11 Behaviors of the IvP Helm

The following is a description of some single-vehicle behaviors currently written for the IvP Helm. The division of single-vehicle behaviors and multi-vehicle behaviors (next section) is somewhat arbitrary. Other behavior modules exist that may be either in a testing state or too specific to a project for discussion here. The below description is for the person who wants to *use* current behaviors in the toolbox. The topic of how to add a new behavior is not covered here.

A behavior has a standard parameters defined at the `IvPBehavior` level as well as unique parameters defined at the subclass level. Parameters are set in the behavior file. For a behavior user, the setting of parameters is the primary venue for affecting the overall autonomy behavior in a vehicle. Parameters may also be dynamically altered once the mission has commenced. A parameter is set with a single line of the form:

```
parameter = value
```

The left-hand side, the parameter component, is case insensitive, while the value component is typically case sensitive. When the helm is launched, each behavior is created and the parameters are set. If a parameter setting in the behavior file references an unknown parameter, or if the value component fails a syntactic or semantic test, the line is noted and the helm ceases to launch.

11.1 BHV_Waypoint

11.1.1 Overview of the BHV_Waypoint Behavior

The `BHV_Waypoint` behavior is used for transiting to a set of specified waypoint in the x-y plane. The primary parameter is the set of waypoints. Other key parameters are the inner and outer radius around each waypoint that determine what it means to have met the conditions for moving on to the next waypoint. The basic idea is shown in Figure 35.

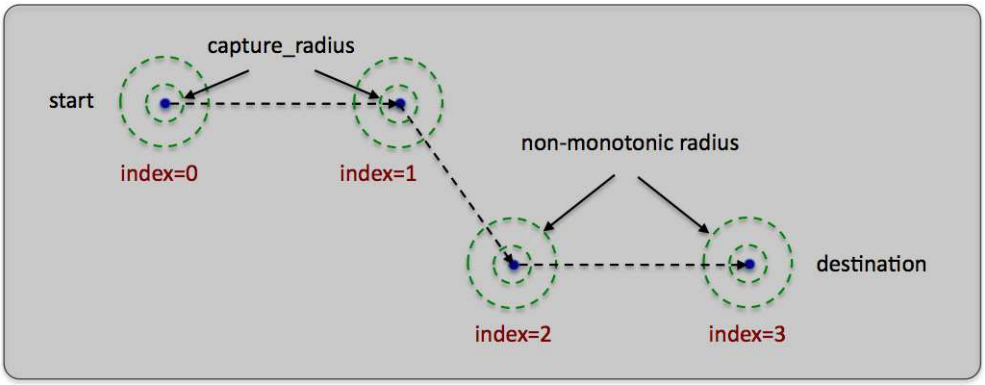

Figure 35: **The `BHV_Waypoint` behavior:** The waypoint behavior basic purpose is to traverse a set of waypoints. A capture radius is specified to define what it means to have achieved a waypoint, and a non-monotonic radius is specified to define what it means to be "close enough" should progress toward the waypoint be noted to degrade.

The behavior may also be configured to perform a degree of track-line following, that is, steering the vehicle not necessarily toward the next waypoint, but to a point on the line between the previous

and next waypoint. This is to ensure the vehicle stays closer to this line in the face of external forces such as wind or current. The behavior may also be set to "repeat" the set of waypoints indefinitely, or a fixed number of times. The waypoints may be specified either directly at start-up, or supplied dynamically during operation of the vehicle. There are also a number of accepted geometry patterns that may be given in lieu of specific waypoints, such as polygons, lawnmower pattern and so on.

11.1.2 Brief Summary of the BHV_Waypoint Behavior Parameters

The following parameters are defined for this behavior. A more detailed description is provided other parts of this section, and in Table 11 on page 147.

POINTS:	A colon separated list of x,y pairs given as points in 2D space, in meters.
POLYGON:	An alias for POINTS.
SPEED:	The desired speed (m/s) at which the vehicle travels through the points.
CAPTURE_RADIUS:	The radius tolerance, in meters, for satisfying the arrival at a waypoint.
RADIUS:	An alias for CAPTURE_RADIUS.
NM_RADIUS:	An "outer" capture radius. Arrival declared when the vehicle is in this range and the distance to the next waypoint begins to increase.
ORDER:	The order in which waypoints are traversed - "normal", or "reverse".
LEAD:	If this parameter is set, track-line following between waypoints is enabled.
LEAD_DAMPER:	Distance from trackline within which the lead distance is stretched out.
REPEAT:	The number of *extra* times traversed through the waypoints.
WPT_STATUS_VAR:	The MOOS variable posting a status report. The default is WPT_STAT.
WPT_INDEX_VAR:	The MOOS variable posting the index of the behavior's next waypoint.
CYCLE_FLAGS:	MOOS variable-value pairs posted at end of each cycle through waypoints.
CYCLE_INDEX_VAR:	The MOOS variable posting # of times cycled through the waypoints.
POST_SUFFIX:	A suffix tagged onto the WPT_STATUS, WPT_INDEX and CYCLE_INDEX variables.

11.1.3 Specifying Waypoints - the points, order, and repeat Parameters

The waypoints may be specified explicitly as a colon-separated list of comma-separate pairs, or implicitly using a geometric description. The order of the parameters may also be reversed with the order parameter. An example specification:

```
points    = 60,-40:60,-160:150,-160:180,-100:150,-40
order     = reverse  // default is "normal"
repeat    = 3        // default is 0
```

A waypoint behavior with this specification will traverse the five points in reverse order (150, −40 first) four times (one initial cycle and then repeated three times) before completing. If there is a syntactic error in this specification at helm start-up, an output error will be generated and the helm will not continue to launch. If the syntactic error is passed as part of a dynamic update (see

Section 7.2.2), the change in waypoints will be ignored and the a warning posted to the `BHV_WARNING` variable. See Section 9 for more methods for specifying sets of waypoints.

11.1.4 The `capture_radius` and `nonmonotonic_radius` Parameters

The `capture_radius` parameter specifies the distance to a given waypoint the vehicle must be before it is considered to have arrived at or achieved that waypoint. It is the inner radius around the points in Figure 35. The non-monotonic radius or `nm_radius` parameter specifies an alternative criteria for achieving a waypoint.

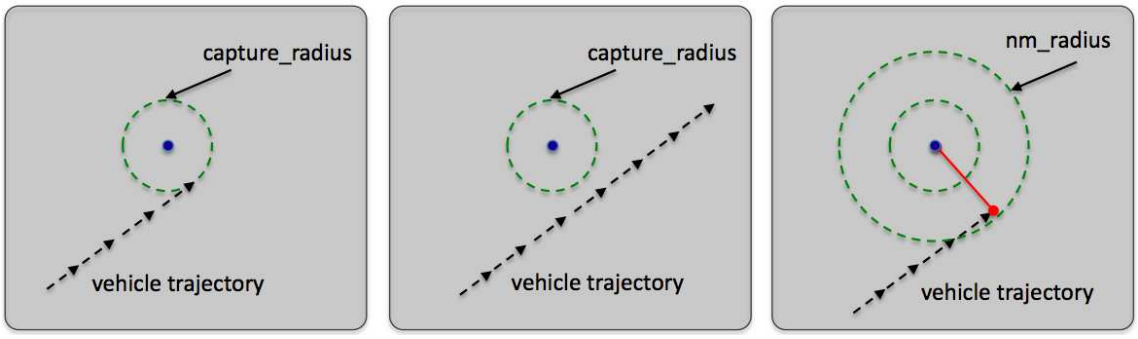

Figure 36: **The capture radius and non-monotonic radius:** (a) a successful waypoint arrival by achieving proximity less than the capture radius. (b) a missed waypoint likely resulting in the vehicle looping back to try again. (c) a missed waypoint but arrival declared anyway when the distance to the waypoint begins to increase and the vehicle is within the non-monotonic radius.

As the vehicle progresses toward a waypoint, the sequence of measured distances to the waypoint decreases monotonically. The sequence becomes non-monotonic when it hits its waypoint or when there is a near-miss of the waypoint capture radius. The `nm_radius`, is a capture radius distance within which a detection of increasing distances to the waypoint is interpreted as a waypoint arrival. This distance would have to be larger than the capture radius to have any effect. As a rule of thumb, a distance of twice the capture radius is practical. The idea is shown in Figure 36. The behavior keeps a running tally of hits achieved with the capture radius and those achieved with the non-monotonic radius. These tallies are reported in a status message described in Section 11.1.6 below.

11.1.5 Track-line Following using the `lead` Parameter

By default the waypoint behavior will output a preference for the heading that is directly toward the next waypoint. By setting the `lead` parameter, the behavior will instead output a preference for the heading that keeps the vehicle closer to the track-line, or the line between the previous waypoint and the waypoint currently being driven to.

The distance specified by the `lead` parameter is based on the perpendicular intersection point on the track-line. This is the point that would make a perpendicular line to the track-line if the other point determining the perpendicular line were the current position of the vehicle. The distance specified by the `lead` parameter is the distance from the perpendicular intersection point toward

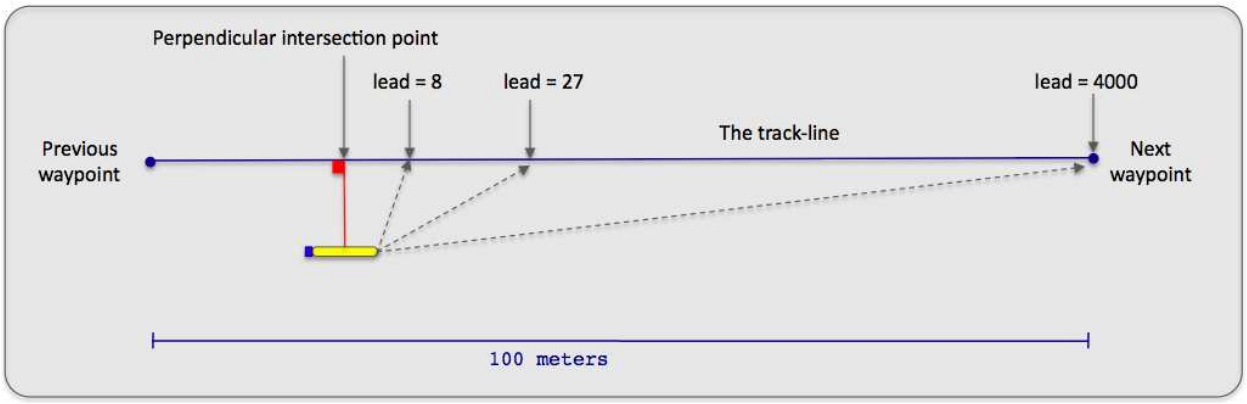

Figure 37: **The track-line mode:** When in track line mode, the vehicle steers toward a point o the track line rather than simply toward the next waypoint. The steering-point is determined by the `lead` parameter. This is the distance from the perpendicular intersection point toward the next waypoint.

the next waypoint, and defines an imaginary point on the track-line. The behavior outputs a heading preference based on this imaginary steering point. If the lead distance is greater than the distance to the next waypoint along the track-line, the imaginary steering point is simply the next waypoint.

If the `lead` parameter is enabled, it may be optionally used in conjuction with the `lead_damper` parameter. This parameter expresses a distance from the trackline in meters. When the vehicle is within this distance, the value of the `lead` parameter is stretched out toward the next waypoint to soften, or dampen, the approach to the trackline and reduce overshooting the trackline.

