The AvoidObstacleV24 Behavior

Fall 2024

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139
`project-pavlab/bhvdocs/bhv_avoid_obstacle`

Contents

1 The AvoidObstacle Behavior

The AvoidObstacleV24 behavior is designed to produce IvP functions for avoiding a single convex polygon obstacle. It is typically configured as a behavior *template* in the helm and works with an *obstacle manager* with the latter spawning new behavior instances for each known obstacle when the vehicle is sufficiently close to the obstacle. In the example in Figure **??** below, two behavior instances are spawned, one for each of the two shown obstacles. Each instance starts exactly in the same way, evolving in state differently as it maneuvers around its respective obstacle. As an obstacle recedes behind the vehicle, its behavior at some point is completed, deleted and forgotten.

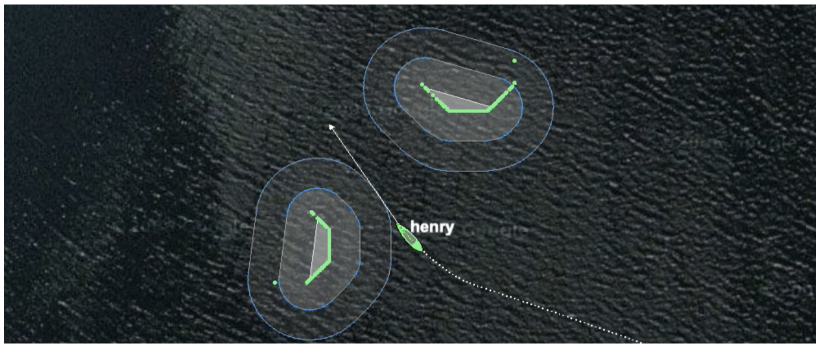


Figure 1: **AvoidObstacleV24 Behavior**: The operation area

1.1 Overview

The AvoidObstacle behavior acts upon a single obstacle given by a convex polygon, Figure 2. When a robot has multiple obstacles, multiple versions of the behavior are spawned, one for each obstacle. During the course of a mission, it is expected that multiple such behavior instances will be spawned and deleted.

The AvoidObstacle behavior is typically configured as a template for spawning instances as obstacles emerge. In less common cases, one or more static behaviors may be configured, instantiated at mission launch time, with a obstacle of a known position and location. Both use cases will be described. In the case of dynamically spawned behaviors, the helm and the AvoidObstacle behavior work in coordination with an obstacle manager which sits between the helm and sensor processing applications to reason about obstacles and post messages at the approprite time to the helm, for generating new AvoidObstacle behaviors. The obstacle manager currently used is another MOOS app called pObstacleMgr and is documented separately. The interface that triggers AvoidObstacle instances will be described.

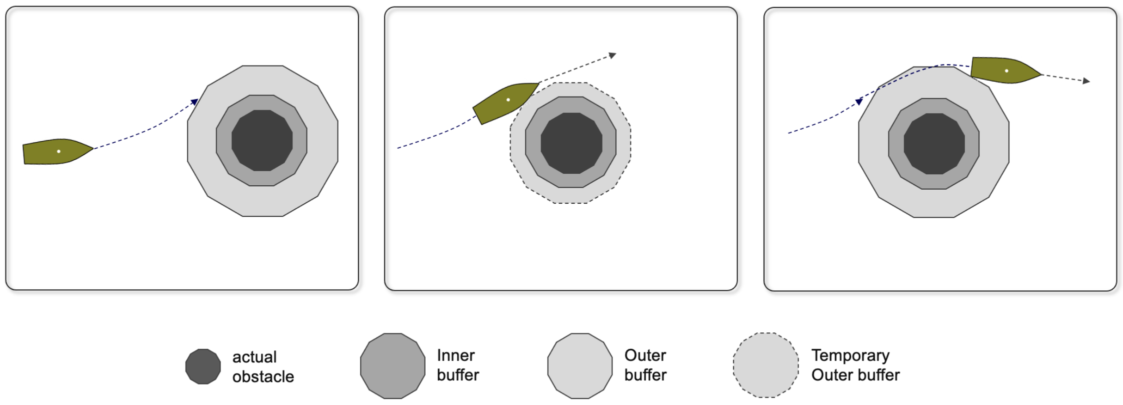


Figure 2: **The AvoidObstacle Behavior**: The behavior reasons about a single convex polygon obstacle. It uses an inner buffer and outer buffer, treating the inner buffer as essentially an extension of the obstacle, and the outer buffer as a region where it is preferred not to be in.

