MIT 2.680

UNMANNED MARINE VEHICLE AUTONOMY, SENSING, AND COMMUNICATIONS

Lecture 9: A Deeper Dive Into Behaviors

March 12th, 2026

Web: http://oceanai.mit.edu/2.680

Email:
Mike Benjamin, mikerb@mit.edu

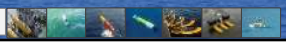


Photo by Arjan Vermeij, CMRE

1

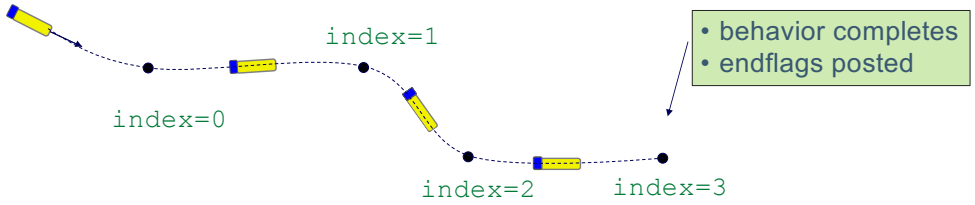
MITMECHE

The Waypoint Behavior (Deeper Dive)

2

Traversing Waypoints

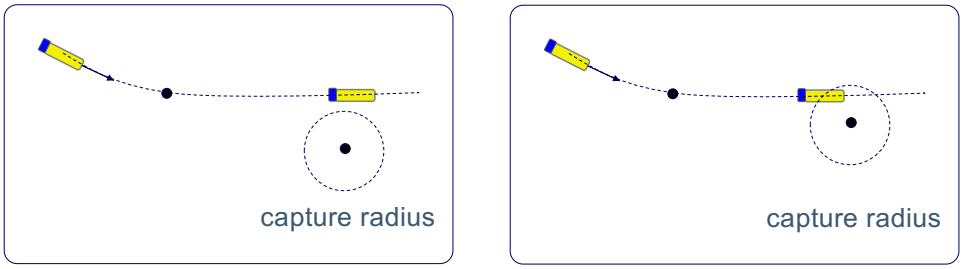
- The set of waypoints, will be traversed in order. Each waypoint has an index
- Upon each waypoint, a waypoint flag may be posted, if configured in the mission
- The behavior will completes when it has visited all waypoints




```
points  = 60,40 : 120,40 : 150,0 : 200,0
endflag = RETURN=true
wptflag = MEASURE=true
```


Achieving a Waypoint – Capture Radius

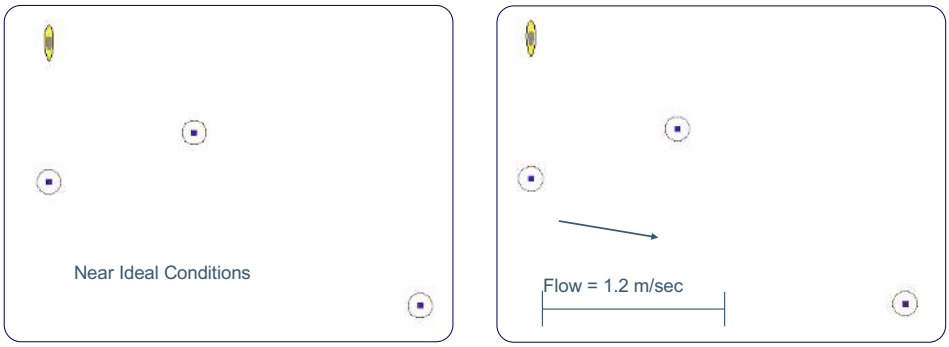
- A vehicle cannot hit a waypoint exactly
- The capture radius determines how close is "good enough"
- Appropriate value depends on quality of control system, navigation, mission objectives




```
points  = 60,40 : 120,40 : 150,0 : 200,0
capture_radius = 10
```

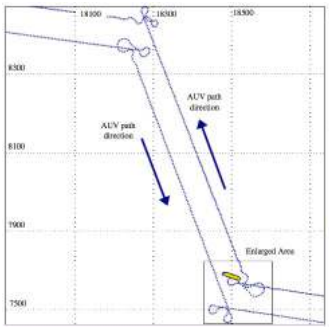
Missing a Waypoint – Loop Backs

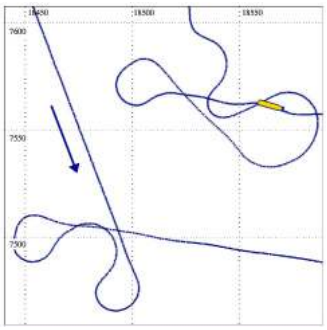
- The loop back occurs when the vehicle barely misses its waypoint.
- The resulting trajectory is a very tight turn, potentially risking the vehicle
- One cause can be not properly accounting for wind, current or other external forces



Monterey Bay California 2006

Adverse Affects of Loop-Backs

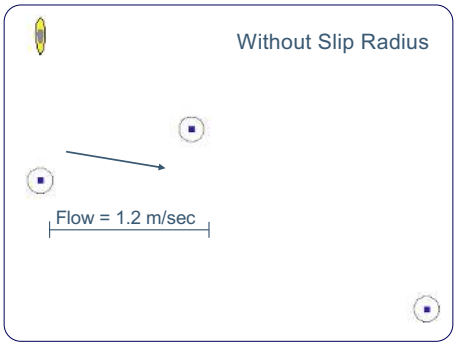


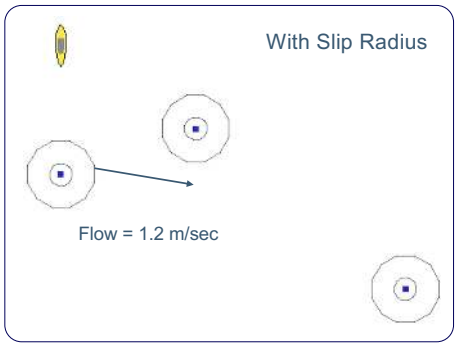


7

Achieving a Waypoint – Slip Radius

- Larger capture radius reduces loop-backs, but means you "arrive" sooner
- The slip radius allows the capture radius to be missed, but still achieve the waypoint
- If the vehicle enters the slip radius, and begins to exit, we say the point is achieved



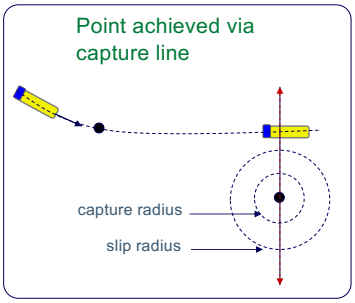



```
points  = 60,40 : 120,40 : 150,0 : 200,0
capture_radius = 10
slip_radius = 25
```


8

Achieving a Waypoint – Capture Line

- A capture line is an additional capture criteria, when robot crosses the line
- Line is perpendicular to the line between the waypoint and the point when the robot begins striving for that point

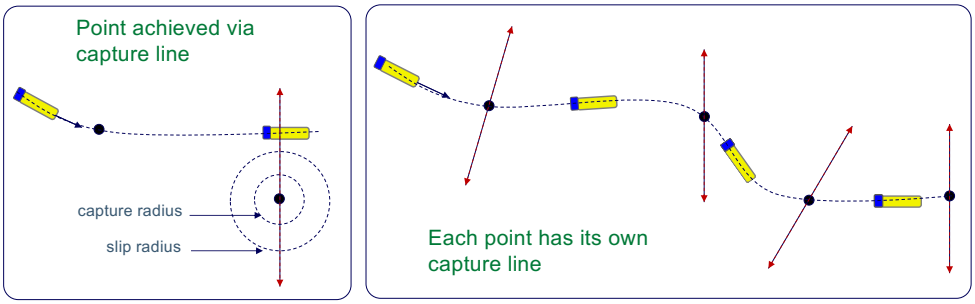



```
points  = 60,40 : 120,40 : 150,0 : 200,0
capture_radius = 10
slip_radius = 25
capture_line = true
```


9

Achieving a Waypoint – Capture Line

- A capture line is an additional capture criteria, when robot crosses the line
- Line is perpendicular to the line between the waypoint and the point when the robot begins striving for that point



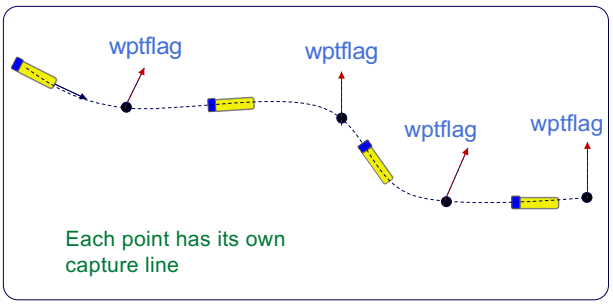

```
points  = 60,40 : 120,40 : 150,0 : 200,0
capture_radius = 10
slip_radius = 25
capture_line = true
```

Behaviors Overview | Waypoint Behavior | LegRun Behavior | FixedTurn FullStop | Obstacle Avoidance | Dynamic Updates | Loiter Behavior | Min/Max Depth | OpRegionV24 Behavior | StationKeep Behavior
Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

10

Achieving a Waypoint – wptflag

- A wptflag may be configured to be posted each time the behavior achieves a waypoint.
- Like other flags (e.g., endflag), a wptflag is a MOOS Variable / Value pair.