11.1.6 Variables Published by the BHV_Waypoint Behavior

The waypoint behavior publishes five variables for monitoring the performance of the behavior as it progresses: `WPT_STATUS`, `WPT_INDEX`, `CYCLE_INDEX`, `VIEW_POINT`, `VIEW_SEGLIST`. The `WPT_STATUS` contains information identifying the vehicle, the index of the current waypoint, the distance to the current waypoint, and the estimated time of arrival to the current waypoint. Example output:

```
WPT_STAT = "vname=alpha,behavior=traverse1,index=0,dist=43,eta=23"
```

The `WPT_INDEX` variable simply publishes the index of the current waypoint. This is a bit redundant, but this variable is logged as a numerical variable, not a string, and facilitates the plotting of the index value as a step function in post mission analysis tools. The `CYCLE_INDEX` variable publishes the number of times the behavior has traversed the entire set of waypoints. The behavior may be configured to post the information in these three variables using alternative variables of the user's liking, or suppress it completely:

```
wpt_status_var  = MY_WPT_STATUS_VAR      // The default is "WPT_STAT"
wpt_index_var   = MY_WPT_INDEX_VAR       // The default is "WPT_INDEX"
cycle_index_var = MY_CYCLE_INDEX_VAR     // The default is "CYCLE_INDEX"
```

or, to suppress the reports:

```
wpt_status_var  = silent  // both case insensitive
wpt_index_var   = silent
cycle_index_var = silent
```

Further posts to the MOOSDB can be configured to be made at the end of each cycle, that is, after reaching the last waypoint. Normally, if the `repeat` parameter remains at its default value of zero, then the end of a cycle and completing are identical and endflags can be used to post the desired information. However, when the behavior is configured to repeat the set of waypoints one or more times before completed, the `cycleflags` parameter may be used to post one or more variable-value pairs at the end of each cycle. Likewise, if the `repeat` parameter is zero, but the behavior is set with `perpetual=true`, the cycle flags will posted each new time that the behavior completes.

The `VIEW_POINT` and `VIEW_SEGLIST` variables provide information consumable by a GUI application such as `pMarineViewer` for rendering the set of waypoints traversed by the behavior (`VIEW_SEGLIST`) and the behavior's next waypoint (`VIEW_POINT`). These two variables are responsible for the visual output in the Alpha Example Mission in Section 4 in Figure 6 on page 32.

11.1.7 The Objective Function Produced by the BHV_Waypoint Behavior

The waypoint behavior produces a new objective function, at each iteration, over the variables `speed` and `course/heading`. The behavior can be configured to generate this objective function in one of two forms, either by coupling two independent one-variable functions, or by generating a single coupled function directly.

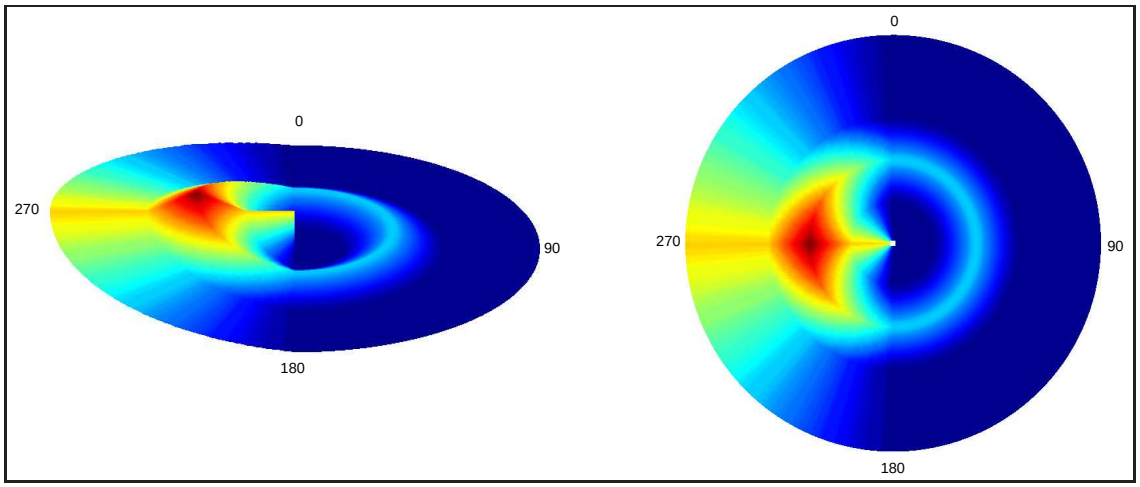

Figure 38: **A waypoint objective function:** The objective function produced by the waypoint behavior is defined over possible heading and speed values. Depicted here is an objective function favoring maneuvers to a waypoint 270 degrees from the current vehicle position and favoring speeds closer to the mid-range of capable vehicle speeds. Higher speeds are represented farther radially out from the center.

11.2 BHV_OpRegion

11.2.1 Overview of the BHV_OpRegion Behavior

This behavior provides four different types of safety functionality, (a) a boundary box given by a convex polygon in the x-y or lat-lon plane, (b) an overall timeout, (c) a depth limit, (d) an altitude limit.

11.2.2 Brief Summary of the BHV_OpRegion Parameters

The following parameters are defined for this behavior. A more detailed description is provided other parts of this section, and in Table 12 on page 148.

POLYGON:	The lat-lon area the vehicle is restricted to stay within. Section 11.2.3.
TRIGGER_ENTRY_TIME:	The time required for the vehicle to have been within the polygon region before triggering the polygon requirement. Section 11.2.3.
TRIGGER_EXIT_TIME:	The time required to have been outside the polygon before declaring a polygon containment failure. Section 11.2.3.
MAX_TIME:	The max allowable time in seconds. Section 11.2.4.
MAX_DEPTH:	The max allowable depth in meters. Section 11.2.5.
MIN_ALTITUDE:	The min allowable altitude in meters. Section 11.2.6.

11.2.3 Safety Checking Applied to an Operation Region

One safety check performed by the OpRegion behavior is to ensure that the vehicle remains in an operation region defined by a convex polygon in the x-y plane.

POLYGON: A colon separated list of x,y pairs given as points in space, typically meters. A pair given by "label,string" can associate an optional label with the point list. *The collection of points must be a convex polygon.* A check for convexity is done upon helm/behavior start-up. Behavior initialization will fail if it is not convex. If no polygon is provided, no X,Y checks are made.

TRIGGER_ENTRY_TIME: The amount of time required for the vehicle to have been within the polygon containment region before triggering the polygon containment requirement. This is useful when launching vehicles from a dock structure such as the MIT Sailing Pavilion. The default setting is zero meaning the polygon containment requirement is active immediately.

TRIGGER_EXIT_TIME: The amount of time required to have been outside the polygon containment region before declaring a polygon containment failure. This is useful if the vehicle NAV_X and NAV_Y position is based on a sensor without outlier detection. The kayaks, for example, are often relying solely on GPS which occasionally emits an outlier well out of the containment region. By setting this value high enough, outliers are ignored. Each time a recorded position is contained within the polygon region, the clock is set to zero. The default setting is zero, meaning the very first detection outside the polygon will result in a polygon containment error.

11.2.4 Safety Checking Applied to a Maximum Mission Operation Time

`MAX_TIME:` The maximum allowable time (in seconds) that the helm is allowed run. The clock starts when the pHelmIvP process first takes control.

11.2.5 Safety Checking Applied to a Maximum Vehicle Depth

`MAX_DEPTH:` The maximum allowable depth of the vehicle (in meters). If no depth is provided, no depth checks are made.

11.2.6 Safety Checking Applied to a Minimum Vehicle Altitude

`MIN_ALTITUDE:` The minimum allowable altitude of the vehicle (in meters). If no depth is provided, no depth checks are made.

11.2.7 Variables Published by the BHV_OpRegion Behavior

The behavior also produces a set of status variables regarding the vehicle position with respect to the containment region. Since a violation of this constraint results in a vehicle full-stop and the helm relinquishing control, other behaviors or MOOS processes may want to take measures to avoid it. These status variables provide information on the position and estimated time between the vehicle and the perimeter, based both on the absolute position as well as the current vehicle trajectory. See Figure 39.

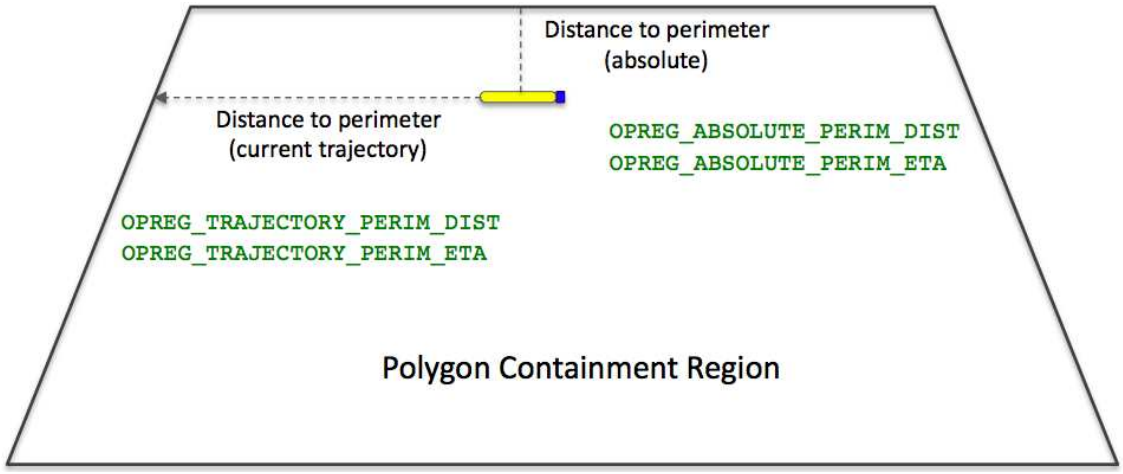

Figure 39: **The OpRegion polygon and status variables:** The OpRegion behavior publishes information regarding its estimated distance and time of arrival (ETA) to the perimeter of the polygon containment region. It publishes two sets of information; one based on the current trajectory and the other based on the absolute distance to the perimeter at top vehicle speed.

The four variables produced by the behavior (and posted to the MOOSDB by the Helm) are:

`OPREG_TRAJECTORY_PERIM_DIST:` The distance (in meters) between the current vehicle position to

the perimeter of the polygon containment region (given by the POLYGON parameter), based on the vehicle remaining on the current trajectory.

OPREG_TRAJECTORY_PERIM_ETA: The amount of time (in seconds) needed for the vehicle to reach the perimeter of the polygon containment region (given by the POLYGON parameter), based on the vehicle remaining on the current trajectory.

OPREG_ABSOLUTE_PERIM_DIST: The distance (in meters) between the current vehicle position to the perimeter of the polygon containment region (given by the POLYGON parameter), regardless of the current vehicle trajectory.

OPREG_ABSOLUTE_PERIM_ETA: The amount of time (in seconds) needed for the vehicle to reach the perimeter of the polygon containment region (given by the POLYGON parameter), regardless of the current vehicle trajectory. Calculated on the maximum vehicle speed.

11.3 BHV_Loiter

A behavior for transiting to and repeatedly traversing a set of waypoints. A similar effect can be achieved with the BHV_Waypoint behavior but this behavior assumes a set of waypoints forming a convex polygon to exploit certain useful algorithms discussed below. This behavior is comparable to the "Obit Task" of the older helm but is more general in that general convex polygons, not just those approximating circles, are allowed. It also utilizes the non-monotonic arrival criteria use in the BHV_Waypoint behavior to avoid loop-backs upon waypoint near-misses. It also robustly handles dynamic exit and re-entry modes when or if the vehicle diverges from the loiter region due to external events. And it is dynamically reconfigurable to allow a mission control module to repeatedly reassign the vehicle to different loiter regions by using a single persistent instance of the behavior. The following parameters are defined for this behavior:

`POLYGON:` A colon separated list of comma-separated x,y pairs indicating points in 2D space. Units are in in meters. Unlike the waypoint behavior, these points must describe a convex polygon; if the convexity condition fails the behavior will not instantiate. As an alternative to listing a sequence of points, a orbit-style polygon can be given by four values (1),(2) the x and y position, (3) the radius in meters, and (4) the number of points on the circle. This specification is denoted with the "radial" tag as follows "radial:50,50,200,16".