The AvoidObstacle behavior is not a path planner. It is a behavior that works in conjuction with a planned path, typically being executed by the Waypoint behavior. The AvoidObstacle behavior is typically used for avoid locally sensed obstacles like buoys, stationary vessels, or any other hazard that may be detected or known about beforehand. The AvoidObstacle behavior, like any IvP Helm behavior, is capable of accepting dynamic updates. This includes obstacle size and location. So the AvoidObstacle behavior is capable of dealing with moving obstacles including vessels. However, when a vessel is known to be a vessel, and has a known position, heading, and speed, the AvoidCollision or AvdColregs behaviors are more appropriate.

1.2 The Nature and Origin of Obstacles

The AvoidObstacle behavior may be configured *statically* or *dynamically*. In a static behavior the obstacle is provided in the mission configuration using the polygon configuration parameter. For example:

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

The vertices may also be specified indirectly using one more supported patterns. For example an octogon polygon, perhaps for avoiding a 1 meter sized buoy at a local coordinates `(45,90)` could be given as:

```
polygon = format=radial, x=45, y=90, radius=2, pts=8
```

In a dynamic behavior, the polygon specification is typically provided by an external source, typically the obstacle manager. In the mission file this behavior leaves the `polygon` parameter unspecified, but does specify a MOOS variable for receiving updates. For example:

```
templating = spawn
updates = OBSTACLE_ALRERT
```

A new AvoidObstacle behavior

In both cases, they may be altered during run time.

1.3 Configuration Parameters

Listing 1.1: Configuration Parameters Common to All Behaviors.

`activeflag`:	A MOOS variable-value pair posted when the behavior is in the *active* state. [more].
`condition`:	Specifies a condition that must be met for the behavior to be running. [more].
`duration`:	Time in behavior will remain running before declaring completion. [more].
`duration_idle_decay`:	When true, duration clock is running even when in the *idle* state. [more].
`duration_reset`:	A variable-pair such as `MY_RESET=true`, that will trigger a duration reset. [more].
`duration_status`:	The name of a MOOS variable to which the vehicle duration status is published. [more].
`endflag`:	A MOOS variable-value pair posted when the behavior has completed. [more].
`idleflag`:	A MOOS variable-value pair posted when the behavior is in the *idle* state. [more].
`inactiveflag`:	A MOOS variable-value posted when the behavior is *not* in the *active* state. [more].
`name`:	The (unique) name of the behavior. [more].
`nostarve`:	Allows a behavior to assert a maximum staleness for a MOOS variable. [more].
`perpetual`:	If true allows the behavior to to run even after it has completed. [more].
`post_mapping`:	Re-direct behavior output normally to one MOOS variable to another instead. [more].

priority:	The priority weight of the behavior. [more].
pwt:	Same as priority.
runflag:	A MOOS variable and a value posted when a behavior has met its conditions. [more].
spawnflag:	A MOOS variable and a value posted when a behavior is spawned. [more].
spawnxflag:	A MOOS variable and a value posted when a behavior is spawned. [more].
templating:	Turns a behavior into a template for spawning behaviors dynamically. [more].
updates:	A MOOS variable from which behavior parameter updates are read dynamically. [more].

Listing 1.2: Configuration Parameters for the AvoidObstacle Behavior.

Parameter	**Description**
completed_dist:	Range to contact outside of which the behavior completes and dies. The default is 500 meters.
max_util_cpa_dist:	Range to contact outside which a considered maneuver will have max utility. Section **??**. The default is 75 meters
min_util_cpa_dist:	Range to contact within which a considered maneuver will have min utility. Section **??**. The default is 10 meters.
pwt_inner_dist:	Range to contact within which the behavior has maximum priority weight. Section 1.6. The default is 50 meters.
pwt_outer_dist:	Range to contact outside which the behavior has zero priority weight. Section 1.6. The default is 200 meters.
use_refinery:	If true, the behavior will produce an optimized objective function that is faster to produce, uses a smaller memory footprint, and contributes to faster helm solution time. The default is false, simply for continuity with prior releases, but there is no downside to enabling this feature. Section **??**.
polygon:	A convex polygon representing the obstacle. Section **??**.
poly:	Same as polygon.
rng_flag:	A flag to be posted on all iterations. It may be conditioned on a range threshold. Section **??**.
cpa_flag:	A flag to be posted upon reaching the closest point of approach to the obstacle. Section **??**.
visual_hints:	A request to override the default visual parameters of the rendered obstacle or its buffer region. Section **??**.
allowable_ttc:	The allowable time-to-collision, in seconds, within which a candidate trajectory will begin to be penalized. Section **??**. id:

Listing 1.3: Example Configuration Block.

```
Behavior = BHV_AvoidObstacleV24
{
  // General Behavior Parameters
  // ---------------------------
  name       = avdob_
  pwt        = 300
  condition  = DEPLOY = true
  templating = spawn
  updates    = OBSTACLE_ALERT

  // Parameters specific to this behavior
  // ------------------------------------
       allowable_ttc = 20   // default
      pwt_outer_dist = 50   // default
      pwt_inner_dist = 10   // default
      completed_dist = 60   // default
   min_util_cpa_dist = 8    // default
   max_util_cpa_dist = 16   // default
        use_refinery = true // default is false

    visual_hints = obstacle_edge_color   = white     // default
    visual_hints = obstacle_vertex_color = gray60    // default is white
    visual_hints = obstacle_vertex_size  = 1         // default is white
    visual_hints = obstacle_fill_color   = gray60    // default
    visual_hints = obstacle_fill_transparency = 0.7  // default

    visual_hints = buffer_min_edge_color = gray60    // default
    visual_hints = buffer_min_vertex_color = blue    // default is dodger_blue
    visual_hints = buffer_min_vertex_size = 1        // default
    visual_hints = buffer_min_fill_color = gray70    // default
    visual_hints = buffer_min_fill_transparency = 0.25  // default

    visual_hints = buffer_max_edge_color = gray60         // default
    visual_hints = buffer_max_vertex_color = dodger_blue  // default
    visual_hints = buffer_max_vertex_size = 0             // default is 1
    visual_hints = buffer_max_fill_color = gray70         // default
    visual_hints = buffer_max_fill_transparency = 0.1     // default
}
```

1.4 Variables Published

The below MOOS variables will be published by the behavior during normal operation, in addition to any configured flags. A variable published by any behavior may be supressed or changed to a different variable name using the `post_mapping` configuration parameter described in Section **??**.

- `AVD_OB_SPAWN`: A request to the obstacle manager specifying the conditions for obstacle alerts.
- `NOTED_RESOLVED`: A posting made when the behavior notes an obstacle has been reported be resolved by the obstacle manager.
- `VIEW_POLYGON`: A polygon rendering of either the obstacle, inner, or outer buffer.

1.5 Configuring and Using the AvoidObstacle Behavior

The AvoidObstacle behavior produces an objective function based on the relative position and trajectory between the vehicle and its obstacle.

The objective function is based on applying a utility to the calculated closest point of approach (CPA) for a candidate maneuver. The user may configure a priority weight, but this weight is typically degraded in proportion to increasing range to the obstacle. The behavior may be configured for avoidance with respect to an obstacle known prior to the start of the mission, or it may be configured to spawn a new instance upon demand as obstacles become known through the obstacle manager.

