uFldCollisionDetect: Detecting Collisions

June 2018

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

1 Overview

The `uFldCollisionDetect` application is run on the shoreside and monitors pairs of vehicles for encounters that come within a certain range. The closest point of approach (CPA) is noted when the range between two vehicles transitions from closing to opening. Depending on the CPA value, one of three events may be declared, either an *encounter*, a *near miss*, or a *collision*, depending on user configured range parameters.

2 Using uFldCollisionDetect

2.1 Setting the Range Thresholds for Events

There are three range threshold parameters that may be set. The first, `encounter_range` is the CPA range beyond which two vehicles are considered to be too far away to be regarded as having had an encounter. Outside this range it's a non-event.

The second parameter, `near_miss_range`, determines a CPA range within which an encounter is considered to be a near miss. Encounters even closer, with the range specified by `collision_range`, are categorized as collisions.

The default values are:

- encounter_range = 20
- near_miss_range = 6
- collision_range = 3

The following relationship between parameters will be enforced:

encounter_range ¿= near_miss_range ¿= collision_range

If this ordering is not respected by the configuration parameters, then the following happens upon startup: (1) If the near_miss_range is smaller than the collision_range, the former is adjusted upward to the latter. (2) If the encounter_range is less than the near_miss_range, then the former is adjusted upward to the latter. (3) A configuration warning is generated and shown in the appcasting output.

Upon each encounter less than or equal to the encounter_range, an encounter counter is internally incremented. Likewise for near misses and collisions. A collision encounter will not however also increment the near miss counter. These counters are maintained globally for all vehicles, as seen in lines 17-19 in the example appcasting output in Listing 3. They are also maintained *per vehicle*, as seen in lines 21-29 in Listing 3.

2.2 Setting Flags to be Posted Upon Events

Flags may be configured to be posted upon each event type - collision, near-miss or encounter. These flags are simply MOOS variable and value pairs like the flags in many other MOOS applications and helm behaviors. There are also macros available that can be filled in a run-time.
The following are the available macros:

- $V1: The name of vehicle 1.
- $V2: The name of vehicle 2.
- $UP_V1: The upper case name of vehicle 1.
- $UP_V2: The upper case name of vehicle 2.
- $IDX: The number of total encounters
- $CPA: The CPA range of the encounter

Note these macros are available for expansion in the contents of the published MOOS message. If the $CPA or $IDX macros are used in isolation in the message, the message will post as a double, not a string. All macros (except $CPA) may also be used in the naming of the MOOS variable. The following are all valid configurations:

```
near_miss_flag  = NEAR_MISS_$UP_V1 = $CPA
encounter_flag  = WARNING = Something happened between $V1 and $V2
collision_flag  = CRITICAL = collision between $UP_V1 and $UP_V2
```

2.3 Applying a Logic Condition

Logic conditions may also be used for disabling/enabling the reporting activities of uFldCollisionDectect. Variables involved in the conditions are automatically subscribed for by uFldCollisionDectect. For example

```
  condition = COLLISION_DETECT=true
  condition = DEPLOY_ALL=true
```

If multiple conditions are specified, they must *all* be true for application to actively report events.

2.4 Configuring Visual Range Pulses to be Generated

Range pulses can be configured to show the user a visual signal that an interaction has occurred. The range pulse can be configured to change both its range and duration using configuration parameters.

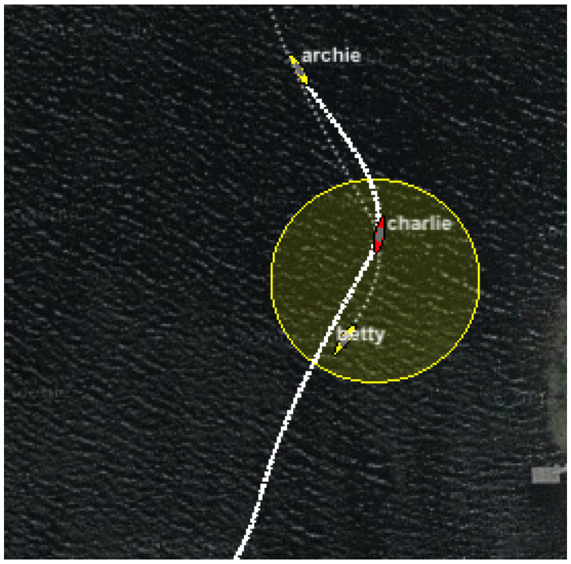

Figure 1: **Using Range Pulse with uFldCollisionDetect:** A range pulse is displayed in pMarineViewer with each detected interaction.