`SPEED:` The desired speed, in meters/second, at which the vehicle travels through the points.

`RADIUS:` The radius tolerance, in meters, for satisfying the arrival at a waypoint. As soon as the vehicle is within this distance to the waypoint the waypoint behavior begins operating on the next waypoint in the sequence, or completes and posts its endflags if there are no more waypoints.

`NM_RADIUS:` As the vehicle progresses toward a waypoint, the sequence of measured distances to the waypoint decreases monotonically. The sequence becomes non-monotonic when it hits its waypoint or when there is a near-miss of the waypoint arrival radius. The `NM_RADIUS`, short for *non-monotonic radius* is an arrival radius distance within which a detection of increasing distances to the waypoint is interpreted as a waypoint arrival. This distance would have to be larger than the arrival radius to have any effect (see Figure 36). As a rule of thumb, a distance of twice the arrival radius is practical.

`CLOCKWISE:` If "true", the behavior will influence the vehicle in a clockwise direction around the polygon. Values are case insensitive, but must spell either true or false. The default is true.

`ACQUIRE_DIST:` The distance in meters between the vehicle and the polygon that will trigger the vehicle to return to *acquire* mode. This notion applies to the case where the vehicle is both inside and outside the polygon. (The re-acquire algorithms are different however.)

`POST_SUFFIX:` This string will be added as a suffix to each of the status variables posted by the behavior (`LOITER_REPORT`, `LOITER_INDEX`, `LOITER_ACQUIRE`, `LOITER_DIST2POLY`). By default, the suffix is the empty string and the variables will be posted as above. When multiple Loiter behaviors are configured in the helm it may help to distinguish the posted variabes by a suffix. A given suffix of `"FOO"` would result in the posting of `LOITER_INDEX_FOO` for example. The extra `'_'` character is inserted automatically.

When the behavior is active, it is in either one of two modes; the *acquire* mode or *normal* mode. In the normal mode it is merely proceeding to the next waypoint on the polygon. In the acquire mode, each iteration begins by first determining the next polygon point to treat as the next waypoint. This is useful for ensuring the entry waypoint isn't followed by a need for a sharp vehicle turn. The acquire point depends on the chosen direction of polygon traversal, as shown in Figure 40.

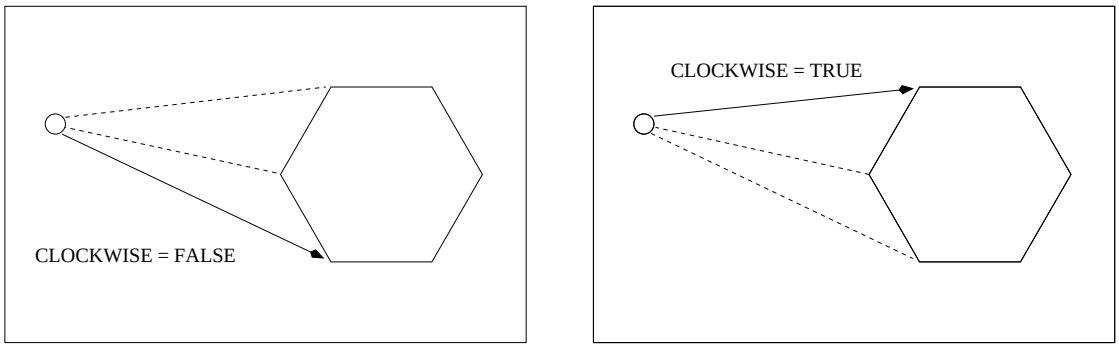

Figure 40: In the *acquire* mode, the polygon points are evaluated for suitability in terms of a smooth entry trajectory. Only the "viewable" points, those viewable if the polygon were an opaque object and the viewer were at the current vehicle location, are contenders. The contenders are rated on the follow-on angle given the desired clockwise or counter-clockwise loiter direction. Larger follow-on angles are preferred as shown.

When the behavior is in the acquire mode and *outside* the polygon, the chosen vertex is the one most tangential in either the clockwise or counter-clockwise direction as shown in the figure. When the vehicle is *inside* the polygon, the chosen vertex is the one which forms the most obtuse angle between the current vehicle position, the vertex, and the follow-on vertex. Unlike the case when outside the polygon, the chosen vertex changes as the vehicle makes progress back to the polygon perimeter. The effect is for the vehicle to "spiral" out to the perimeter for the smoothest re-entry in to a normal loitering path.

The circumstance most common for triggering the acquire mode is the initial assignment to the vehicle to loiter at a new given region in the X,Y plane. This assignment *could* occur while the vehicle happens to already be within the polygon for a number of reasons. Furthermore, the vehicle could be driven off the polygon loiter trajectory due to environmental (wind or current) forces or the temporary dominance of other vehicle behaviors such as collision avoidance or tracking of another vehicle.

Once the behavior enters the acquire mode, it remains in this mode until arriving at the first waypoint (defined by the arrival and non-monotonic radii settings), after which it switches to normal mode until the acquire mode is re-triggered or the behavior run conditions are no longer met. There is currently no "complete" condition for this behavior other than a time-out which is defined for all behaviors.

11.4 BHV_PeriodicSpeed

This behavior will periodically influence the speed of the vehicle while remaining neutral at other times. The timing is specified by a given period length in which the influence is on, and a gap length specifying the time between periods. It was conceived for use on an AUV equipped with an acoustic modem to periodically slow the vehicle to reduce self-noise and reduce communication difficulty. One can also specify a flag (a MOOS variable and value) to be posted at the start of the period to prompt an outside action such as the start of communication attempts. The following parameters are defined for this behavior:

`PERIOD_LENGTH:` The duration of the period, in seconds, during which the behavior will produce an objective function over the desired speed.

`PERIOD_GAP:` The duration of time in seconds between periods.

`PERIOD_FLAG:` A flag (MOOS variable) to be posted at the beginning of each active period. The argument is of the form `VAR=VAL`. If if no value is specified, the value will be the period index, incremented on each new period commencement.

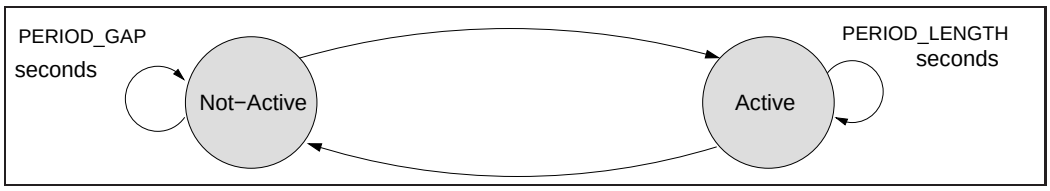

Figure 41: In active mode the behavior will produce an objective function defined over speed that will potentially influence the speed of the vehicle. In the inactive mode, it simply will not produce an objective function.

`STAT_PENDING_ACTIVE:` The number of seconds remaining until the behavior reaches the *active* state. By default this is empty and no status is posted by the behavior. To reduce posting volume, the value posted will be rounded to the nearest second until less than one second remains in which case fractions are posted.

`STAT_PENDING_INACTIVE:` The number of seconds remaining until the behavior reaches the *inactive* state. By default this is empty and no status is posted by the behavior. To reduce posting volume, the value posted will be rounded to the nearest second until less than one second remains in which case fractions are posted.

`PERIOD_SPEED:` The desired speed in meters per second.

`PERIOD_PEAKWIDTH:` The width of the peak in meters per second in the speed objective function.

`PERIOD_BASEWIDTH:` The width of the base, in meters per second in the speed objective function.

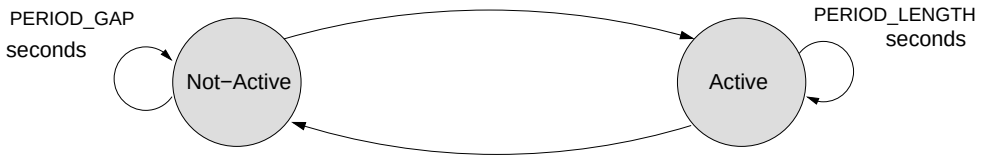

Figure 42: In (a) the preference is a for a particular speed and a slight tolerance in either direction. In (b) the preference is for a particular range of speeds with a slight tolerance either way. In (c) the preference is for anything less than a given speed with some tolerance for higher speeds. In (d) the preference is for anything greater than a given speed with a no tolerance for lower speeds.

11.5 BHV_PeriodicSurface

This behavior will periodically influence the depth and speed of the vehicle while remaining neutral at other times. The purpose is to bring the vehicle to the surface periodically to achieve some specified event specified by the user, typically the receipt of a GPS fix. Once this event is achieved, the behavior resets its internal clock to a given period length and will remain idle until a clock time-out occurs. The behavior can be in one of four states as described in Figure 43 below.

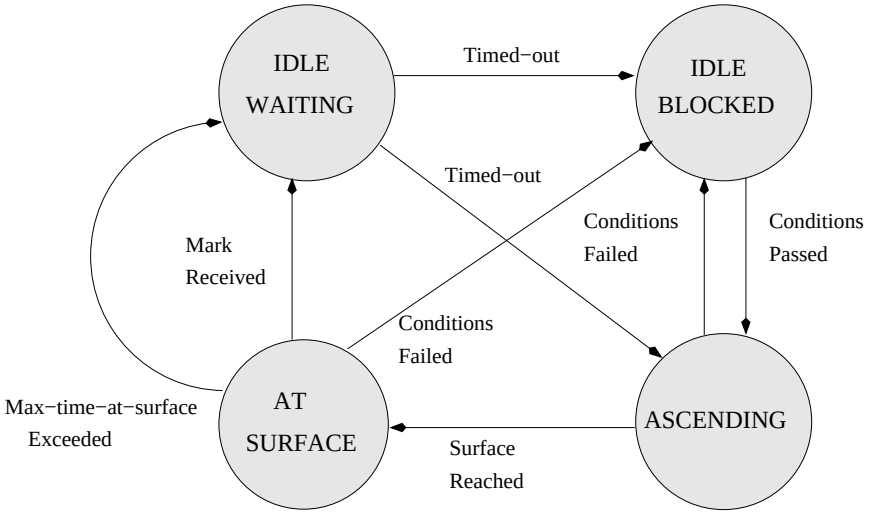

Figure 43: Possible modes of the PeriodicSurface behavior.

In the `IDLE_WAITING` state the behavior is simply waiting for its clock to wind down to zero. The duration is given by the `PERIOD` parameter listed below. The clock is active despite any other run conditions that may apply to the behavior. It is started when the behavior is first instantiated and also when the desired event occurs at the surface. The `IDLE_BLOCKED` state indicates that the behavior timer has reached zero, but another run condition has not been met. This is to prevent the behavior from trying to surface the vehicle when other circumstances override the need to surface. In the `ASCENDING` state, the behavior will produce an objective function over depth and speed to bring the vehicle to the surface. A couple parameters described below can determine the trajectory of the vehicle during ascent. This state can transition back to the `IDLE_BLOCKED` state if run conditions become no longer satisfied prior to the vehicle reaching the surface. In the `AT_SURFACE` state the vehicle is at the surface waiting for a specified event.