1.6 Specifying the Behavior Priority Weight Policy

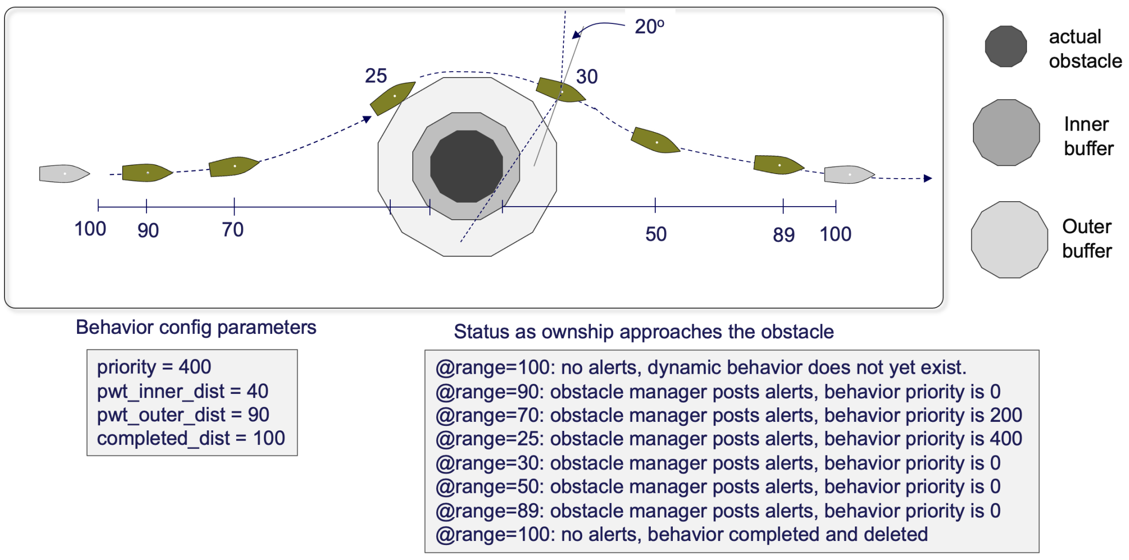
The AvoidObstacle behavior may be configured to increase its priority as it closes range to the obstacle. The priority weight specified in its configuration represents the *maximum* possible priority applied to the behavior, presumably in close range to the obstacle. The range at which this maximum priority applies is specified in the `pwt_inner_dist` parameter. Likewise, the `pwt_outer_dist` parameter specifies a range to the obstacle where the priority weight becomes zero, regardless of the priority weight specified in the configuration file.

So the *current priority* will always be between zero and the maximum priority set in the behavior `priority` configuration parmeter. To be more precise:

Current Priority =

- 0 if current range to obstacle is greater than or equal to `pwt_outer_dist`
- 100 if current range to obstacle is less than or equal to `pwt_inner_dist`
- otherwise ((`pwt_outer_dist` - current range) / (`pwt_outer_dist` - `pwt_inner_dist`)) * priority

This relationship is shown in Figure 3.



Behavior config parameters

```
priority = 400
pwt_inner_dist = 40
pwt_outer_dist = 90
completed_dist = 100
```

Status as ownship approaches the obstacle

```
@range=100: no alerts, dynamic behavior does not yet exist.
@range=90: obstacle manager posts alerts, behavior priority is 0
@range=70: obstacle manager posts alerts, behavior priority is 200
@range=25: obstacle manager posts alerts, behavior priority is 400
@range=30: obstacle manager posts alerts, behavior priority is 0
@range=50: obstacle manager posts alerts, behavior priority is 0
@range=89: obstacle manager posts alerts, behavior priority is 0
@range=100: no alerts, behavior completed and deleted
```

Figure 3: **Scaling priority weights based on ownship range to obstacle**: The range between the vehicle and the obstacle affects whether the behavior is spawned, is active and with what priority weight. Beyond the range specified by `pwt_outer_dist` the behavior will have a zero priority weight, if it even exists. Within the range of `pwt_outer_dist`, the behavior is active with a non-zero priority weight growing as the obstacle comes closer. Within the range of `pwt_inner_dist`, the behavior is active with 100% of its configured priority weight.

The example shown below in Figure **??** shows the effect of the `pwt_outer_dist` parameter. The vehicle on the left is proceeding east, oblivious to the two approaching vessels. The two westbound vessels, `ben` and `cal` are simulated exactly on top of one another. They are oblivious to one another, but will use the collision avoidance behavior to avoid the eastbound vessel, abe. The only difference between `ben` and `cal` is that `cal` begins winding up its priority weight at 80 meters range to `abe`, as opposed to 30 meters for `ben`. The short simulation shows the resulting difference in trajectory.