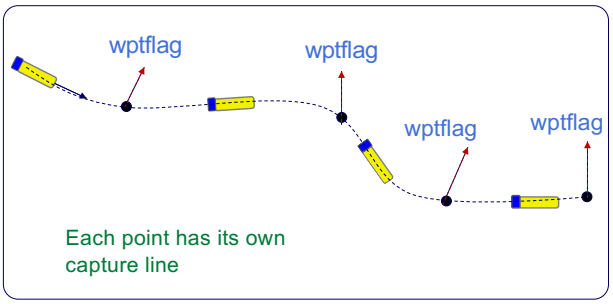



```
points  = 60,40 : 120,40 : 150,0 : 200,0
capture_radius = 10
slip_radius = 25
capture line = true
wptflag = WPT_FRESH=true
```

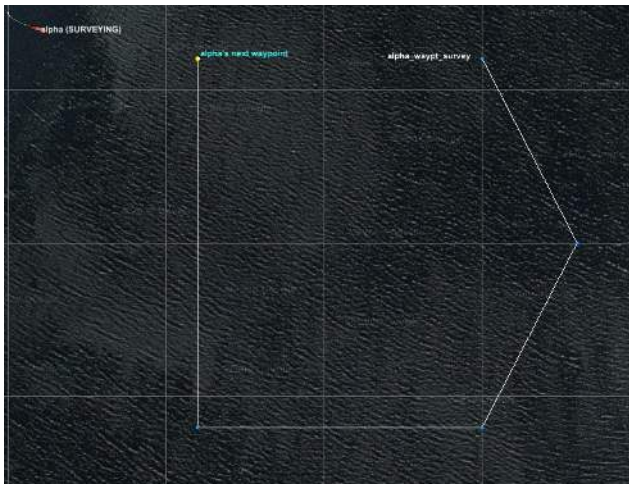

11

Achieving a Waypoint – wptflag

- A wptflag may be configured to be posted each time the behavior achieves a waypoint.
- Like other flags (e.g., endflag), a wptflag is a MOOS Variable / Value pair.




```
points  = 60,40 : 120,40 : 150,0 : 200,0
capture_radius = 10
slip_radius = 25
capture line = true
wptflag = WPT_FRESH=true
```

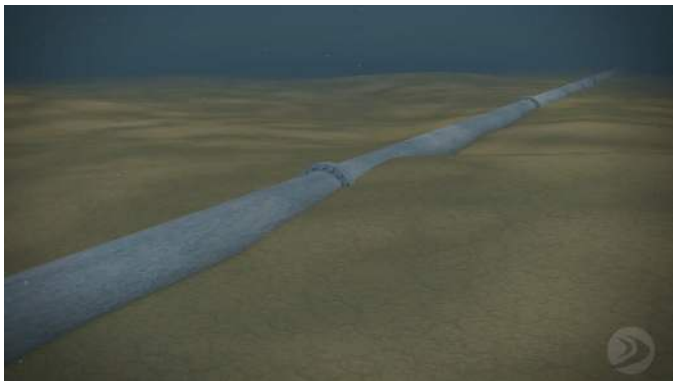


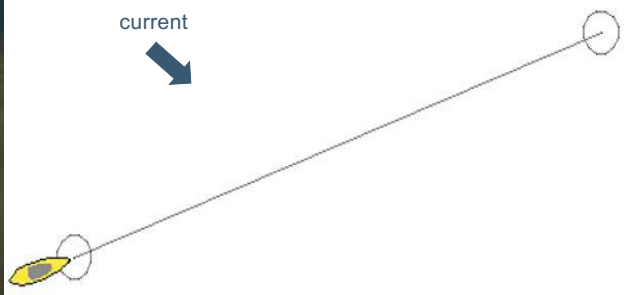
Behaviors Overview | Waypoint Behavior | LegRun Behavior | FixedTurn FullStop | Obstacle Avoidance | Dynamic Updates | Loiter Behavior | Min/Max Depth | OpRegionV24 Behavior | StationKeep Behavior
Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

12

Track-line Following

- In some missions, a vehicle needs to follow a track-line, for optimal sensing
- This may be hard due to vehicle dynamics
- The environment (current, wind) may also cause problems

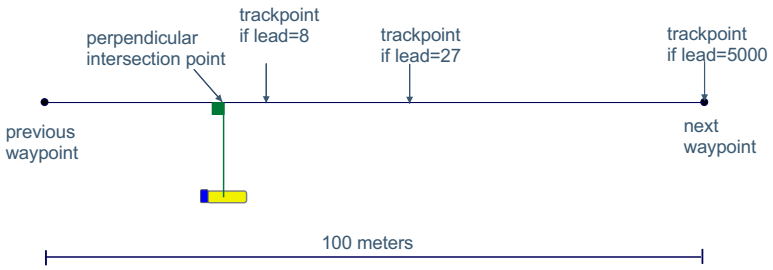




13

The Track Point

- The lead parameter specifies an imaginary point on the track line, the track point
- The lead distance is from the perpendicular intersection point



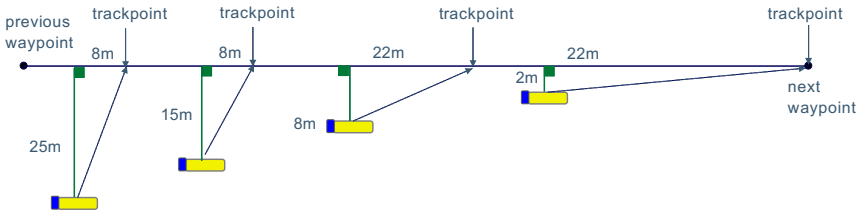

```
points  = 60,40 : 120,40 : 150,0 : 200,0
lead = 8
```


14

Track Point Damper

- The lead_damper parameter allows the track point to be adjusted outward as the vehicle gets closer to the track line.
- The lead_damper is the range to the track line, beyond which the lead distance is the tightest.

```
Example: lead=8
         lead_damper=15
```

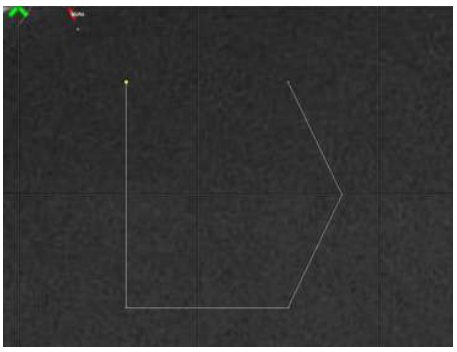



```
points      = 60,40 : 120,40 : 150,0 : 200,0
lead        = 8
lead_damper = 15
```

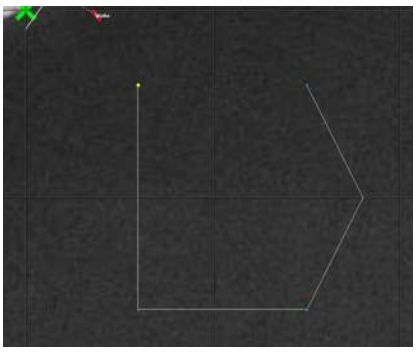

15

Alpha With and Without Track-Line

Alpha With Track-Line



Alpha Without Track-Line



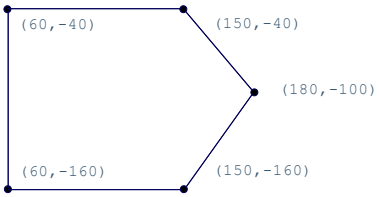
```
points  = 60,-40 : 60,-160 : 150,-160 : 180,-100 : 150,-40
capture_radius = 5
slip_radius = 15
lead = 8
```


16

Specifying Waypoints Explicitly

- Waypoints may be configured explicitly (as in the Alpha mission)

```
points  = 60,-40 : 60,-160 : 150,-160 : 180,-100 : 150,-40
```



- Or simply a single point

```
point  = 60,-40
```


- Or simply

```
point  = start
```

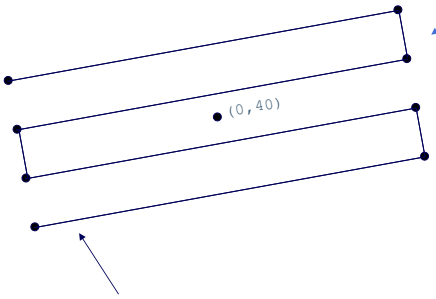

17

Waypoints as a Lawnmower Pattern

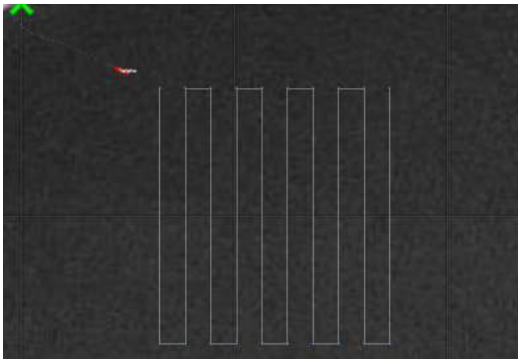
- Waypoints may be configured implicitly via lawnmower pattern parameters

```
points  = format=lawnmower, label=foxtrot, x=0, y=40, height=60, width=180,
          lane_width=15, rows=east-west, degs=45, startx=-20, starty=-300
```