PERIOD: The duration of the period, in seconds, during which the behavior will remain in the IDLE_WAITING state.

MARK_VARIABLE: The name of a variable used for indicating when the behavior witnesses the event that would reset the period clock. On each iteration, the variable is checked against its last known value and if different, the clock is reset. The default value for this parameter is GPS_UPDATE_RECEIVED. If this variable is populated by another process with a value indicating the time a GPS fix is obtained, then the mark will occur on each GPS fix. Since the value of this argument names a MOOS variable, it is case sensitive.

PENDING_STATUS_VAR: This variable will be written to with the value of the remaining time on the idle clock, rounded to integer seconds. The default value is PENDING_SURFACE. Since the value of this argument names a MOOS variable, it is case sensitive.

ATSURFACE_STATUS_VAR: This variable will be written to with the number of seconds that the vehicle has been waiting at the surface (for the event indicated by the MARK_VARIABLE). The number of seconds is rounded to the nearest integer and will be zero when the vehicle is not at the surface. The default value is TIME_AT_SURFACE. Since the value of this argument names a MOOS variable, it is case sensitive.

ASCENT_SPEED: This parameter indicates the desired speed (m/s) of the vehicle during the ascent state. If left unspecified, the ascent speed will be equal to the current noted speed at moment it transitions into the ascent state.

ASCENT_GRADE: This parameter indicates the manner in which the ascent speed approaches zero as the vehicle progresses toward the ZERO_SPEED_DEPTH. It has four legal values: *fullspeed*, *linear*, *quadratic*, and *quasi*. In all four cases, the initial speed is determined by the parameter ASCENT_SPEED, and the desired speed will be zero once the ZERO_SPEED_DEPTH has been achieved. The four settings determine the manner of slowing to zero speed during the ascent. The *fullspeed* setting indicates that desired speed should remain constant through the ascent right up to the instant the vehicle achieves ZERO_SPEED_DEPTH. For the other three settings the speed reduction is relative to the starting depth (the depth noted at the outset of the ascent state) and the ZERO_SPEED_DEPTH. With the *linear* setting, the speed reduction is linear. With the *quadratic* setting, the speed reduction is quadratic (quicker initial speed reduction). With the *quasi* setting the speed reduction is between linear and quadratic. The value passed to this parameter is not case sensitive.

ZERO_SPEED_DEPTH: The depth (in meters) during the ascent state at which the desired speed becomes zero, and presumably further ascent is achieved through positive buoyancy.

MAX_TIME_AT_SURFACE: The maximum time (in seconds) spent in the AT_SURFACE state, waiting for the event indicated by the MARK_VARIABLE, before the behavior transitions into the IDLE state.

11.6 BHV_ConstantDepth

This behavior will drive the vehicle at a specified depth. Analogous to the ConstantDepthTask in the pHelm task library, but somewhat different. This behavior merely expresses a preference for a particular depth. If other behaviors also have a depth preference, coordination/compromise will take place through the multi-objective optimization process. The following parameters are defined for this behavior:

`DEPTH:` The desired depth in meters.

`PEAKWIDTH:` The width of the peak in meters in the produced objective function.

`BASEWIDTH:` The width of the base, in meters in the produced objective function.

`DURATION:` This is a parameter defined for all general behaviors, but for this behavior, specification is mandatory for safety reasons. The default if not specified is 0 seconds which will result in the behavior completing immediately. If no duration limit is desired, e.g., if the behavior is tied to another behavior or event via condition variables, then setting "duration = no-time-limit" will result in no time duration checks for this behavior.

11.7 BHV_ConstantHeading

This behavior will drive the vehicle at a specified depth. Analogous to the ConstantHeadingTask in the pHelm task library, but somewhat different. This behavior merely expresses a preference for a particular heading. If other behaviors also have a heading preference, coordination/compromise will take place through the multi-objective optimization process. The following parameters are defined for this behavior:

`HEADING:` The desired heading in degrees (-180, +180].

`PEAKWIDTH:` The width of the peak in degrees in the produced objective function.

`BASEWIDTH:` The width of the base, in degrees in the produced objective function.

`DURATION:` This is a parameter defined for all general behaviors, but for this behavior, specification is mandatory for safety reasons. The default if not specified is 0 seconds which will result in the behavior completing immediately. If no duration limit is desired, e.g., if the behavior is tied to another behavior or event via condition variables, then setting "duration = no-time-limit" will result in no time duration checks for this behavior.

11.8 BHV_ConstantSpeed

This behavior will drive the vehicle at a specified speed. Analogous to the ConstantSpeedTask in the pHelm task library, but somewhat different. This behavior merely expresses a preference for a particular speed. If other behaviors also have a speed preference, coordination/compromise will take place through the multi-objective optimization process. The following parameters are defined for this behavior:

`SPEED:` The desired speed in meters/second.

`PEAKWIDTH:` The width of the peak in meters/second in the produced objective function.

`BASEWIDTH:` The width of the base, in meters/second in the produced objective function.

`DURATION:` This is a parameter defined for all general behaviors, but for this behavior, specification is mandatory for safety reasons. The default if not specified is 0 seconds which will result in the behavior completing immediately. If no duration limit is desired, e.g., if the behavior is tied to another behavior or event via condition variables, then setting "duration = no-time-limit" will result in no time duration checks for this behavior.

11.9 BHV_GoToDepth

This behavior will drive the vehicle to a sequence of specified depths and duration at each depth. The duration is specified in seconds and reflects the time at depth *after* the vehicle has first achieved that depth, where achieving depth is defined by the CAPTURE_DELTA parameter. The behavior subscribes for NAV_DEPTH to examine the current vehicle depth against the target depth. If the current depth is within the delta given by CAPTURE_DELTA, that depth is considered to have been achieved. The behavior also stores the previous depth from the prior behavior iteration, and if the target depth is between the prior depth and current depth, the depth is considered to be achieved regardless of whether the prior or current depth is actually within the CAPTURE_DELTA. This behavior merely expresses a preference for a particular depth. If other behaviors also have a depth preference, coordination/compromise will take place through the multi-objective optimization process. The following parameters are defined for this behavior:

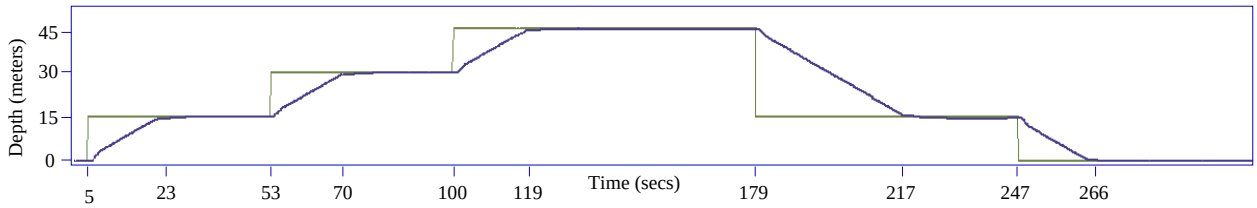

Figure 44: Depth log from simulation with the depth parameters shown in Listing 8. The lighter, step-like line indicates the values of DESIRED_DEPTH generated by the helm, and the darker line indicates the recorded depth value of the vehicle. The depth plateaus start from the moment the vehicle achieves depth. For example, the vehicle achieved a depth of 45 meters at 119 seconds and retained that desired depth for another 60 seconds as requested in the configuration shown in Listing 8.

DEPTH: A colon-separated list of comma-separated pairs. Each pair contains a desired depth and a duration at that depth. The duration applies from the point in time that the depth is first achieved. If a time duration is not provided for any pair, it defaults to zero. Thus "depth = 20" is a valid parameter setting.

REPEAT: The number of times the vehicle will traverse through the evolution of depths, proceeding to the 1st depth after the nth depth has been hit. The default value is zero.

PERPETUAL: If equal to *true*, when the vehicle completes its evolution of depths (perhaps several evolutions if REPEAT is non-zero), the endflags will be posted. But rather than setting the complete variable to true and thus never receiving any further run consideration, the behavior is reset to its initial state. Presumably the user sets endflags that will cause the condition flags to be not immediately satisfied, thus putting the behavior in a state waiting again for an external event flag to be posted. The default value of this parameter is *false*.

CAPTURE_DELTA: The delta depth, in meters, between the current observed depth and the current target depth, below which the behavior will declare the depth to have been achieved.

CAPTURE_FLAG: The name of a MOOS variable incremented each time a target depth level has been achieved. Useful for logfile debugging/analyzing and also allows other behaviors to be

conditioned on a depth event. If this behavior is completed in *perpetual* mode, the counter is reset to zero. If the behavior is repeating a set of depths by setting `REPEAT` greater than zero, the counter will continue to increment through evolutions.

11.10 BHV_MemoryTurnLimit

The objective of the Memory-Turn-Limit behavior is to avoid vehicle turns that may cross back on its own path and risk damage to the towed equipment. Its configuration is determined by the two parameters described below which combine to set a vehicle turn radius limit. However, it is not strictly described by a limited turn radius; it stores a time-stamped history of recent recorded headings and maintains a *heading average*, and forms its objective function on a range deviation from that average. This behavior merely expresses a preference for a particular heading. If other behaviors also have a heading preference, coordination/compromise will take place through the multi-objective optimization process. The following parameters are defined for this behavior:

`MEMORY_TIME`: The duration of time for which the heading history is maintained and heading average calculated.

`TURN_RANGE`: The range of heading values deviating from the current heading average outside of which the behavior reflects sharp penalty in its objective function.

The heading history is maintained locally in the behavior by storing the currently observed heading and keeping a queue of n recent headings within the `MEMORY_TIME` threshold. The heading average calculation below handles the issue of angle wrap in a set of n headings $h_0 \ldots h_{n-1}$ where each heading is in the range $[0, 359]$.

$$\text{heading\_avg} = \text{atan2}(s, c) \cdot 180/\pi,$$

where s and c are given by:

$$s = \sum_{k=0}^{n-1} \sin\left(h_k \pi/180)\right), \qquad c = \sum_{k=0}^{n-1} \cos\left(h_k \pi/180)\right).$$

The vehicle turn radius r is not explicitly a parameter of the behavior, but is given by:

$$r = v/((u/180)\pi),$$

where v is the vehicle speed and u is the turn rate given by:

$$u = \texttt{TURN\_RANGE}/\texttt{MEMORY\_TIME}.$$

The same turn radius is possible with different pairs of values for `TURN_RANGE` and `MEMORY_TIME`. However,m larger values of `TURN_RANGE` allow sharper initial turns but temper the turn rate after the initial sharper turn has been achieved.

A Rendering of the MemoryTurnLimit Objective Function

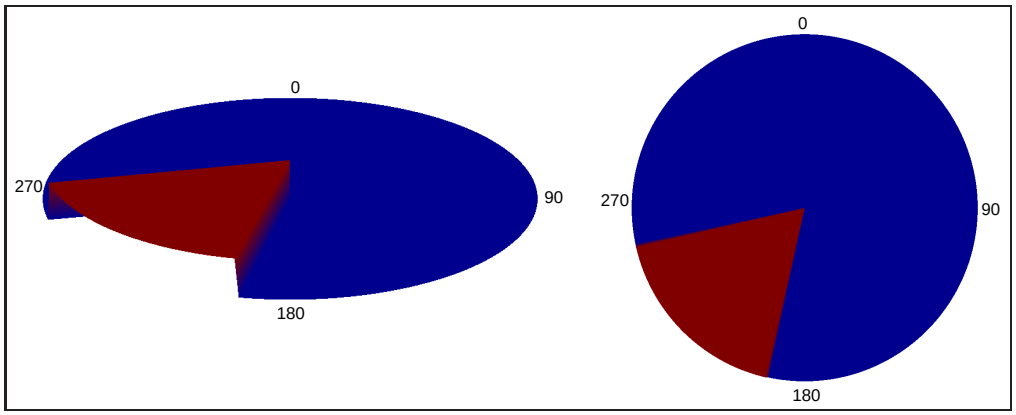

Figure 45: **The MemoryTurnLimit objective function**: The objective function produced by the MemoryTurnLimit behavior is defined over possible heading values. Depicted here is an objective function formed when the recent heading history is 225 degrees and the `turn_range` parameter is set to 30 degrees. The resulting objective function highly favors headings in the range of 190-240 degrees. One the right is a "birds-eye" view of the function, and on the right the function is viewed at an angle to appreciate the 3D quality of the function. Higher (red) values correspond to higher utility.