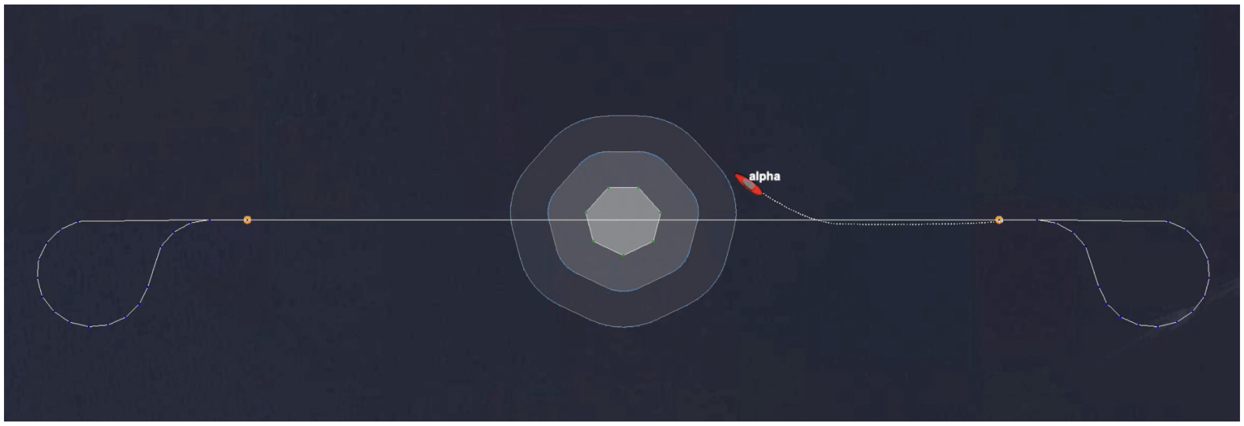


Figure 4: A vehicle approaches an obstacle. The physical obstacle is shown by the inner polygon. The middle inner polygon represents a buffer region around the physical obstacle regarded with the same severity of consequence as hitting the physical obstacle. The outer polygon represents the all clear boundary. The buffer region of the outer polygon has a linearly scaled utility.
`video:(0:15): https://vimeo.com/856588988`

By default, the priority weight decreases linearly between the two depicted ranges. The `pwt_grade` parameter allows the degradation from maximum priority to zero priority to fall more steeply by setting `pwt_grade=quadratric`.

Visual Configuration Options

The visual artifacts generated by the `AvoidCollisionV24` behavior are related to the three polygon regions representing (a) the obastacle, (b) the inner buffer, and (c) outer buffer polygons. There are also seven visual aspects for each polygon that may be modified: (1) the vertex size, (2) the edge size, (3) the vertex color, (4) the edge color, (5) the label color, (6) the polygon fill color, and (7) polygon transparency.

By setting `edge_color=green`, this sets the default edge color for all three polygons. This can be overruled for say the outer buffer polygon by setting `buff_max_edge_color=white`. If the latter is left unspecified, the default edge colore is used. In the example lines below, the first group of lines are the default aspects for all polygons. The second block of lines pertain to the obstacle polygon and overrule the default aspects. The third block of lines pertain to the inner buffer polygon, and the final block of lines pertain ot the outer buffer polygon. All lines below represent the defaults used by the `AvoidCollisionV24` behavior if no visual hints were provided in the behavior configuration.

```
  m_hints.setMeasure("vertex_size", 0);
  m_hints.setMeasure("edge_size", 1);
  m_hints.setColor("vertex_color", "gray50");
  m_hints.setColor("edge_color", "gray50");
  m_hints.setColor("fill_color", "off");
  m_hints.setColor("label_color", "white");

  m_hints.setColor("obst_edge_color", "white");
  m_hints.setColor("obst_vertex_color", "white");
  m_hints.setColor("obst_fill_color", "gray60");
  m_hints.setMeasure("obst_vertex_size", 1);
  m_hints.setMeasure("obst_fill_transparency", 0.7);

  m_hints.setColor("buff_min_edge_color", "gray60");
  m_hints.setColor("buff_min_vertex_color", "dodger_blue");
  m_hints.setColor("buff_min_fill_color", "gray70");
  m_hints.setColor("buff_min_label_color", "off");
  m_hints.setMeasure("buff_min_vertex_size", 1);
  m_hints.setMeasure("buff_min_fill_transparency", 0.25);

  m_hints.setColor("buff_max_edge_color", "gray60");
  m_hints.setColor("buff_max_vertex_color", "dodger_blue");
  m_hints.setColor("buff_max_fill_color", "gray70");
  m_hints.setColor("buff_max_label_color", "off");
  m_hints.setMeasure("buff_max_vertex_size", 1);
  m_hints.setMeasure("buff_max_fill_transparency", 0.1);
```

The posting of the inner and outer polygons can be disabled simply with:

```
  draw_buff_max_poly = false
  draw_buff_min_poly = false
```

1.7 Flags and Macros