Rotation specified



The first waypoint is the closest to the point given by `startx` and `starty`

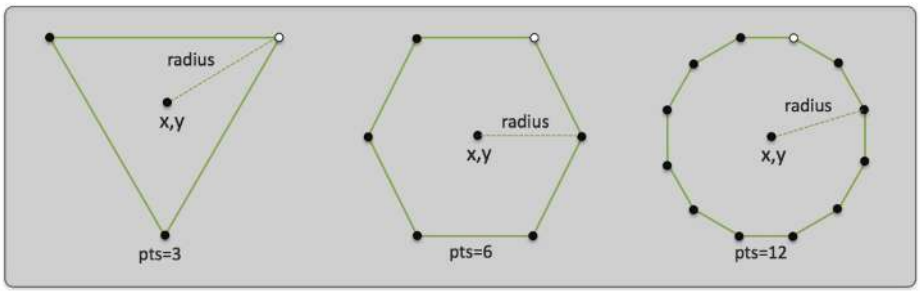


18

Waypoints as a Radial Polygon

- Waypoints may be configured with radial/circular pattern parameters

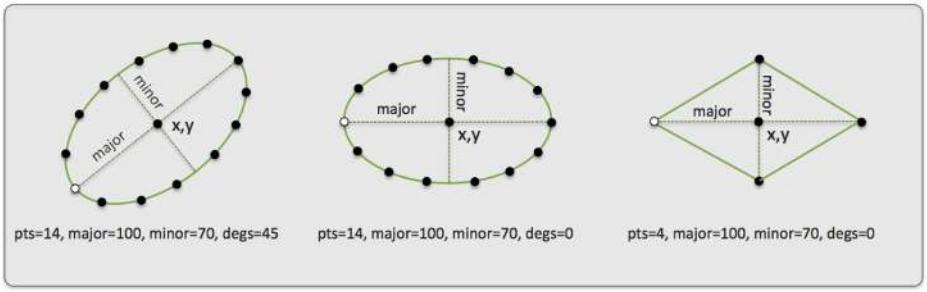
```
polygon = format=radial, x=0, y=40, radius=60, pts=6, snap=1
```



Waypoints as an Ellipse

- Waypoints may be configured with elliptical pattern parameters

```
polygon = format=ellipse, x=0, y=40, degs=45, pts=14, snap=1, major=100, minor=70
```

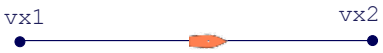


LegRun Behavior

21

The Leg Run Behavior

The **LegRun** behavior is originally designed to facilitate the testing of a marine robotic platform by enabling the vehicle traverse back and forth over a legrun track line prescribed by two vertices.



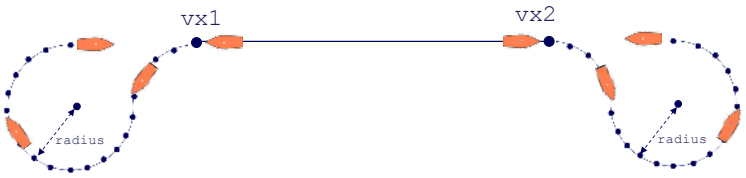
- Configuration Parameters:

vx1	location of vertex 1
vx2	location of vertex 1
speed	Traversal speed, m/s

22

The Leg Run Behavior

The **LegRun** behavior is originally designed to facilitate the testing of a marine robotic platform by enabling the vehicle traverse back and forth over a legrun track line prescribed by two vertices. After reaching the end of the track line, the vehicle switches into a Williamson turn mode to ensure the vehicle aligns with the track line when starting in the other direction.

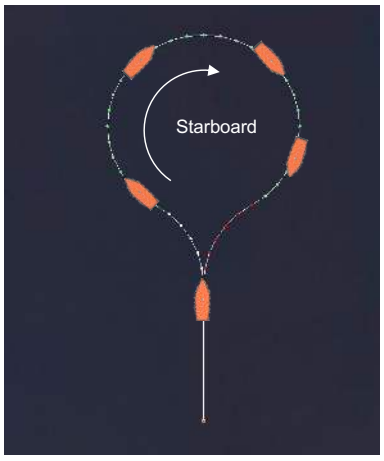


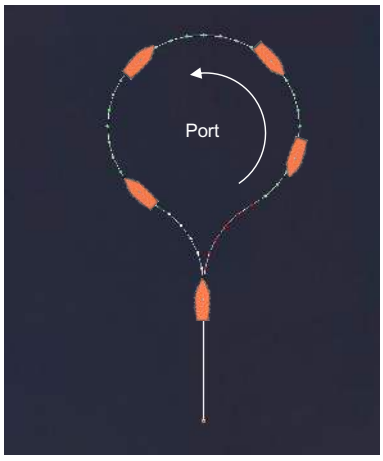
Configuration Parameters:

vx1	Location of vertex 1
vx2	Location of vertex 1
speed	Traversal speed, m/s
turn1_rad	Turn radius at vx1
turn2_rad	Turn radius at vx
turn_rad	Turn radius at both ends

Williamson Turns

- The **Williamson turn** is an alternative manoeuvre used to bring a ship or boat under power back to a point it previously passed through, often for the purpose of recovering a casualty at sea.
- It was named for John Williamson, USNR, who used it in 1943 to recover a man who had fallen overboard. However, according to *Uncommon Carriers* by John McPhee, the maneuver was originally called the *Butakov pipe* and was used in the Russo-Japanese War as a way of keeping guns at the same distance from an enemy. It was also used by U.S. Navy nuclear submarines to clear their sonar dead zones. - Wikipedia

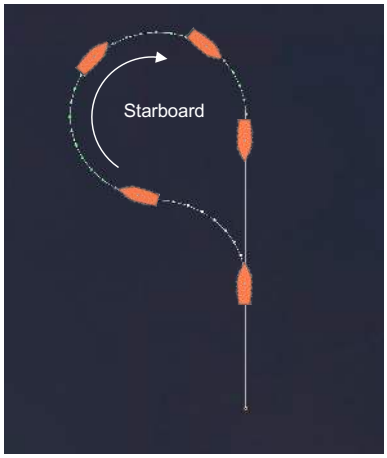


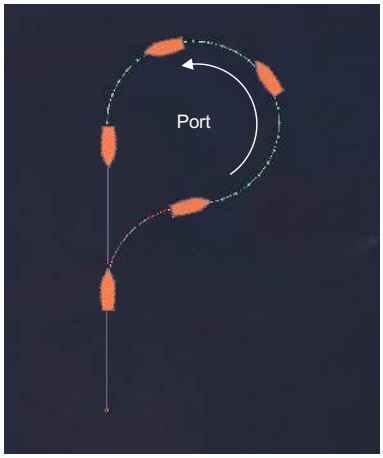


Williamson Turns

Differences in our case:

- Unlike a man-overboard maneuver, our primary objective is to drive straight back down the track line from which we came.
- We will implement and execute our turn by using generated waypoints

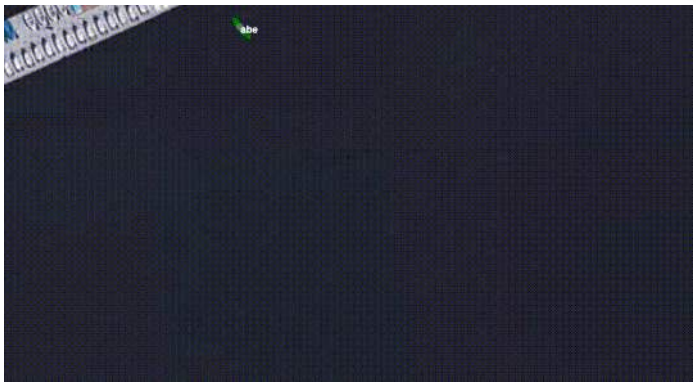




The Leg Run Behavior

The *turn direction* can be set individually for each end. In this case both turns are starboard turns (the center point is always on the starboard side of the vehicle).