11.11 BHV_StationKeep

11.11.1 Overview of the BHV_StationKeep Behavior

This behavior is designed to keep the vehicle at a given lat/lon or x,y station-keep position by varying the speed to the station point as a linear function of its distance to the point. The parameters allow one to choose the two distances between which the speed varies linearly, the range of linear speeds, and a default transit speed if the vehicle is outside the outer radius.

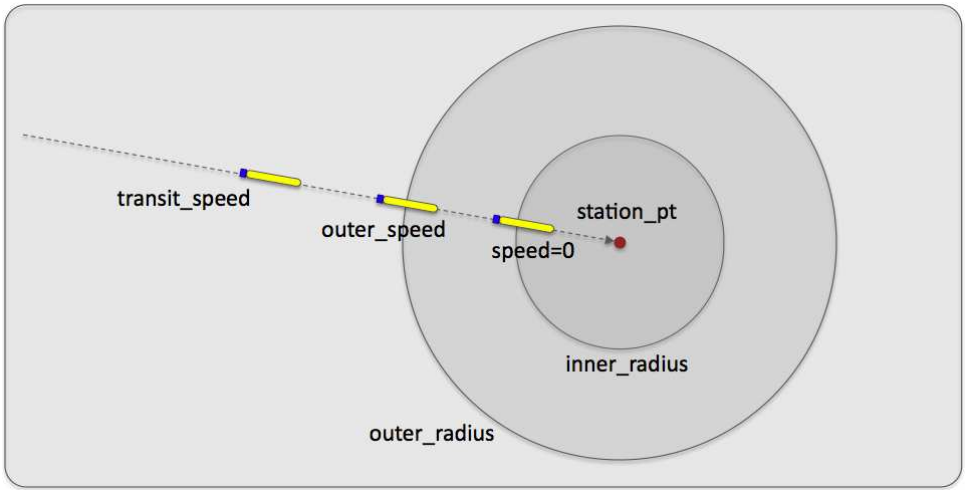

Figure 46: **The station-keep behavior parameters:** The station-keep behavior can be configured to approach the outer station circle with a given transit speed, and will decrease its preference for speed linearly between the outer radius and inner radius. The preferred speed is zero when the vehicle is at or inside the inner radius.

An alternative to this station keeping behavior is an active loiter around a very tight polygon with the `BHV_LOITER` behavior. This station keeping behavior conserves energy and aims to minimize propulsor use. The behavior can be configured to station-keep at a pre-set point, or wherever the vehicle happens to be when the behavior transitions into an active state.

The station-keep behavior was initially developed for use on an autonomous kayak. It's worth pointing out that a vehicle's control system, i.e., the front-seat driver described in Section 2.3, may have a native station-keeping mode, in which case the activation of this behavior would be replaced by a message from the backseat autonomy system to invoke the station-keeping mode. It's also worth pointing out that most UUVs are positively buoyant and will simply come to the surface if commanded with a zero-speed.

11.11.2 Brief Summary of the BHV_StationKeep Behavior Parameters

The following parameters are defined for this behavior. A more detailed description is provided other parts of this section, and in Table 21 on page 157.

STATION_PT:	An x,y pair given as a point in local coordinates.
POINT:	A supported alias for STATION_POINT.
CENTER_ACTIVE:	If true, station-keep at position upon activation.
INNER_RADIUS:	Distance to station-point within which the preferred speed is zero.
OUTER_RADIUS:	Distance within which the preferred speed begins to decrease.
OUTER_SPEED:	Preferred speed at outer radius, decreasing toward inner radius.
SWING_TIME:	Duration of drift of station circle with vehicle upon activation.
TRANSIT_SPEED:	Preferred speed beyond the outer radius.
EXTRA_SPEED:	A deprecated alias for TRANSIT_SPEED.
PASSIVE_STATION_RADIUS:	A radius used for low-power, passive station-keeping.
PASSIVE_STATION_VARIABLE:	Name of MOOS variable used for conveying passive-station mode.

11.11.3 Setting the Station-Keep Point and Radial-Speed Relationships

The station-keep point is set in one of two ways: either with a pre-specified fixed position, or with the vehicle's current position when the vehicle transitions into the running state. To set a fixed station-keep position:

```
station_pt = 100,250
```

To configure the behavior to station-keep at the vehicle's current position when it enters the running state:

```
center_active = true  // "true" is case insensitive
```

At the outset of station-keeping via center_activate, the vehicle typically is moving at some speed. Despite the fact that station-keeping is immediately active and typically results in a desired speed of zero if no other behaviors are active, the vehicle will continue some distance before coming to a near or complete stop in the water, thus "over-shooting" the station-keep point. This often means that the station-keep behavior will immediately turn the vehicle around to come back to the station-keep point. This can be countered by setting the behavior's "swing time" parameter, the amount of time after initial center-activation that the station-keep point is allowed to drift with the current position of the vehicle before becoming fixed. The format is:

```
swing_time = <time-duration>   // default is 0
```

The <time-duration> is given in seconds and the duration is clipped by the range $[0, 60]$.

If the behavior enters the running state, but center-activation is not set to true, and no pre-specified fixed position is given, the behavior will not produce an objective function. It will remain in the running state, but not the active state. (See Section 6.5.3 for more detail on behavior run states.) In this situation, a warning will be posted: BHV_WARNING="STATION_POINT_NOT_SET".

The INNER_RADIUS and OUTER_RADIUS parameters affect the preferred speed of the behavior as it relates to the vehicle's current range to the station point. The preferred speed at the outer radius is given by the parameter OUTER_SPEED. The preferred speed decreases linearly to zero as the

vehicle approaches the inner radius. The default values for the inner and outer radii are 4 and 15 respectively. If configured with values such that the inner is greater then the outer, this will not trigger an error, but the two radii parameters will be collapsed to the value of the inner radius on the first iteration of the behavior.

11.11.4 Passive Low-Energy Station Keeping Mode

The station-keep behavior can be configured to operate in a "passive" mode. This mode differs from the default mode primarily in the way it acts after it reaches the inner-radius, i.e., the point at which the behavior regards the vehicle to be on-station and outputs a preferred speed of zero. In the normal mode, the behavior will begin to output a preferred heading and non-zero speed as soon as the vehicle slips beyond the inner-radius. In the passive mode, the behavior will let the vehicle drift or otherwise move to a distance specified by the `PASSIVE_STATION_RADIUS` before it resumes outputting a preferred heading and non-zero speed. The idea is shown in Figure 47.

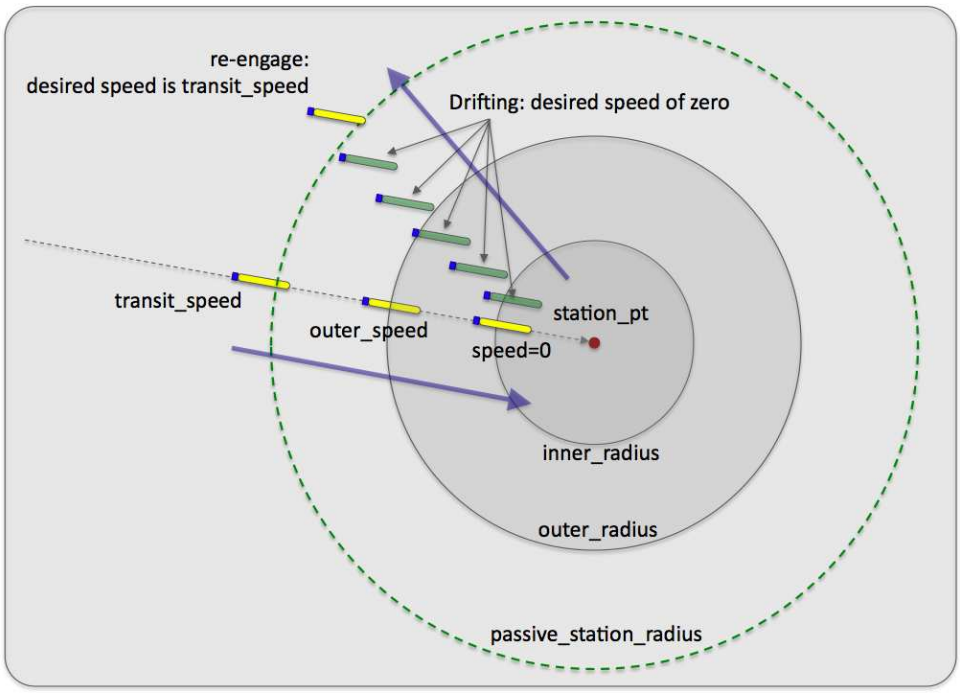

Figure 47: **Passive station-keeping:** The station-keep behavior can be configured in the "passive" mode. The vehicle will move toward the station point until it reaches the inner_radius or until progress ceases. It will then drift until its distance to the station point is beyond the passive_station_radius. At this point it will re-engage to reach the station-point and may trigger another behavior to dive.

This mode was built with UUVs in mind. Most UUVs are deployed having a positive buoyancy (battery dies - vehicle floats to the surface). They need to be moving at some speed to maintain a depth. Furthermore, it may not be safe to assume that a UUV can effectively execute a desired heading when it is operating on the surface. For these reasons, when operating in the passive mode, this behavior will publish a variable indicating whether it is in the mode of drifting or attempting

to make progress toward the station point. The status is published in the variable PSKEEP_MODE, short for "passive station-keeping mode". This variable will be set to "SEEKING_STATION" when outputting a non-zero speed preference, and presumably moving toward the station-point. The variable will be set to "HIBERNATING" otherwise. This opens the option of configuring the helm with the ConstantDepth behavior to work in conjunction with the StationKeep behavior by conditioning the ConstantDepth behavior to be running only when PSKEEP_MODE="SEEKING_STATION". The idea is shown in Figure 48.

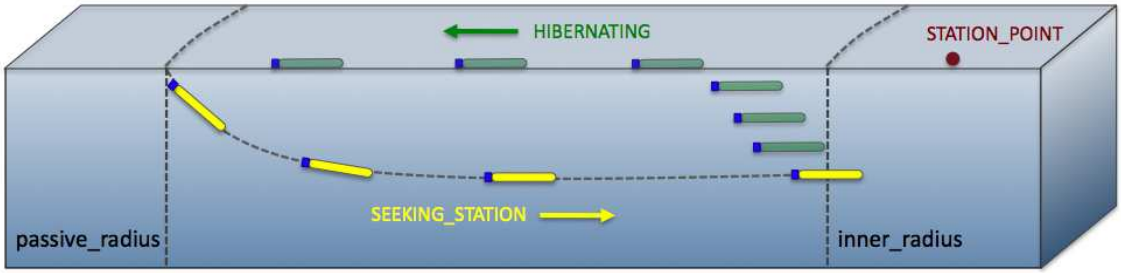

Figure 48: **Passive station-keeping with depth coordination:** The passive mode can be coordinated with the ConstantDepth behavior to dive each time the StationKeep behavior enters the "SEEKING_STATION" mode. This ensures that a UUV needing to be at depth to have reliable heading control will indeed be at depth when it needs to be.

