configure the uTimerScript

To enable uTimerScript to be launched, it needs to be added to the Antler configuration block in alpha.moos. Just add the last line as in the example below.

```
//------------------------------------------
// Antler configuration  block
ProcessConfig = ANTLER
{
  MSBetweenLaunches = 200

  Run = MOOSDB          @ NewConsole = false
  Run = pLogger         @ NewConsole = false
  Run = uSimMarine      @ NewConsole = false
  Run = pMarinePID      @ NewConsole = false
  Run = pHelmIvP        @ NewConsole = false
  Run = pMarineViewer   @ NewConsole = false
  Run = uProcessWatch   @ NewConsole = false
  Run = pNodeReporter   @ NewConsole = false
  Run = uTimerScript    @ NewConsole = false
}
```

The documentation for uTimerScript can be found at the below link:
   http://oceanai.mit.edu/ivpman/apps/uTimerScript

The configuration block should look like that below. You can copy it into your mission file, but try to understand the configuration parameters by reading the documentation.

```
//------------------------------------------
// uTimerScript configuration block
ProcessConfig = uTimerScript
{
  AppTick   = 4
  CommsTick = 4

  condition    = DEPLOY = true
  randvar      = varname=RND_DEPTH, min=20, max=80, key=at_reset
  event        = var=DEPTH_UPDATE, val=depth=$[RND_DEPTH], time=120
  reset_max    = nolimit
  reset_time   = all-posted
}
```

2.5 Exercise 5 - Use the MaxDepth Behavior to Limit Depth

In this exercise the goal is to extend the previous mission to include a MaxDepth behavior. This behavior will be used to make sure that the maximum command depth will be 50 meters, even though random depths between 20 and 80 meters will continue to be generated by the script.

You will need to do the following steps:

- Copy the `s11_alpha` mission, naming it `s12_alpha`
- Add a `MaxDepth` behavior to the behavior file, limiting the maximum commanded depth to be 50 meters
- Ensure the priority weight for this behavior is higher than the ConstantDepth behavior.

The documentation for `MaxDepth` behavior can be found at the below link:
`http://oceanai.mit.edu/ivpman/bhvs/MaxDepth`

The configuration block should look like that below. You can copy it into your behavior file, but try to understand the configuration parameters by reading the documentation.

```
//-----------------------------------------------
Behavior=BHV_MaxDepth
{
   name       = max_depth
   pwt        = 200
   condition  = DEPLOY = true
   duration   = no-time-limit
   max_depth  = 50
   basewidth  = 0
}
```

2.6 Exercise 6 - Simulate Surfacing for GPS

In this exercise the goal is to extend the `s9_alpha` mission to have a loitering UUV periodically stop and float to the surface, and weight at the surface for a period of time (presumably to get a GPS fix). Two new behavior instances will be added. Both are Timer behaviors. These behavior do not produce any objective function, no influence over heading, speed or depth decisions. They simply use the built-in timer (duration) feature available to all behaviors, to periodically post end flags.

You will need to do the following steps:

- Copy the `s9_alpha` mission, naming it `s13_alpha`
- Add a `Timer` behavior to the behavior file. This behavior will be on when (a) the vehicle is deployed (`DEPLOY=true`), and when a GPS fix is not needed (`NEED_GPS=false`). It will have duration of 300 seconds, and when it completes, it will declare that a GPS fix is needed (`NEED_GPS=true`).
- Add a second `Timer` behavior to the behavior file. This behavior will be on when (a) the vehicle is deployed (`DEPLOY=true`), and when a GPS fix is needed (`NEED_GPS=true`), and when the vehicle is still not yet at the surface (`NAV_DEPTH < 2`). It will have duration of 30 seconds, and when it completes, it will declare that a GPS fix is not needed (`NEED_GPS=false`).
- The behavior file will need to declare that initially a GPS fix is not needed (`initialize NEED_GPS = false`).

The documentation for `MaxDepth` behavior can be found at the below link:
http://oceanai.mit.edu/ivpman/bhvs/MaxDepth

The configuration block should look like that below. You can copy it into your behavior file, but try to understand the configuration parameters by reading the documentation.

```
//-----------------------------------------------
Behavior=BHV_MaxDepth
{
   name       = max_depth
   pwt        = 200
   condition  = DEPLOY = true
   duration   = no-time-limit
   max_depth  = 50
   basewidth  = 0
}
```

2.7 Exercise 7 - Add Simulated Drift to the Mission

In this exercise the goal is to extend the `s13_alpha` mission to have include simulated drift in the simulation

You will need to do the following steps:

- Copy the `s13_alpha` mission, naming it `s14_alpha`
- Add two buttons to the `pMarineViewer` configuration to turn drift on and off. Add a drift at of 0.3 m/sec at heading 45 degrees.
- Note the performance of the Loiter behavior under drift.

The documentation for the `uSimMarine` MOOS App can be found at the below link. The relevant part for this exercise is the drift vector feature. (This is just FYI - no need to modify the simulator in this exercise)
http://oceanai.mit.edu/ivpman/apps/uSimMarine

The line in the `pMarineViewer` configuration block should look like that below. You can copy it into your behavior file, but try to understand the configuration parameters by reading the documentation. The last two lines are the key additions for this exercise.

```
button_one = DEPLOY # DEPLOY=true
button_one = MOOS_MANUAL_OVERRIDE=false # RETURN=false
button_two = RETURN # RETURN=true

button_three = DRIFT_ON  # DRIFT_VECTOR = 45,0.3
button_four  = DRIFT_OFF # DRIFT_VECTOR = 0,0
```

3 Where to Find the Solutions

All solutions can be dowloaded from the server:

Exercise 1:

- `http://oceanai.mit.edu/ntu/s8_alpha/alpha.moos`
- `http://oceanai.mit.edu/ntu/s8_alpha/alpha.bhv`

Exercise 2:

- `http://oceanai.mit.edu/ntu/s9_alpha/alpha.moos`
- `http://oceanai.mit.edu/ntu/s9_alpha/alpha.bhv`

Exercise 3:

- `http://oceanai.mit.edu/ntu/s10_alpha/alpha.moos`
- `http://oceanai.mit.edu/ntu/s10_alpha/alpha.bhv`

Exercise 4:

- `http://oceanai.mit.edu/ntu/s11_alpha/alpha.moos`
- `http://oceanai.mit.edu/ntu/s11_alpha/alpha.bhv`

Exercise 5:

- `http://oceanai.mit.edu/ntu/s12_alpha/alpha.moos`
- `http://oceanai.mit.edu/ntu/s12_alpha/alpha.bhv`

Exercise 6:

- `http://oceanai.mit.edu/ntu/s13_alpha/alpha.moos`
- `http://oceanai.mit.edu/ntu/s13_alpha/alpha.bhv`

Exercise 7:

- `http://oceanai.mit.edu/ntu/s14_alpha/alpha.moos`
- `http://oceanai.mit.edu/ntu/s14_alpha/alpha.bhv`