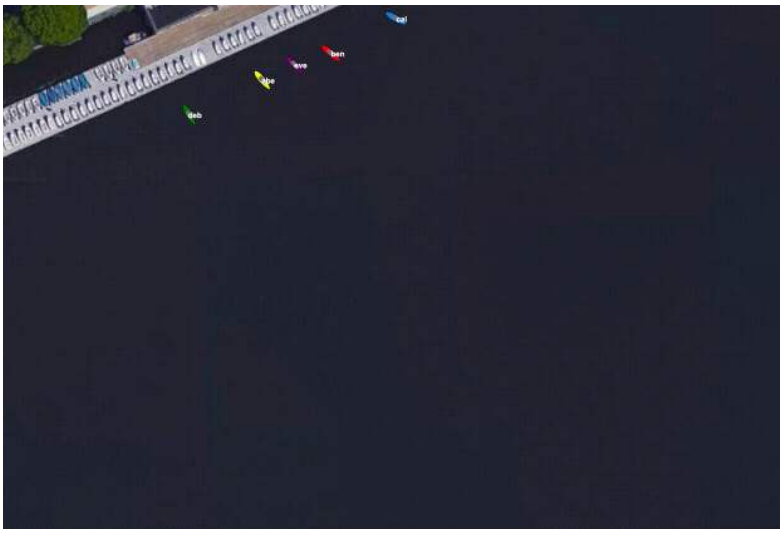
vx1	Location of vertex 1
vx2	Location of vertex 1
speed	Traversal speed, m/s
turn1_rad	Turn radius at vx1
turn2_rad	Turn radius at vx
turn_rad	Turn radius at both ends
turn1_dir	Port/Star turn at vx1
turn2_dir	Port/Star turn at vx2
turn_dir	Port/Star turn both ends



The Leg Run Behavior (with Synchronization)

Coordination: Introducing inter-vehicle messaging, the LegRun behavior can be configured to coordinate timing between vehicles.

vx1	Location of vertex 1
vx2	Location of vertex 1
speed	Traversal speed
coord	true/false
coord_on_leg	true/false



27

The Leg Run Behavior (with Synchronization)

Coordination: Introducing inter-vehicle messaging, the LegRun behavior can be configured to coordinate timing between vehicles.

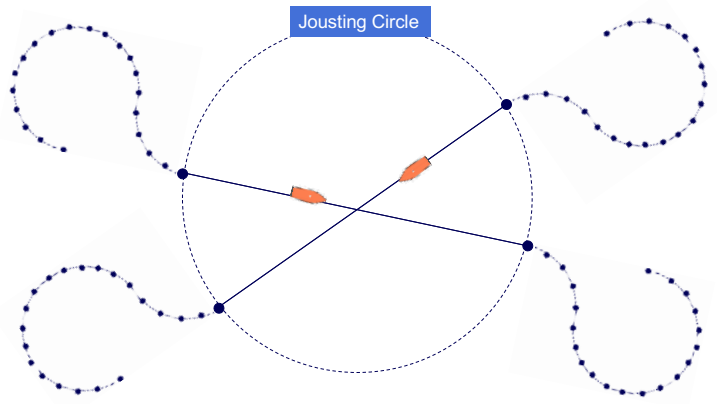
Behaviors Overview | Waypoint Behavior | LegRun Behavior | FixedTurn FullStop | Obstacle Avoidance | Dynamic Updates | Loiter Behavior | Min/Max Depth | OpRegionV24 Behavior | StationKeep Behavior
Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

28

The Leg Run – For Jousting

The LegRun: is a utility behavior that can be used for setting up experiments.

One of those experiments is a Jousting Mission for testing collision avoidance capabilities.



29

The Leg Run – For Jousting

The LegRun: is a utility behavior that can be used for setting up experiments.

One of those experiments is a Jousting Mission for testing collision avoidance capabilities.

3-Vehicle Jousting

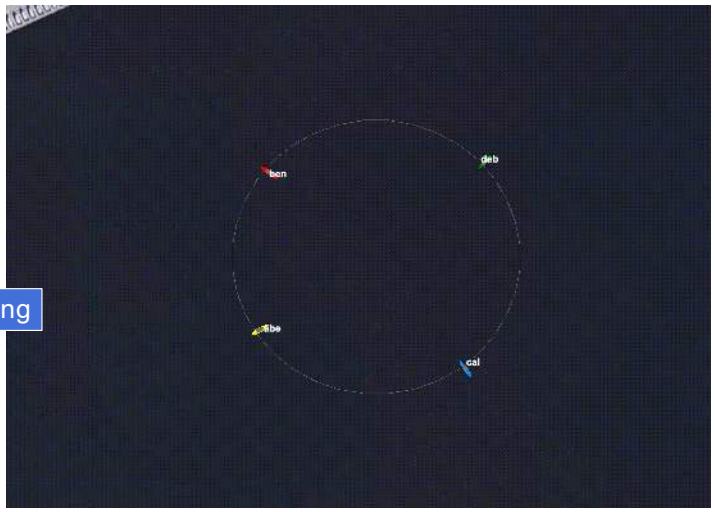
30

The Leg Run – For Jousting

The LegRun: is a utility behavior that can be used for setting up experiments.

One of those experiments is a Jousting Mission for testing collision avoidance capabilities.

4-Vehicle Jousting



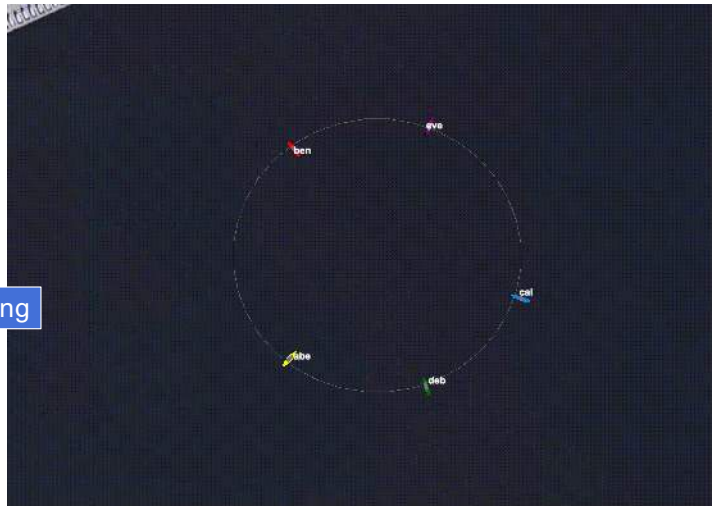
31

The Leg Run – For Jousting

The LegRun: is a utility behavior that can be used for setting up experiments.

One of those experiments is a Jousting Mission for testing collision avoidance capabilities.

5-Vehicle Jousting

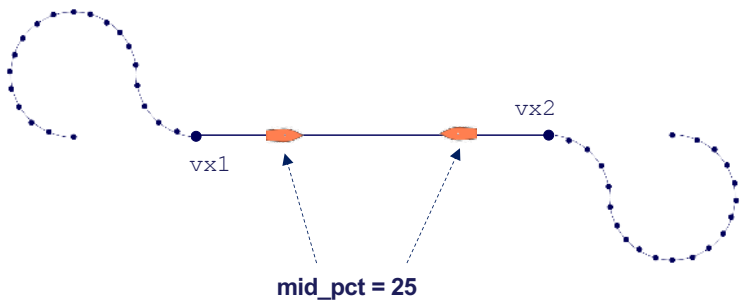


32

The Leg Run Behavior

Configuration Parameters:

vx1	Location of vertex 1
vx2	Location of vertex 1
speed	Traversal speed, m/s
turn1_rad	Turn radius at vx1
turn2_rad	Turn radius at vx
turn_rad	Turn radius at both ends
turn1_dir	Port/Star turn at vx1
turn2_dir	Port/Star turn at vx2
turn_dir	Port/Star turn both ends
mid_flag	Configurable event flag
mid_pct	Pt on leg for mid_flag



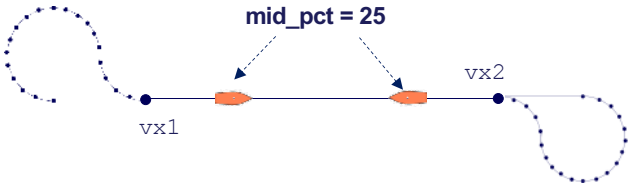
The **mid_flag** is like any other event flag. It holds a MOOS variable and value to be posted. For example:

```
mid_flag = FIX_TURN=true
```


The Leg Run Behavior

Configuration Parameters:

vx1	Location of vertex 1
vx2	Location of vertex 1
speed	Traversal speed, m/s
turn1_rad	Turn radius at vx1
turn2_rad	Turn radius at vx
turn_rad	Turn radius at both ends
turn1_dir	Port/Star turn at vx1
turn2_dir	Port/Star turn at vx2
turn_dir	Port/Star turn both ends
mid_flag	Configurable event flag
mid_pct	Pt on leg for mid_flag



The **mid_flag** is like any other event flag. It holds a MOOS variable and value to be posted. For example:

```
mid_flag = FIX_TURN=true
```

Behaviors Overview | Waypoint Behavior | LegRun Behavior | FixedTurn FullStop | Obstacle Avoidance | Dynamic Updates | Loiter Behavior | Min/Max Depth | OpRegionV24 Behavior | StationKeep Behavior
Michael Benjamin ©2026 MIT Dept of Mechanical Engineering
34

The FixedTurn Behavior

The **FixTurn** behavior allows a vehicle to execute a turn with parameters for configuring the duration of the turn (in degrees), and the commanded changes in desired heading during the turn. The behavior is primarily motivated by supporting surface vehicle shake-out tests, but may be used in any mission where this pattern is desired. The behavior can be configured with varying number of turns, order of turns, and the number of degrees for each turn, and the manner in which the turn is requested.