This behavior mode is regarded as "low-power" due to the presumably long periods of drifting before resuming actively seeking the station point. A couple of safeguards are designed to ensure that when the behavior is in the "STATION_SEEKING" mode, that it does not get hung or stuck in this mode for much longer than intended or needed. How could one become stuck in this mode? Two ways - by either reaching an equilibrium at-speed, (and perhaps at-depth) state where the vehicle is neither progressing toward or way from the inner_radius, or by repeatedly "missing" the inner_radius by heading right past it.

Both cases can be guarded against and detected by monitoring the history of vehicle speed in the direction of the station-point. If this speed becomes zero, an equilibrium state is assumed, and if it becomes negative, it is assumed that the vehicle missed the inner radius circle entirely. In short, the StationKeep behavior exits the "STATION_SEEKING" mode and enters the "HIBERNATING" mode when it detects the vehicle speed toward the station-point reach zero. To calculate this vehicle speed, a ten-second history of range to the station-point is kept by the behavior. A zero speed, or "stale-progress" criteria is declared simply if the range to the station-point for the most recent measure in the history is not less than the range of ten seconds ago in the history list. The behavior will transition into the "HIBERNATING" mode if either the inner-radius or stale-progress criteria are met.

It is also possible that when the StationKeep behavior enters the "SEEKING_STATION" mode from the "HIBERNATING" mode, that the vehicle initially begins to open its range to the station-point before it begins to close range. This would be expected, for example, if the vehicle were pointed away from the station-point when the behavior first entered the "SEEKING_STATION" mode. In this case it's quite possible that the behavior would correctly, but unwantingly, infer that the stale-

progress criteria has been met. For this reason, the stale-progress criteria is not applied until an "initial-progress" criteria is met after entering the "SEEKING_STATION" mode. The same ten second history is used to detect when the vehicle begins to make initial progress, i.e., closing range, toward the station-point.

11.11.5 Station Keeping On Demand

A common, and perhaps recommended configuration, is to have one station-keep behavior defined for a given helm configuration and have it set to be usable in one of three ways: (a) station-keep at a default pre-specified position, (b) station-keep at a specified position dynamically provided, or (c) station-keep at the vehicle's present position when activated. The behavior would be configured as follows:

```
STATION_PT    = 100,200 // The default station-keep point
CENTER_ACTIVE = false
UPDATES       = STATION_UPDATES
CONDITION     = STATION_REQUEST = true
```

Then, to use the station-keep behavior in the above three ways, the following three pairs of postings, i.e., pokes, to the MOOSDB would be used. See Section 7.2.2 for more on the UPDATES parameter defined for all behaviors - by utilizing this dynamic configuration hook, the one behavior configuration above can be used in these different manners. The first pair would result in the behavior keeping station at its pre-arranged point of 100,200:

```
STATION_REQUEST = true
STATION_UPDATES = CENTER_ACTIVATE=false"
```

The second line above dynamically configures the behavior parameter CENTER_ACTIVATE to be false to ensure that the point given by the original STATION_PT parameter is used. Even though the CENTER_ACTIVATE parameter is initially set to false, the above usage sets it to false anyway, to be safe, and in case it has been dynamically set to true in a prior usage.

In the second case below, again the CENTER_ACTIVATE parameter is dynamically set to false for the same reasons. In this case the STATION_POINT parameter is also dynamically configured with a given point:

```
STATION_REQUEST = true
STATION_UPDATES = "STATION_PT=45,-150 # CENTER_ACTIVATE=false"
```

In the last case, below, the behavior is activated and configured to station-keep at the vehicle's present position when activated. There is no need to tinker with the STATION_PT parameter since this parameter is ignored when CENTER_ACTIVATE is true:

```
STATION_REQUEST = true
STATION_UPDATES = "CENTER_ACTIVATE=true"
```

It's worth noting that above variable-value pairs that trigger the station-keep behavior could have come from a variety of sources. They could be endflags from another behavior. They could have come from a poke using uPokeDB, uTermCommand, pMarineViewer or any third party command and control interface.

11.12 BHV_Timer

This behavior can nearly be considered a no-op behavior; it has no functionality beyond what is derived from the parent IvPBehavior class. It can be used to set a timer between the observation of one or more events (with condition flags) and the posting of one or more events (with end flags). The `DURATION`, `DURATION_STATUS`, `CONDITION`, `RUNFLAG` and `ENDFLAG` parameters are all defined generally for behaviors. There are no additional parameters defined for this behavior.

13 Appendix - Behavior Summaries

Parameter Summary for BHV_Waypoint

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	loiter-west-zone	yes	*mandatory*	71
duration	double	600	-	-1	74
duration_status	MOOSVAR	loiter_remaining	yes	-	74
priority, pwt	double	100	-	100	71
runflag	MOOSVAR=value	LOITERING = maybe	yes	-	62
endflag	MOOSVAR=value	LOITERING = done	yes	-	62
activeflag	MOOSVAR=value	LOITERING = yes	yes	-	62
inactiveflag	MOOSVAR=value	LOITERING = off	yes	-	62
idleflag	MOOSVAR=value	LOITERING = no	yes	-	62
nostarve	MOOSVAR,double	INFO,60	yes	-	75
perpetual	string	false	no	false	75
updates	MOOSVAR	LOITER_INFO	yes	-	73
condition	Logic Expression	QUALITY <= 7	yes	-	61
points, polygon	string	0,0:45,0:45,80:0,0	yes	-	118
capture_radius	double	7	-	0	119
lead	double	10	no	-1	119
nm_radius	double	18	no	0	119
order	string	reverse	no	normal	118
post_suffix	string	IKE	yes	""	118
repeat	int	3	no	0	118
speed	double	1.2	-	0	118
wpt_status_var	MOOSVAR	WPT_REPORT	yes	WPT_STAT	121
wpt_index_var	MOOSVAR	WPT_IX	yes	WPT_INDEX	121
cycle_index_var	MOOSVAR	CYCLE_IX	yes	CYCLE_INDEX	121
cycleflag	MOOSVAR=value	CYCLED=true	yes	-	121

Table 11: Parameters for the BHV_Waypoint behavior.

Example Behavior File Configuration for BHV_Waypoint

Listing 13.1 - An example BHV_Waypoint configuration.

```
 0  Behavior = BHV_Waypoint
 1  {
 2    name       = waypt_survey
 3    priority   = 100
 4    updates    = WPT_SURVEY_UPDATES
 5    condition  = (DEPLOY == true) or (SURVEY == on))
 6    endflag    = SURVEY = COMPLETE
 7
 8            points = label,survey_points:-57,-60:-70,-109:-77,-144:-51
 9
10             speed = 3.0  // meters per second
11    capture_radius = 8.0  // meters
12         nm_radius = 16.5 // meters
13            repeat = 0    // number of iterations
14              lead = 10   // meters
15  }
```

Parameter Summary for BHV_OpRegion

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	loiter-west-zone	yes	*mandatory*	71
duration	double	600	-	-1	74
duration_status	MOOSVAR	loiter_remaining	yes	-	74
priority, pwt	double	100	-	100	71
runflag	MOOSVAR=value	LOITERING = maybe	yes	-	62
endflag	MOOSVAR=value	LOITERING = done	yes	-	62
activeflag	MOOSVAR=value	LOITERING = yes	yes	-	62
inactiveflag	MOOSVAR=value	LOITERING = off	yes	-	62
idleflag	MOOSVAR=value	LOITERING = no	yes	-	62
nostarve	MOOSVAR,double	INFO,60	yes	-	75
perpetual	string	false	no	false	75
updates	MOOSVAR	LOITER_INFO	yes	-	73
condition	Logic Expression	QUALITY <= 7	yes	-	61
polygon	string	0,0:45,0:45,80:0,80:0,0	-	-	122
max_depth	double	200	-	0	123
min_altitude	double	25	-	0	123
max_time	double	3600	-	0	123
trigger_entry_time	double	1.5	-	0	122
trigger_exit_time	double	2.4	-	0	122

Table 12: Parameters for the BHV_OpRegion behavior.

Example Behavior File Configuration for BHV_OpRegion

Listing 13.2 - An example BHV_OpRegion configuration.

```
 0  Behavior = BHV_OpRegion
 1  {
 2    name               = bhv_opregion
 3    polygon            = label,opregion : -57,-60 : -70,-109 : -77,-144
 4
 5    max_depth          = 50     // meters
 6    min_altitude       = 10     // meters
 7    max_time           = 3600   // seconds
 8    trigger_entry_time = 0.5    // seconds
 9    trigger_exit_time  = 1.0    // seconds
10 }
```

Parameter Summary for BHV_Loiter

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	loiter-west-zone	yes	*mandatory*	71
duration	double	600	-	-1	74
duration_status	MOOSVAR	loiter_remaining	yes	-	74
priority, pwt	double	100	-	100	71
runflag	MOOSVAR=value	LOITERING = maybe	yes	-	62
endflag	MOOSVAR=value	LOITERING = done	yes	-	62
activeflag	MOOSVAR=value	LOITERING = yes	yes	-	62
inactiveflag	MOOSVAR=value	LOITERING = off	yes	-	62
idleflag	MOOSVAR=value	LOITERING = no	yes	-	62
nostarve	MOOSVAR,double	INFO,60	yes	-	75
perpetual	string	false	no	false	75
updates	MOOSVAR	LOITER_INFO	yes	-	73
condition	Logic Expression	QUALITY <= 7	yes	-	61
polygon	string	0,0:45,0:45,80:0,80:0,0	yes	-	125
speed	double	1.5	-	0	125
radius	double	10	-	0	125
nm_radius	double	25	-	0	125
clockwise	string	FALSE	no	true	125
acquire_dist	double	15	-	10	125
post_suffix	string	REGION-1	yes	""	125

Table 13: Parameters for the BHV_Loiter behavior.

Example Behavior File Configuration for BHV_Loiter

Listing 13.3 - An example BHV_Loiter configuration.

```
 0  Behavior = BHV_Loiter
 1  {
 2    name     = loiter_alpha
 3    pwt      = 100
 4    duration = 3600 // One hour
 5    updates  = LOITER_ALPHA_UPDATES
 7
 8        polygon = radial:100,-100,80,12
 9          speed = 3.0
10         radius = 8.0
11      nm_radius = 16.0
12      clockwise = true
13    aquire_dist = 25
14  }
```

Parameter Summary for BHV_PeriodicSpeed

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	`loiter-west-zone`	yes	*mandatory*	71
duration	double	`600`	-	`-1`	74
duration_status	MOOSVAR	`loiter_remaining`	yes	-	74
priority, pwt	double	`100`	-	`100`	71
runflag	MOOSVAR=value	`LOITERING = maybe`	yes	-	62
endflag	MOOSVAR=value	`LOITERING = done`	yes	-	62
activeflag	MOOSVAR=value	`LOITERING = yes`	yes	-	62
inactiveflag	MOOSVAR=value	`LOITERING = off`	yes	-	62
idleflag	MOOSVAR=value	`LOITERING = no`	yes	-	62
nostarve	MOOSVAR,double	`INFO,60`	yes	-	75
perpetual	string	`false`	no	`false`	75
updates	MOOSVAR	`LOITER_INFO`	yes	-	73
condition	Logic Expression	`QUALITY <= 7`	yes	-	61
period_length	double	`60`	-	`0`	127
period_gap	double	`600`	-	`0`	127
period_speed	double	`0.8`	-	`0`	127
period_peakwidth	double	`0.2`	-	`0`	127
period_basewidth	double	`0.5`	-	`0`	127
stat_pending_inactive	MOOSVAR	`PS_PENDING_INACTIVE`	yes	-	127
stat_pending_active	MOOSVAR	`PS_PENDING_ACTIVE`	yes	-	127

Table 14: Parameters for the BHV_PeriodicSpeed behavior.