2.5 Ignoring Collisions Between Certain Contacts

In certain missions, it may be desirable to ignore encounters between certain contacts. Perhaps we want to focus on a particular vehicle. Or perhaps there are multiple contacts without collision avoidance enabled (hopefully only in simulation). In any event, there are two configuration parameters for `uFldCollisionDetect` that allow certain encounters to be disregarded. The `ignore_group` parameter and the `reject_group` parameter. Both parameters name one or more vehicle *groups*.

```
ignore_group = commercial
reject_group = sailing,rowing
```

For a vehicle in one of the *ignore* groups, encounters between this vehicle and any other vehicle also in an ignore group, the encounter will be disregarded. However, if the other vehicle is not in an ignore or reject group, the encounter will be included in the analysis. For a vehicle in one the *reject* groups, incoming node reports for this vehicle are rejected on arrival. Any encounter with this vehicle and any other vehicle will not be included in the analysis.

An example is given in Figure 2.5 below:

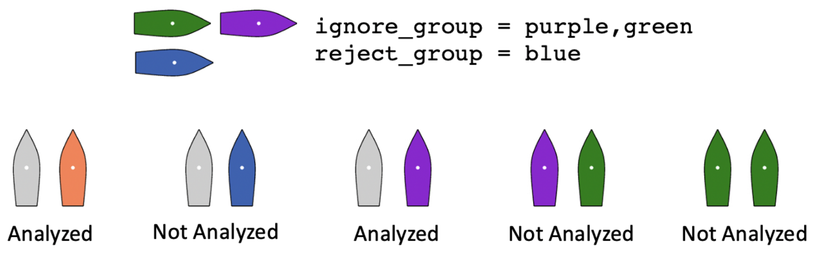

Figure 2: **Filtering out encounters base on group:** An encounter is only analyzed if at least one of vehicles is not in an ignore group *and* both vehicles are not members of a reject group.

The group name for a vehicle comes from the incoming node report, and the group name is set in the configuration of `pNodeReporter` for that vehicle.

2.6 Posting Range Status Messages

On each iteration of `uFldCollisionDetect`, the range between all pairs of vehicles is noted. For that instant in time, the minimum range of all ranges is also noted, and published in the variable:

`UCD_CLOSEST_RANGE`

This variable is only published when configured to do so:

`post_closest_range=true`

Additionally, the minimum range ever noted between any two vehicles is published in the variable:

`UCD_CLOSEST_RANGE_EVER`

3 Configuration Parameters for uFldCollisionDetect

The `uFldCollisionDetect` application may be configured with a configuration block within a MOOS mission file, typically with a `.moos` file suffix. The following parameters are defined for `uFldCollisionDetect`.

Listing 3.1: Configuration Parameters for uFldCollisionDetect.

`collision_flag`:	A MOOS variable and value posted upon the detection of a collision. Section 2.2.
`collision_range`:	The inter-vehicle range, within which the CPA is regarded as a collision. Section 2.1.
`condition`:	A logic condition which if not evaluated to be true, the monitoring of collisions will not be conducted. Section 2.3.
`encounter_flag`:	A MOOS variable and value posted upon the detection of an encounter. Section 2.2.

encounter_range:	The inter-vehicle range, within which the CPA is considered to be an encounter. Section 2.1.
ignore_group:	A comma-separated lists of group names. Encounters for vehicles in one of these groups will be ignored, if the other vehicle is also in one of these groups. Section 2.5.
near_miss_flag:	A MOOS variable and value posted upon the detection of a near miss. Section 2.2.
near_miss_range:	The inter-vehicle range, within which the CPA is regarded as a near miss. Section 2.1.
pulse_duration:	Sets the duration of the visual range pulse posted. Default is 10 seconds. 2.4.
pulse_range:	Sets the range of the visual range pulse posted. Default is 20 meters. 2.4.
pulse_render:	Determines if a visual range pulse will be posted on encounters. Default is true. Section 2.4.
reject_group:	A comma-separated lists of group names. Encounters for vehicles in one of these groups will be completely disregarded, regardless of the group of the other vehicle. Section 2.5.
report_all_encounters:	If true, all encounters will be published in the UCD_REPORT, not just near misses or collisions. The default is false.