Turns in a FixedTurn behavior are accomplished by continuously requesting a heading change of a configured fixed value, monitoring the number of degrees in a turn, until the turn completion

Configuration Parameters:

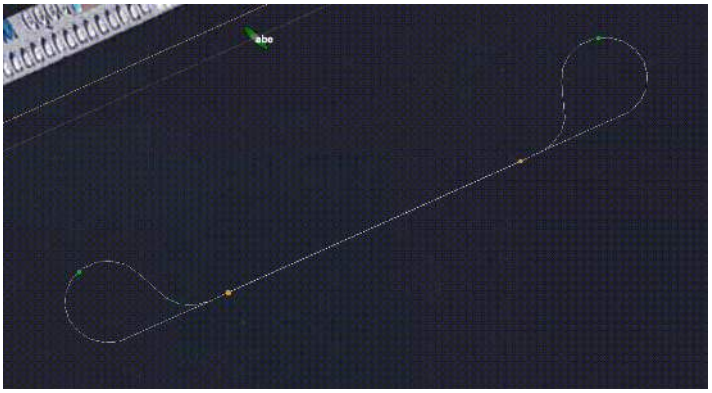
fix_turn	Turn threshold in degrees
mod_heading	Requested delta heading
turn_dir	Port or starboard
speed	Speed during turn
timeout	Timeout duration for turn

The Full Stop Behavior

The **FullStop** behavior is dead simple behavior designed to facilitate the testing of a marine robotic platform by transitioning into an all-stop, monitoring the decreasing ownship speed, and declaring the full stop to complete when it reaches a configurable target speed.

- Begin deceleration (zero thrust)
- Target speed reached

Optional params:
- Post mark_flag
- mark_duration
- Post unmark_flag
- delay_complete
- Post end_flags



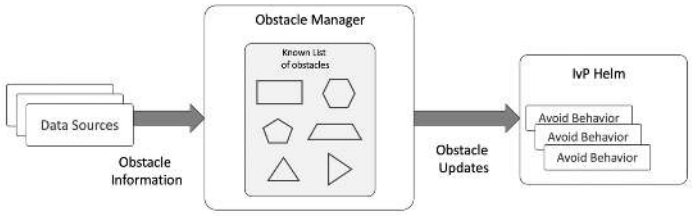
Configuration Parameters:

stop_thresh	Speed considered to be stopped
mark_flag	Flag posted at mark event
unmark_flag	Flag posted at unmark event
mark_duration	Duration of mark phase
delay_complete	Extra time after end of mark phase before end flags posted

Obstacle Avoidance

The **AvoidObstacle** behavior is a dynamically spawned behavior that works in conjunction with the obstacle manager

The **Obstacle Manager** is a separate MOOS App that interfaces with the sensor applications to maintain a list of known obstacles and locations.



Obstacle Avoidance

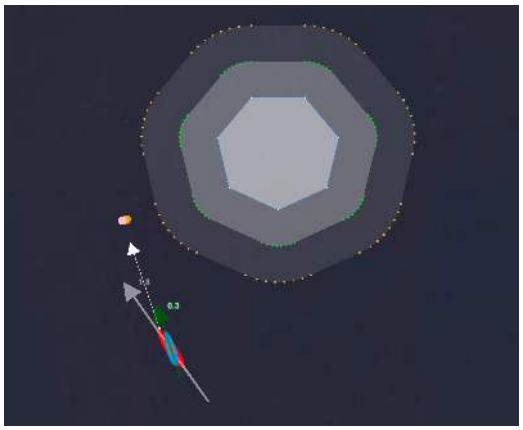
The **AvoidObstacle** behavior is a dynamically spawned behavior that works in conjunction with the obstacle manager

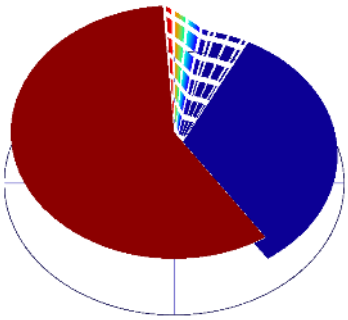
The **Obstacle Manager** is a separate MOOS App that interfaces with the sensor applications to maintain a list of known obstacles and locations.

Obstacle Avoidance

The **AvoidObstacle** behavior is a dynamically spawned behavior that works in conjunction with the obstacle manager

The **Obstacle Manager** is a separate MOOS App that interfaces with the sensor applications to maintain a list of known obstacles and locations.





Obstacle Avoidance

The **AvoidObstacle** behavior is a dynamically spawned behavior that works in conjunction with the obstacle manager

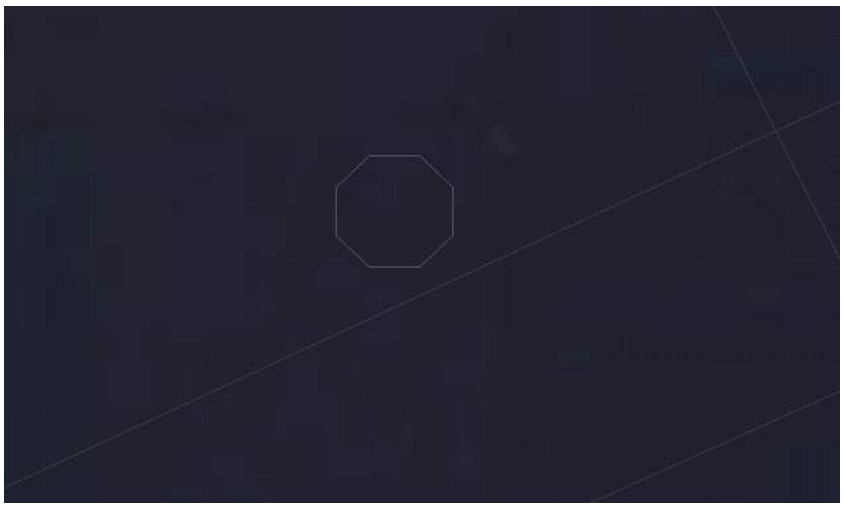
Obstacle Avoidance

The **AvoidObstacle** behavior is a dynamically spawned behavior that works in conjunction with the obstacle manager

41

Obstacle Avoidance

The **AvoidObstacle** behavior is a dynamically spawned behavior that works in conjunction with the obstacle manager



Behaviors Overview | Waypoint Behavior | LegRun Behavior | FixedTurn FullStop | Obstacle Avoidance | Dynamic Updates | Loiter Behavior | Min/Max Depth | OpRegionV24 Behavior | StationKeep Behavior

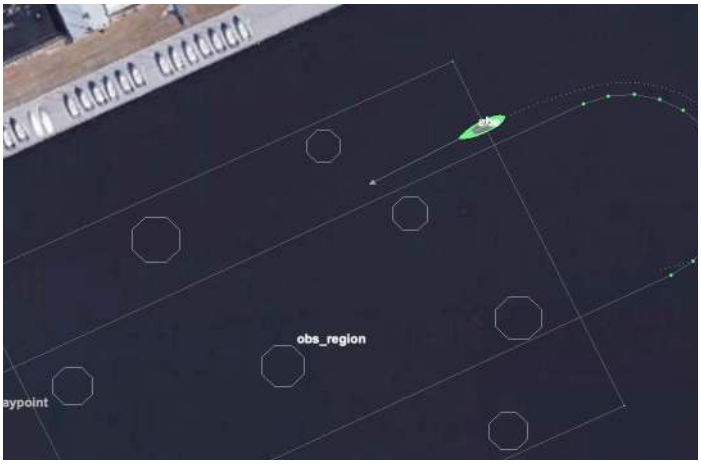
Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

42

Obstacle Avoidance

The **AvoidObstacle** behavior is a dynamically spawned behavior that works in conjunction with the obstacle manager



Dynamic Behavior Updates with the `updates` Parameter

Behavior Parameters

- Certain parameters are *specific to a particular behavior*. Waypoint behavior has:
  - points
  - capture_radius
  - slip_radius
  - capture_line
  - order
  - lead
  - lead_damper
  - lead_to_start
  - wptflag
  - cycleflag
  - point
  - lead_condition
  - repeat
  - xpoints
  - speed
  - wpt_flag_on_start

45

Behavior Parameters

- Certain parameters are *specific to a particular behavior*. Waypoint behavior has:
  - points
  - capture_radius
  - slip_radius
  - capture_line
  - order
  - lead
  - lead_damper
  - lead_to_start
  - wptflag
  - cycleflag
  - point
  - lead_condition
  - repeat
  - xpoints
  - speed
  - wpt_flag_on_start

- Certain parameters are *common to all behaviors*, for example:
  - name: A unique name – no two behavior instances can have the same name
  - priority: priority weight
  - condition: logic condition determining run state
  - endflag: posted when the behavior completes
  - idleflag: posted when the behavior enters the idle state
  - runflag: posted when the behavior enters the running state
  - runxflag: posted when the behavior is in the running state
  - activeflag: posted when the behavior enters the active state
  - inactiveflag: posted when the behavior enters the inactive state
  - spawnflag: posted when the behavior is first spawned