Example Behavior File Configuration for BHV_PeriodicSpeed

Listing 13.3 - An example BHV_PeriodicSpeed configuration.

```
 0  Behavior = BHV_PeriodicSpeed
 1  {
 2    name     = periodic_speed
 3    priority = 500
 4
 5       period_length      = 30       // seconds
 6          period_gap      = 120      // seconds
 7        period_speed      = 0.5      // meters/sec
 8    period_peakwidth      = 0.1
 9    period_basewidth      = 0.5
10    stat_pending_active   = PS_PENDING_ACTIVE
11    stat_pending_inactive = PS_PENDING_INACTIVE
12  }
```

Parameter Summary for BHV_PeriodicSurface

Parameter	Argument Type	Example	Case-Sense	Default	Page
name	string	loiter-west-zone	yes	*mandatory*	71
duration	double	600	-	-1	74
duration_status	MOOSVAR	loiter_remaining	yes	-	74
priority, pwt	double	100	-	100	71
runflag	MOOSVAR=value	LOITERING = maybe	yes	-	62
endflag	MOOSVAR=value	LOITERING = done	yes	-	62
activeflag	MOOSVAR=value	LOITERING = yes	yes	-	62
inactiveflag	MOOSVAR=value	LOITERING = off	yes	-	62
idleflag	MOOSVAR=value	LOITERING = no	yes	-	62
nostarve	MOOSVAR,double	INFO,60	yes	-	75
perpetual	string	false	no	false	75
updates	MOOSVAR	LOITER_INFO	yes	-	73
condition	Logic Expression	QUALITY <= 7	yes	-	61
period	double	60	-	300	128
mark_variable	MOOSVAR	GPS_RECEIVED	yes	GPS_UPDATE_RECEIVED	129
atsurface_status_variable	MOOSVAR	TIME_AT_SURFACE	yes	TIME_AT_SURFACE	129
pending_status_variable	MOOSVAR	PENDING_SURFACE	yes	PENDING_SURFACE	129
ascent_speed	double	1.0	-	*	129
ascent_grade	string	quasi	no	linear	129
zero_speed_depth	double	2.5	-	0	129
max_time_at_surface	MOOSVAR	60	yes	300	129

Table 15: Parameters for the BHV_PeriodicSurface behavior.

Example Behavior File Configuration for BHV_PeriodicSurface

Listing 13.4 - An example BHV_PeriodicSurface configuration.

```
 0  Behavior = BHV_PeriodicSurface
 1  {
 2    name          = bhv_periodic_surface
 3    priority      = 500
 4    active_flag   = SURFACING, IN_PROGRESS
 5    inactive_flag = SURFACING, NO
 6
 7                 period = 3600     // seconds
 8           ascent_speed = 1.0      // meters per second
 9       zero_speed_depth = 2.5      // meters
10    max_time_at_surface = 120      // seconds
11           ascent_grade = linear
12         mark_variable = GPS_UPDATE_RECEIVED
13       status_variable = PERIODIC_PENDING_SURFACE
14  }
```

Parameter Summary for BHV_ConstantDepth

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	`loiter-west-zone`	yes	*mandatory*	71
duration	double	`600`	-	`-1`	74
duration_status	MOOSVAR	`loiter_remaining`	yes	-	74
priority, pwt	double	`100`	-	`100`	71
runflag	MOOSVAR=value	`LOITERING = maybe`	yes	-	62
endflag	MOOSVAR=value	`LOITERING = done`	yes	-	62
activeflag	MOOSVAR=value	`LOITERING = yes`	yes	-	62
inactiveflag	MOOSVAR=value	`LOITERING = off`	yes	-	62
idleflag	MOOSVAR=value	`LOITERING = no`	yes	-	62
nostarve	MOOSVAR,double	`INFO,60`	yes	-	75
perpetual	string	`false`	no	`false`	75
updates	MOOSVAR	`LOITER_INFO`	yes	-	73
condition	Logic Expression	`QUALITY <= 7`	yes	-	61
depth	double	`35`	-	`0`	130
peakwidth	double	`5`	-	`0`	130
basewidth	double	`15`	-	`2`	130

Table 16: Parameters for the BHV_ConstantDepth behavior.

Example Behavior File Configuration for BHV_ConstantDepth

Listing 13.5 - An example BHV_ConstantDepth configuration.

```
 0  Behavior = BHV_ConstantDepth
 1  {
 2    // General Behavior Parameters
 3    name      = constant_depth_survey
 4    priority  = 100
 5    condition = AUTONOMY_MODE = SURVEY
 6    duration  = no-time-limit
 7    updates   = NEW_SURVEY_DEPTH
 8    nostarve  = NAV_DEPTH, 3.0
 9
10    // BHV_ConstantDepth Behavior Parameters
11        depth = 50      // meters
12    peakwidth = 5
13    basewidth = 10
14  }
```

Parameter Summary for BHV_ConstantHeading

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	`loiter-west-zone`	yes	*mandatory*	71
duration	double	`600`	-	`-1`	74
duration_status	MOOSVAR	`loiter_remaining`	yes	-	74
priority, pwt	double	`100`	-	`100`	71
runflag	MOOSVAR=value	`LOITERING = maybe`	yes	-	62
endflag	MOOSVAR=value	`LOITERING = done`	yes	-	62
activeflag	MOOSVAR=value	`LOITERING = yes`	yes	-	62
inactiveflag	MOOSVAR=value	`LOITERING = off`	yes	-	62
idleflag	MOOSVAR=value	`LOITERING = no`	yes	-	62
nostarve	MOOSVAR,double	`INFO,60`	yes	-	75
perpetual	string	`false`	no	`false`	75
updates	MOOSVAR	`LOITER_INFO`	yes	-	73
condition	Logic Expression	`QUALITY <= 7`	yes	-	61
heading	double	`35`	-	`0`	130
peakwidth	double	`5`	-	`10`	130
basewidth	double	`175`	-	`170`	130

Table 17: Parameters for the BHV_ConstantHeading behavior.

Example Behavior File Configuration for BHV_ConstantHeading

Listing 13.6 - An example BHV_ConstantHeading configuration.

```
 0  Behavior = BHV_ConstantHeading
 1  {
 2    name      = bhv_constant_heading
 3    priority  = 100
 4    duration  = 60
 5    condition = AUTONOMY_MODE = PID_TEST
 6    updates   = NEW_TEST_HEADING
 7    nostarve  = NAV_HEADING, 3.0
 8
 9      heading = 45  // degrees
10    peakwidth =  0
11    basewidth =  5
12  }
```

Parameter Summary for BHV_ConstantSpeed

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	`loiter-west-zone`	yes	*mandatory*	71
duration	double	`600`	-	`-1`	74
duration_status	MOOSVAR	`loiter_remaining`	yes	-	74
priority, pwt	double	`100`	-	`100`	71
runflag	MOOSVAR=value	`LOITERING = maybe`	yes	-	62
endflag	MOOSVAR=value	`LOITERING = done`	yes	-	62
activeflag	MOOSVAR=value	`LOITERING = yes`	yes	-	62
inactiveflag	MOOSVAR=value	`LOITERING = off`	yes	-	62
idleflag	MOOSVAR=value	`LOITERING = no`	yes	-	62
nostarve	MOOSVAR,double	`INFO,60`	yes	-	75
perpetual	string	`false`	no	`false`	75
updates	MOOSVAR	`LOITER_INFO`	yes	-	73
condition	Logic Expression	`QUALITY <= 7`	yes	-	61
speed	double	`1.2`	-	`0.0`	131
peakwidth	double	`0.1`	-	`0.0`	131
basewidth	double	`0.6`	-	`2.0`	131

Table 18: Parameters for the BHV_ConstantSpeed behavior.

Example Behavior File Configuration for BHV_ConstantSpeed

Listing 13.7 - An example BHV_ConstantSpeed configuration.

```
 0  Behavior = BHV_ConstantSpeed
 1  {
 2    name        = const_speed_bravo
 3    priority    = 100
 4    duration    = 60
 5    active_flag = BRAVO_SPEED_TEST = in-progress
 6    nostarve    = NAV_SPEED, 2.0
 7
 8    speed     = 1.8  // meters per second
 9    peakwidth = 0.3
10    basewidth = 1.0
11  }
```

Parameter Summary for BHV_GoToDepth

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	`loiter-west-zone`	yes	*mandatory*	71
duration	double	`600`	-	`-1`	74
duration_status	MOOSVAR	`loiter_remaining`	yes	-	74
priority, pwt	double	`100`	-	`100`	71
runflag	MOOSVAR=value	`LOITERING = maybe`	yes	-	62
endflag	MOOSVAR=value	`LOITERING = done`	yes	-	62
activeflag	MOOSVAR=value	`LOITERING = yes`	yes	-	62
inactiveflag	MOOSVAR=value	`LOITERING = off`	yes	-	62
idleflag	MOOSVAR=value	`LOITERING = no`	yes	-	62
nostarve	MOOSVAR,double	`INFO,60`	yes	-	75
perpetual	string	`false`	no	`false`	75
updates	MOOSVAR	`LOITER_INFO`	yes	-	73
condition	Logic Expression	`QUALITY <= 7`	yes	-	61
depth, depths	string	`50,10:40,60`	yes	-	132
repeat	int	`5`	-	`0`	132
capture_delta	double	`2`	-	`2.5`	132
capture_flag	MOOSVAR	`DEPTH_HIT`	yes	-	132

Table 19: Parameters for the BHV_GoToDepth behavior.

Example Behavior File Configuration for BHV_GoToDepth

Listing 13.8 - An example BHV_GoToDepth configuration.

```
 0  Behavior = BHV_GoToDepth
 1  {
 2    name      = goto_depth_set_alpha
 3    priority  = 100
 4    condition = DEPLOY == true
 9    endflag   = GOTO_DEPTH_ALPHA = DONE
 5
 6           depths = 15,30: 30,30: 45,60: 15,30
 7    capture_delta = 1  // meters
 8     capture_flag = DEPTH_LEVELS_ACHIEVED
10  }
```

Parameter Summary for BHV_MemoryTurnLimit

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	`loiter-west-zone`	yes	*mandatory*	71
duration	double	`600`	-	`-1`	74
duration_status	MOOSVAR	`loiter_remaining`	yes	-	74
priority, pwt	double	`100`	-	`100`	71
runflag	MOOSVAR=value	`LOITERING = maybe`	yes	-	62
endflag	MOOSVAR=value	`LOITERING = done`	yes	-	62
activeflag	MOOSVAR=value	`LOITERING = yes`	yes	-	62
inactiveflag	MOOSVAR=value	`LOITERING = off`	yes	-	62
idleflag	MOOSVAR=value	`LOITERING = no`	yes	-	62
nostarve	MOOSVAR,double	`INFO,60`	yes	-	75
perpetual	string	`false`	no	`false`	75
updates	MOOSVAR	`LOITER_INFO`	yes	-	73
condition	Logic Expression	`QUALITY <= 7`	yes	-	61
memory_time	double	`60`	-	`-1`	134
turn_range	double	`45`	-	`-1`	134

Table 20: Parameters for the BHV_MemoryTurnLimit behavior.