46

The updates Parameter

- The updates parameter names MOOS variable
- The helm will subscribe for the variable on behalf of the behavior
- Mail to this variable can change parameters originally configured for this behavior

Behavior launched with:

```
name      =  foobar
param     =  100
updates   =  WPT_UPDATE
```

MOOS mail received:

```
WPT_UPDATE = "param=50"
```

Behavior now configured:

```
name      =  foobar
param     =  50
updates   =  WPT_UPDATE
```


Alpha Mission Example

In-Mission Speed Changes with updates

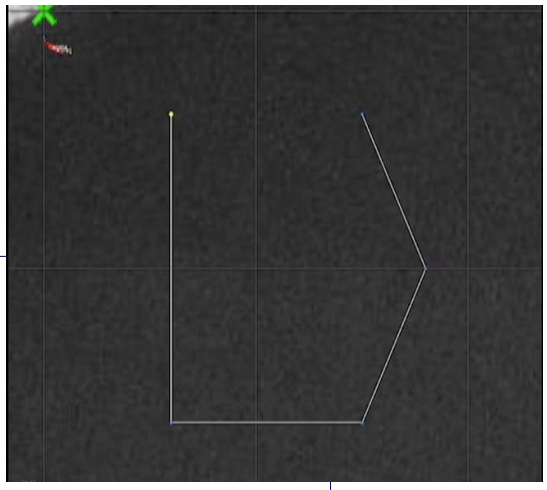
- The updates parameter used in the Alpha Mission
- Modify the transit speed
- Initially 4.0 meters / second
- Change to 1.0 m/s after launch

```
name       =  waypoint_survey
priority   =  100
condition  =  RETURN=false
condition  =  DEPLOY=true
endflag    =  RETURN=true
speed      =  4.0
updates    =  WPT_UPDATES
polygon    =  60,-40 : 60,-160 : 150,-160 : 180,100 : 150,-40
```

Alpha Mission Example

In-Mission Speed Changes with `updates`

- The `updates` parameter used in the Alpha Mission
- Modify the transit speed
- Initially 4.0 meters / second
- Change to 1.0 m/s after launch



```
name      =  waypoint_survey
priority  =  100
condition =  RETURN=false
condition =  DEPLOY=true
endflag   =  RETURN=true
speed     =  4.0
updates   =  WPT_UPDATES
polygon   =  60,-40 : 60,-160 : 150,-160 : 180,100 : 150,-40
```

Behaviors Overview | Waypoint Behavior | LegRun Behavior | FixedTurn FullStop | Obstacle Avoidance | **Dynamic Updates** | Loiter Behavior | Min/Max Depth | OpRegionV24 Behavior | StationKeep Behavior

Alpha Mission Example

In-Mission Reverse with `updates`

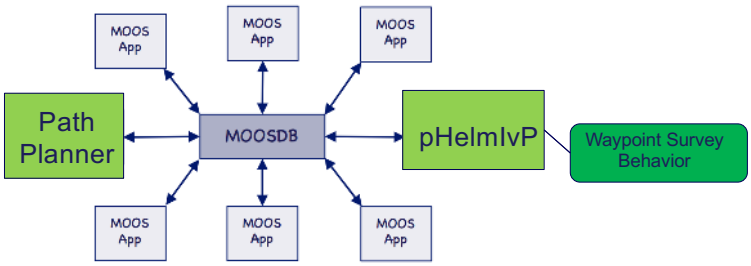
- After traversing the waypoints once, the `cycleflag` is published
- The `cycleflag` publishes to the `updates` variable, reversing the pattern direction for the second cycle.


```
name      =  waypoint_survey
priority  =  100
condition =  RETURN=false
condition =  DEPLOY=true
endflag   =  RETURN=true
speed     =  4.0
cycleflag =  WPT_UPDATES=order=reverse
updates   =  WPT_UPDATES
polygon   =  60,-40 : 60,-160 : 150,-160 : 180,100 : 150,-40
```

Behaviors Overview | Waypoint Behavior | LegRun Behavior | FixedTurn FullStop | Obstacle Avoidance | **Dynamic Updates** | Loiter Behavior | Min/Max Depth | OpRegionV24 Behavior | StationKeep Behavior

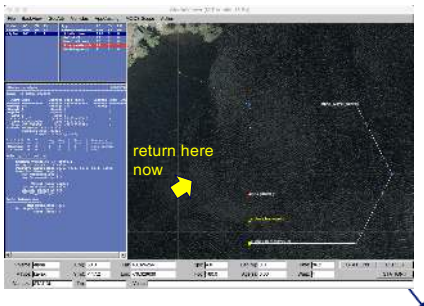
Behavior Updates for Path Planning

- Path planning MOOS App generates waypoints
- Behavior receives new waypoints through the updates

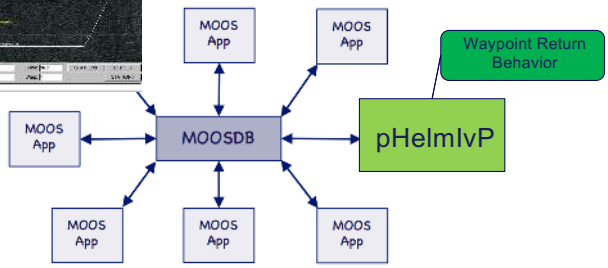


Behavior Updates for Command and Control

- User command and control GUI accept return point by mouse click
- GUI posts return point to variable set in the waypoint updates parameter

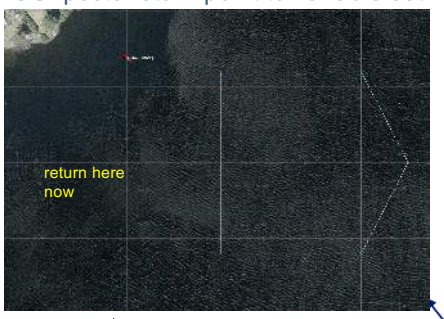


Command and Control GUI



Behavior Updates for Command and Control

- User command and control GUI accept return point by mouse click
- GUI posts return point to variable set in the waypoint updates parameter



Command and Control GUI

pMarineViewer posts:

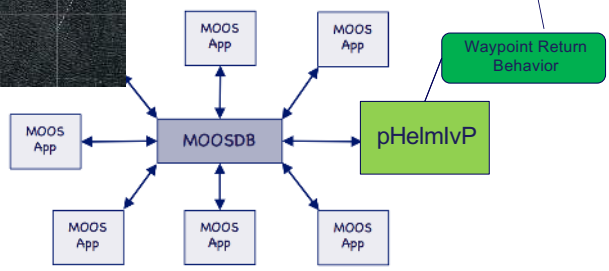
`RPT = point=20,-43`

MOOS Variable

Waypoint Behavior configuration parameter

`updates = RPT`

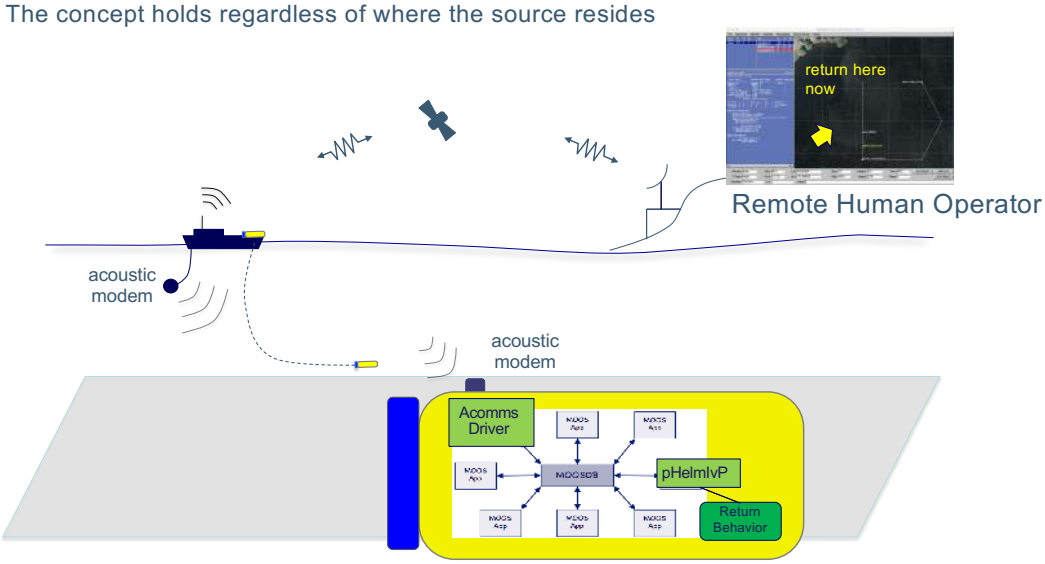
configured



53

Remote Command and Control

The concept holds regardless of where the source resides



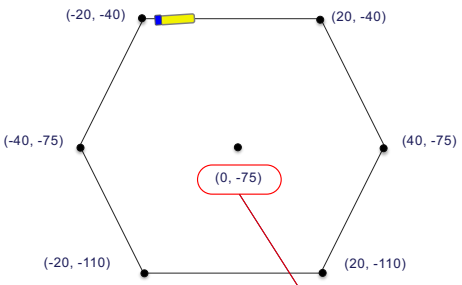
54

The Loiter Behavior

55

The Loiter Behavior

- Vehicle will traverse a loiter polygon, which can be any convex polygon
- Traversal in either clockwise or counter-clockwise direction, *indefinitely*