Example Behavior File Configuration for BHV_MemoryTurnLimit

Listing 13.9 - An example BHV_MemoryTurnLimit configuration.

```
zv
 0  Behavior = BHV_MemoryTurnLimit
 1  {
 2    name      = memturnlimit
 3    priority  = 1000
 4
 5    memory_time = 60  // seconds
 6     turn_range = 35  // degrees
 7  }
```

Parameter Summary for BHV_StationKeep

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	loiter-west-zone	yes	*mandatory*	71
duration	double	600	-	-1	74
duration_status	MOOSVAR	loiter_remaining	yes	-	74
priority, pwt	double	100	-	100	71
runflag	MOOSVAR=value	LOITERING = maybe	yes	-	62
endflag	MOOSVAR=value	LOITERING = done	yes	-	62
activeflag	MOOSVAR=value	LOITERING = yes	yes	-	62
inactiveflag	MOOSVAR=value	LOITERING = off	yes	-	62
idleflag	MOOSVAR=value	LOITERING = no	yes	-	62
nostarve	MOOSVAR,double	INFO,60	yes	-	75
perpetual	string	false	no	false	75
updates	MOOSVAR	LOITER_INFO	yes	-	73
condition	Logic Expression	QUALITY <= 7	yes	-	61
station_pt, point	string	50,75	yes	0,0	137
center_activate	string	TRUE	no	false	137
inner_radius	double	10	-	4	137
outer_radius	double	25	-	15	137
outer_speed	double	1.8	-	1.2	137
transit_speed	double	1.9	-	2.5	137
passive_station_radius	double	200	-	0	138
passive_station_variable	MOOSVAR	PSK_MODE_CHARLIE	-	PSKEEP_MODE	138

Table 21: Parameters for the BHV_StationKeep behavior.

Example Behavior File Configuration for BHV_StationKeep

Listing 13.10 - An example BHV_StationKeep configuration.

```
 0  Behavior = BHV_StationKeep
 1  {
 2    name       = bhv_station_keep
 3    priority   = 100
 4    condition  = (ON_STATION=true) and (RETURN=false)
 5    updates    = STATION_UPDATES
 6
 7                  station_pt = 200,-150
 8              center_activate = true
 9                 inner_radius = 10
10                 outer_radius = 40
11                  outer_speed = 0.8
12                transit_speed = 1.8
13      passive_station_radius = 400          // meters
14    passive_station_variable = PSKEEP_MODE  // the default
15  }
```

Parameter Summary for BHV_Timer

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	`loiter-west-zone`	yes	*mandatory*	71
duration	double	`600`	-	`-1`	74
duration_status	MOOSVAR	`loiter_remaining`	yes	-	74
priority, pwt	double	`100`	-	`100`	71
runflag	MOOSVAR=value	`LOITERING = maybe`	yes	-	62
endflag	MOOSVAR=value	`LOITERING = done`	yes	-	62
activeflag	MOOSVAR=value	`LOITERING = yes`	yes	-	62
inactiveflag	MOOSVAR=value	`LOITERING = off`	yes	-	62
idleflag	MOOSVAR=value	`LOITERING = no`	yes	-	62
nostarve	MOOSVAR,double	`INFO,60`	yes	-	75
perpetual	string	`false`	no	`false`	75
updates	MOOSVAR	`LOITER_INFO`	yes	-	73
condition	Logic Expression	`QUALITY <= 7`	yes	-	61
No additional parameters for this behavior					

Table 22: Parameters for the BHV_Timer behavior.

Example Behavior File Configuration for BHV_Timer

Listing 13.11 - An example BHV_Timer configuration.

```
0  Behavior = BHV_Timer
1  {
2    name      = bhv_timer_a
3    duration  = 60             // seconds
4    condition = loiter = alpha
5    end_flag  = loiter = beta
6  }
```

Parameter Summary for BHV_AvoidCollision

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	`loiter-west-zone`	yes	*mandatory*	71
duration	double	`600`	-	`-1`	74
duration_status	MOOSVAR	`loiter_remaining`	yes	-	74
priority, pwt	double	`100`	-	`100`	71
runflag	MOOSVAR=value	`LOITERING = maybe`	yes	-	62
endflag	MOOSVAR=value	`LOITERING = done`	yes	-	62
activeflag	MOOSVAR=value	`LOITERING = yes`	yes	-	62
inactiveflag	MOOSVAR=value	`LOITERING = off`	yes	-	62
idleflag	MOOSVAR=value	`LOITERING = no`	yes	-	62
nostarve	MOOSVAR,double	`INFO,60`	yes	-	75
perpetual	string	`false`	no	`false`	75
updates	MOOSVAR	`LOITER_INFO`	yes	-	73
condition	Logic Expression	`QUALITY <= 7`	yes	-	61
contact	string	Alliance	yes	-	142
on_no_contact_ok	boolean	true	no	`true`	142
extrapolate	boolean	true	no	`false`	142
decay	double,double	10, 30	-	`0,0`	142
active_inner_distance	double	50	-	`50`	142
active_outer_distance	double	200	-	`200`	142
all_clear_distance	double	100	-	`75`	143
collision_distance	double	10	-	`10`	143

Table 23: Parameters for the BHV_AvoidCollision behavior.

Example Behavior File Configuration for BHV_AvoidCollision

Listing 13.12 - An example BHV_AvoidCollision configuration.

```
 0  Behavior = BHV_AvoidCollision
 1  {
 2    name      = avoid_collision_alpha
 3    pwt       = 100
 4    condition = AVOIDANCE_MODE != INACTIVE
 5
 6                   contact = alpha
 7     active_outer_distance = 150
 8     active_inner_distance = 75
 9        collision_distance = 15
10        all_clear_distance = 80
11              active_grade = linear
12          on_no_contact_ok = true
13               extrapolate = true
14                     decay = 30,60
15  }
```

Parameter Summary for BHV_CutRange

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	`loiter-west-zone`	yes	*mandatory*	71
duration	double	`600`	-	`-1`	74
duration_status	MOOSVAR	`loiter_remaining`	yes	-	74
priority, pwt	double	`100`	-	`100`	71
runflag	MOOSVAR=value	`LOITERING = maybe`	yes	-	62
endflag	MOOSVAR=value	`LOITERING = done`	yes	-	62
activeflag	MOOSVAR=value	`LOITERING = yes`	yes	-	62
inactiveflag	MOOSVAR=value	`LOITERING = off`	yes	-	62
idleflag	MOOSVAR=value	`LOITERING = no`	yes	-	62
nostarve	MOOSVAR,double	`INFO,60`	yes	-	75
perpetual	string	`false`	no	`false`	75
updates	MOOSVAR	`LOITER_INFO`	yes	-	73
condition	Logic Expression	`QUALITY <= 7`	yes	-	61
contact	string	Alliance	yes	-	142
on_no_contact_ok	boolean	true	no	`true`	142
extrapolate	boolean	true	no	`false`	142
decay	double,double	10, 30	-	`0,0`	142
dist_priority_interval	double,double	`40,100`	-	`0,0`	144
time_on_leg	double	`60`	-	`15`	144
give_up_range	double	`500`	-	`0`	144
patience	double	`50`	-	`0`	144

Table 24: Parameters for the BHV_CutRange behavior.

Example Behavior File Configuration for BHV_CutRange

Listing 13.13 - An example BHV_CutRange configuration.

```
0  Behavior = BHV_CutRange
1  {
2    name         = bhv_cutrange
3    pwt          = 100
4    contact      = zulu
5
5    dist_priority_interval = 25,100
6                time_on_leg = 60
7              give_up_range = 400
8                   patience = 75
9  }
```

Parameter Summary for BHV_Shadow

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	`loiter-west-zone`	yes	*mandatory*	71
duration	double	`600`	-	`-1`	74
duration_status	MOOSVAR	`loiter_remaining`	yes	-	74
priority, pwt	double	`100`	-	`100`	71
runflag	MOOSVAR=value	`LOITERING = maybe`	yes	-	62
endflag	MOOSVAR=value	`LOITERING = done`	yes	-	62
activeflag	MOOSVAR=value	`LOITERING = yes`	yes	-	62
inactiveflag	MOOSVAR=value	`LOITERING = off`	yes	-	62
idleflag	MOOSVAR=value	`LOITERING = no`	yes	-	62
nostarve	MOOSVAR,double	`INFO,60`	yes	-	75
perpetual	string	`false`	no	`false`	75
updates	MOOSVAR	`LOITER_INFO`	yes	-	73
condition	Logic Expression	`QUALITY <= 7`	yes	-	61
contact	string	Alliance	yes	-	142
on_no_contact_ok	boolean	true	no	`true`	142
extrapolate	boolean	true	no	`false`	142
decay	double,double	10, 30	-	`0,0`	142
max_range	double	`100`	-	`0`	144
heading_peakwidth	double	`10`	-	`20`	144
heading_basewidth	double	`170`	-	`160`	144
speed_peakwidth	double	`0.3`	-	`0.1`	144
speed_basewidth	double	`0.5`	-	`2.0`	144

Table 25: Parameters for the BHV_Shadow behavior.

Example Behavior File Configuration for BHV_Shadow

Listing 13.14 - An example BHV_Shadow configuration.

```
 0  Behavior = BHV_Shadow
 1  {
 2    name      = bhv_shadow
 3    pwt       = 100
 4    contact   = delta
 5
 6            max_range = 200
 7    heading_peakwidth = 10
 8    heading_basewidth = 170
 9      speed_peakwidth = 10
10      speed_basewidth = 170
11  }
```

Parameter Summary for BHV_Trail

Parameter	Argument Type	Example	Case-Sensitive	Default	Page
name	string	`loiter-west-zone`	yes	*mandatory*	71
duration	double	`600`	-	`-1`	74
duration_status	MOOSVAR	`loiter_remaining`	yes	-	74
priority, pwt	double	`100`	-	`100`	71
runflag	MOOSVAR=value	`LOITERING = maybe`	yes	-	62
endflag	MOOSVAR=value	`LOITERING = done`	yes	-	62
activeflag	MOOSVAR=value	`LOITERING = yes`	yes	-	62
inactiveflag	MOOSVAR=value	`LOITERING = off`	yes	-	62
idleflag	MOOSVAR=value	`LOITERING = no`	yes	-	62
nostarve	MOOSVAR,double	`INFO,60`	yes	-	75
perpetual	string	`false`	no	`false`	75
updates	MOOSVAR	`LOITER_INFO`	yes	-	73
condition	Logic Expression	`QUALITY <= 7`	yes	-	61
contact	string	Alliance	yes	-	142
on_no_contact_ok	boolean	true	no	`true`	142
extrapolate	boolean	true	no	`false`	142
decay	double,double	10, 30	-	`0,0`	142
trail_range	double	`20`	-	`50`	144
trail_angle	double	`270`	-	`180`	144
trail_angle_type	string	`absolute`	no	`relative`	144
radius	double	`8`	-	`5`	144
nm_radius	double	`20`	-	`20`	144
max_range	double	`50`	-	`0`	144

Table 26: Parameters for the BHV_Trail behavior.

Example Behavior File Configuration for BHV_Trail

Listing 13.15 - An example BHV_Trail configuration.

```
 0  Behavior = BHV_Trail
 1  {
 2    name             = bhv_trail
 3    priority         = 100
 5
 5    contact          = delta
 6    extrapolate      = true
 7    on_no_contact_ok = true
 8    decay            = 20,60    // seconds
 9
10         trail_range = 50       // meters
11         trail_angle = 185      // degrees
12    trail_angle_type = relative
13              radius = 10       // meters
14           nm_radius = 30       // meters
15           max_range = 300      // meters
16  }
```