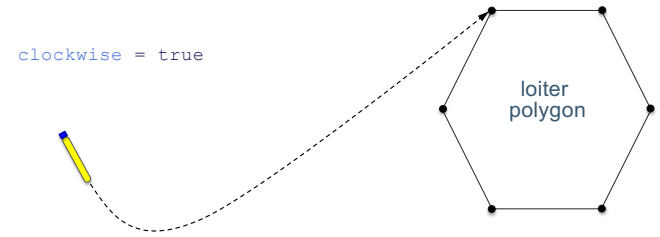



```
points    = polygon = format=radial, x=0, y=-75, radius=40, pts=6
clockwise = true
```


56

The Loiter Behavior Entry

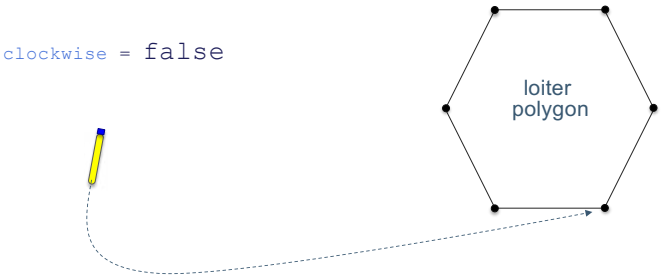
- Loiter direction depends on how the `clockwise` parameter is set
- The most appropriate initial vertex is chosen automatically for entry




```
points    = polygon = format=radial, x=0, y=-75, radius=40, pts=6
clockwise = true
```


The Loiter Behavior Entry

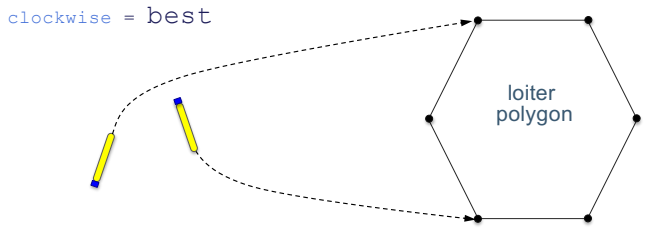
- Loiter direction depends on how the `clockwise` parameter is set
- The most appropriate initial vertex is chosen automatically for entry




```
points    = polygon = format=radial, x=0, y=-75, radius=40, pts=6
clockwise = false
```

The Loiter Behavior Entry

- When `clockwise` parameter is set to `best`, direction chosen automatically
- UUV position and orientation when behavior begins to run will determine direction

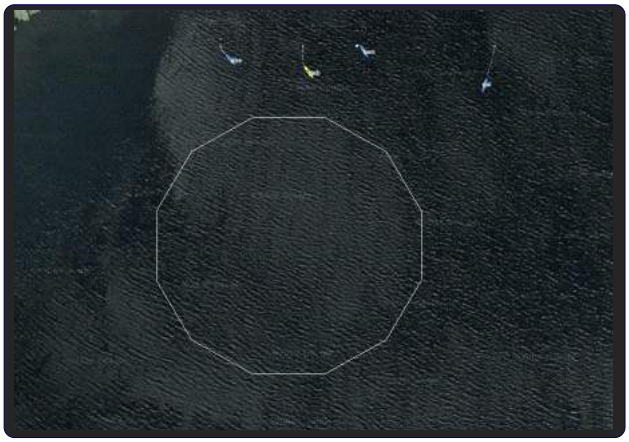



```
points    = polygon = format=radial, x=0, y=-75, radius=40, pts=6
clockwise = best
```


59

Multi-Vehicle Loiter Example

- Note robustness on entry angle
- collision avoidance makes entry non-trivial

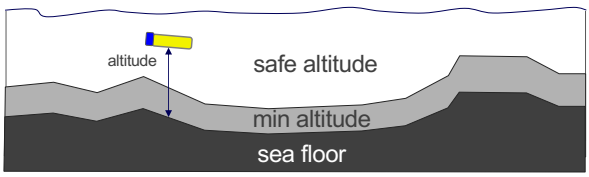


60

The MinAltitude and MaxDepth Behaviors

The MinAltitude Behavior

Disallow depths below specified altitude to the sea floor

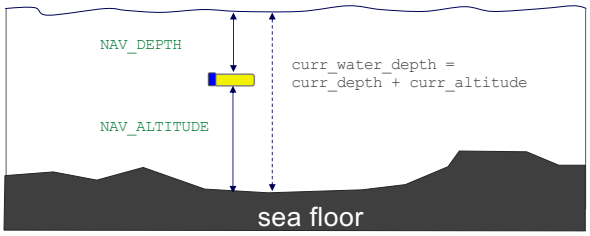


- The `min_altitude` parameter specifies a minimum distance to the sea floor that commanded depths must have
- The `missing_altitude_critical` parameter determines if a missing or stale altitude measurement is cause for halting the vehicle (and coming to the surface). The default is true.

```
min_altitude = 20
missing_altitude_critical = true
```

Determining The MinAltitude Depth

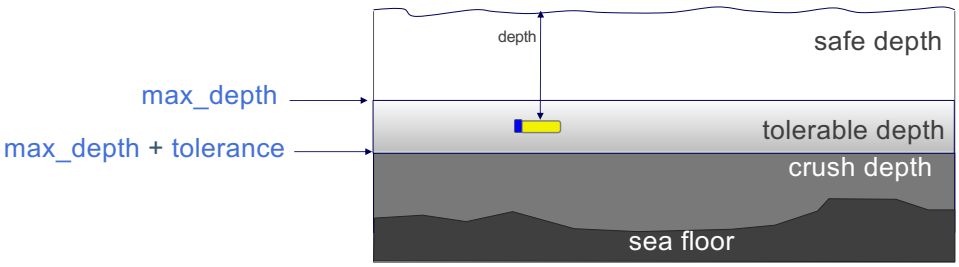
- The UUV has two sensors for (a) depth and (b) altitude
- These are published in the MOOS variables: `NAV_DEPTH` and `NAV_ALTITUDE`



- The current allowed maximum depth is: (curr_water_depth – min_altitude_depth)
- The behavior produces an objective function solely over the depth decision variable.

The MaxDepth Behavior

Disallow depths deeper than a specified max_depth + tolerance
Discourage depths within the tolerance

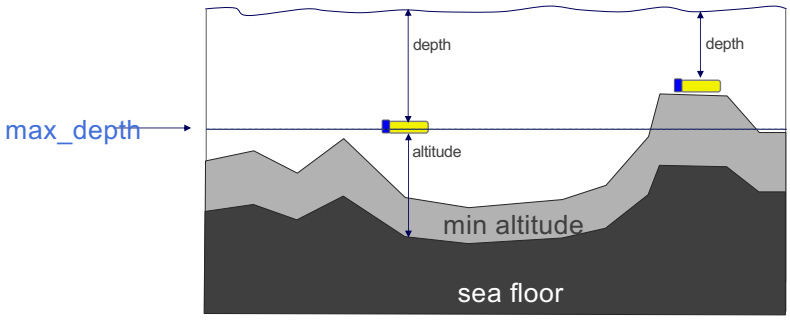


- The `max_depth` parameter is the maximum allowed depth.
- The `tolerance` parameter is a tolerable but discouraged depth below `max_depth`. The default is 0.

```
max_depth = 200
tolerance = 40
```

The MinAltitude and MaxDepth Behaviors Combined

- The two behaviors can be used in combination, each producing a depth objective function.
- The IvP solver will resolve the two limits influences on depth.

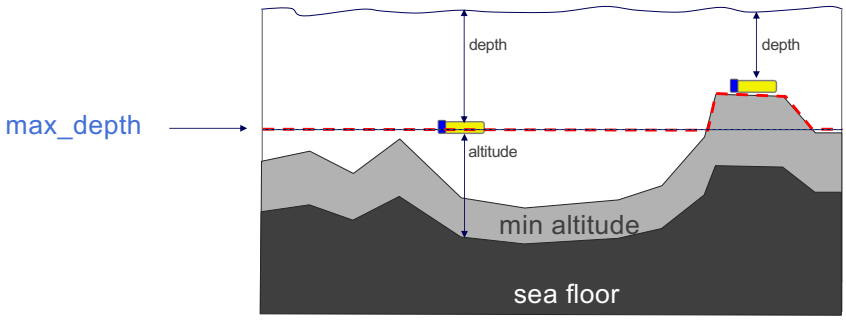


65

The MinAltitude and MaxDepth Behaviors Combined

- The two behaviors can be used in combination, each producing a depth objective function.
- The IvP solver will resolve the two limits influences on depth.

Combined allowed depth



66

The OpRegionV24 Behavior

(the “V24” indicates new in 2024 release)

67

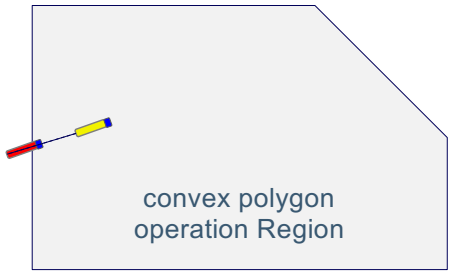
The OpRegionV24 Behavior

OpRegionV24 behavior provides four different types of safety functionality:

- a boundary box given by a convex polygon in the x-y or lat-lon plane
- an overall timeout
- a depth limit
- an altitude limit

Pure Safety/Shutdown Mode

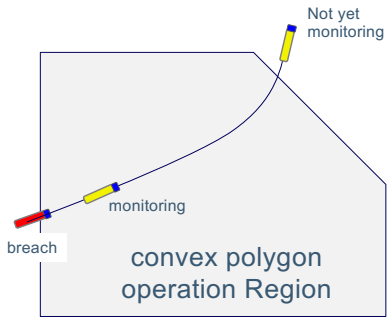
- The behavior does not produce an objective function to influence the vehicle to avoid violating these safety constraints.
- This behavior merely monitors the constraints and posts an error which results in the posting of all-stop commands,



68

OpRegionV24 Polygon Containment

- The OpRegionV24 behavior can specify a convex polygon indicating the allowable area of operation for the vehicle
  - Monitoring is not active until the vehicle enters the polygon
  - **trigger_entry_time** is the time (secs) within the polygon before monitoring becomes active
  - **trigger_exit_time** is the time (secs) outside the polygon before alarm is triggered
  - **breached_poly_flag** is a MOOS variable and value to be posted when/if the vehicle exits the polygon region.




```
polygon = 0,-50:0,-150:150,-150:150,-50
trigger_entry_time = 1
trigger_exit_time  = 1
breached_poly_flag = COME_TO_SURFACE = true
```


OpRegionV24: Maximum Mission Time

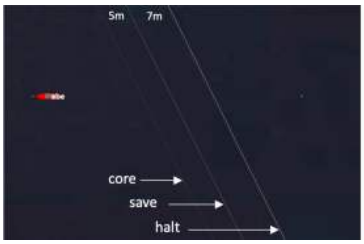
The OpRegionV24 behavior can specify a convex max_time indicating the total allowable mission time.

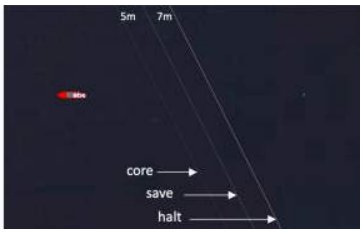
- max_time is the time (secs) after which an alarm is posted
- breached_time_flag is a MOOS variable and value to be posted when/if the vehicle times out
- The time begins when the helm is launched

```
max_time  = 3600
breached_time_flag = MAX_TIME_ALERT = true
```

OpRegionV24: Recover Before Too Late

The OpRegionV24 behavior can be configured to try to recover before violating op-area bounds.

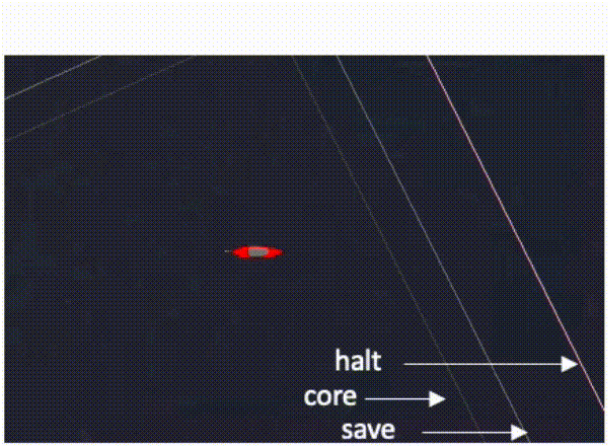




71

OpRegionV24: Recover Before Too Late

The OpRegionV24 behavior Recovery objective function

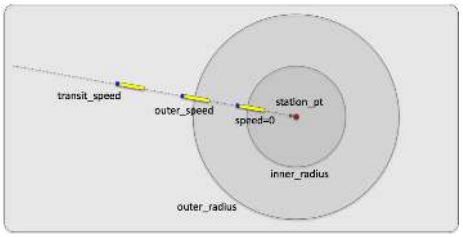


72

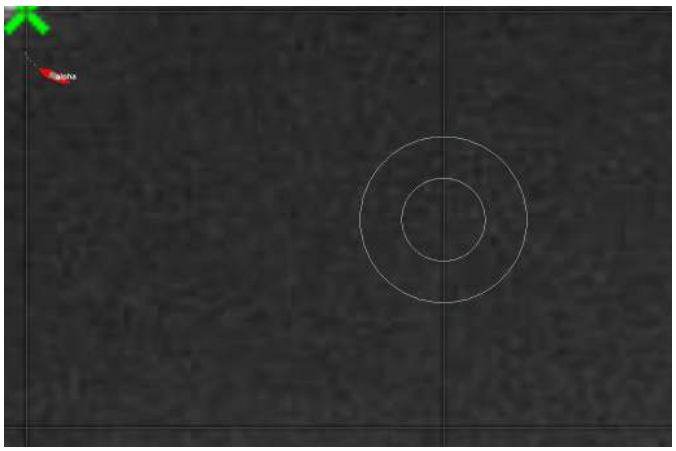
The StationKeep Behavior

The StationKeep Behavior

- **StationKeep behavior** keeps a vehicle on station defined by a point
- It can be set to continuously adjust
- It can be set to periodically adjust while drifting during inactivity (low-power mode)




```
station_pt   = 150,-50
inner_radius = 10
outer_radius = 30
transit_speed = 10
outer_speed   = 30
```



Dynamic Activation

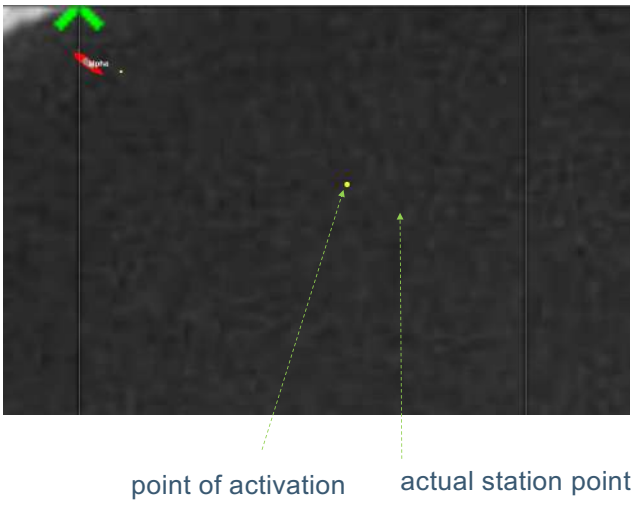
- When center_activate is set to true, the behavior will station keep at the point of activation.
- Notice that the vehicle momentum carries beyond the station keep point.

```
center_activate = true
inner_radius    = 10
outer_radius    = 30
transit_speed   = 10
outer_speed     = 30
```


75

Dynamic Activation

- When center_activate is set to true, the behavior will station keep at the point of activation.
- Notice that the vehicle momentum carries beyond the station keep point
- The swing_time parameter is the number of seconds after activation that the station point is marked



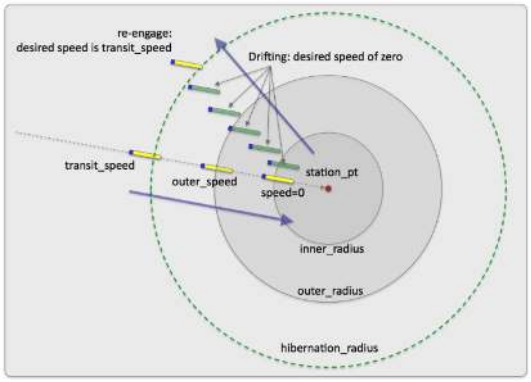

```
center_activate = true
swing time      = 10
inner_radius    = 10
outer_radius    = 30
transit_speed   = 10
outer_speed     = 30
```

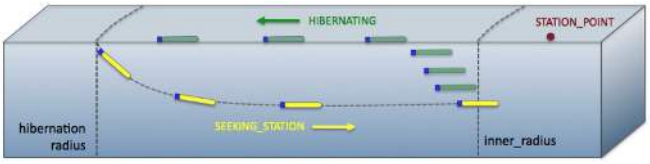

76

Low Power Station Keeping

- The hybernation_radius is a distance within which no corrective position keeping is used
- It may allow for long periods with no thrust

```
center_activate = true
hybernation_radius = 100
inner_radius  = 10
outer_radius  = 30
transit_speed = 10
outer_speed   = 30
```





END