An Example MOOS Configuration Block

An example MOOS configuration block is provided in Listing 2 below. This can also be obtained from a terminal window with:

```
$ uFldCollisionDetect --example or -e
```

Listing 3.2: Example configuration of the uFldCollisionDetect application.

```
===============================================================
uFldCollisionDetect Example MOOS Configuration
===============================================================

ProcessConfig = uFldCollisionDetect
{
  AppTick   = 4
  CommsTick = 4

  condition = DEPLOY_ALL = true

  encounter_flag = ENCOUNTER = $CPA
  collision_flag = COLLISION = $CPA
  near_miss_flag = NEAR_MISS = vname1=$V1,vname2=$V2,cpa=$CPA

  encounter_range = 10         // the default in meters
  near_miss_range = 6          // the default in meters
  collision_range = 3           // the default in meters

  ignore_group = alpha
  reject_group = bravo

  pulse_render   = true          // default true
  pulse_range    = 20            // default is 20 meters
  pulse_duration = 10            // default is 10 seconds

  report_all_encounters = true  // default is false
}
```

4 Publications and Subscriptions for uFldCollisionDetect

The interface for uFldCollisionDetect, in terms of publications and subscriptions, is described below. This same information may also be obtained from the terminal with:

```
$ uFldCollisionDetect --interface or -i
```

4.1 Variables Published by uFldCollisionDetect

The user may configure any number of postings (flags) to be generated upon an encounter, near miss or collision.

- APPCAST: Contains an appcast report identical to the terminal output. Reports are posted only in response to an appcast request messages.
- COLLISION_DETECT_PARAMS: The values for encounter_range, near_miss_range, and collision_range are posted upon startup to other applications may use this information. For example:
  `COLLISION_DETECT_PARAMS="collision_range=3,near_miss_range=6,encounter_range=20"`
- ENCOUNTER_TOTAL: Total number of encounters, within encounter_range, regardless of CPA.
- UCD_REPORT: A report generated on each encounter showing which vehicles where involved and the CPA value. For example:
  `UCD_REPORT="cpa=11.2,vname1=henry,vname2=gus,rank=near_miss"`
- UCD_CLOSEST_RANGE: If the post_closest_range parameter is set to true, this variable is published. The value is the minimum range between all currently monitored vehicles.
- VIEW_RANGE_PULSE: A visual artifact generated upon an encounter centered at the mid-point between the two vehicles.

4.2 Variables Subscribed for by uFldCollisionDetect

The uFldCollisionDetect application will subscribe for the following variables:

- APPCAST_REQ: A request to generate and post a new apppcast report, with reporting criteria, and expiration.

- NODE_REPORT: A report on a vehicle location and status.

Additionally, any variable used in a logic condition will also be subscribed for. See Section 2.3 for more on the use of logic conditions.

Command Line Usage of uFldCollisionDetect

The uFldCollisionDetect application is typically launched as a part of a batch of processes by pAntler, but may also be launched from the command line by the user. To see command-line options enter the following from the command-line:

```
$ uFldCollisionDetect --help or -h
```

This will show the output shown below.

```
Usage: uFldCollisionDetect file.moos [OPTIONS]

Options:
  --alias=<ProcessName>
      Launch uFldCollisionDetect with the given process
      name rather than uFldCollisionDetect.
  --example, -e
      Display example MOOS configuration block
  --help, -h
      Display this help message.
  --interface, -i
     Display MOOS publications and subscriptions.
  --version,-v
      Display the release version of uFldCollisionDetect.
```

5 Terminal and AppCast Output

The uFldCollisionDetect application produces some useful information to the terminal and identical content through appcasting. An example is shown in Listing 3 below. On line 2, the name of the local community, typically the shoreside community, is listed on the left. On the right, "0/0(2769) indicates there are no configuration or run warnings, and the current iteration of uFldCollisionDetect is 2769. Lines 4-11 show the how the application was configured at launch time.

Lines 13-19 convey the the overall application state, whether or not the application is actively assessing encounters (line 16), and the total number of encounters, near missions and collisions (lines 17-19).

Listing 5.3: Example terminal or appcast output for uFldCollisionDetect.

```
1  ===================================================================
2  uFldCollisionDetect shoreside                             0/0(2769)
3  ===================================================================
4  Configuration:
5  ============================================
6         encounter_dist: 30.00
```

```
 7         collision_dist: 8.00
 8         near_miss_dist: 12.00
 9     range_pulse_render: true
10   range_pulse_duration: 30.00
11      range_pulse_range: 20.00
12
13  ============================================
14  State Overall:
15  ============================================
16               Active: true
17     Total Encounters: 19
18    Total Near Misses: 0
19     Total Collisions: 0
20
21  ============================================
22  State By Vehicle:
23  ============================================
24  Vehicle  Encounters  Near Misses  Collisions
25  -------  ----------  -----------  ----------
26  abe      9           0            0
27  ben      9           0            0
28  cal      10          0            0
29  deb      10          0            0
30
31  ===================================================================
32  Most Recent Events (8):
33  ===================================================================
34  [806.84]: ben::deb, cpa=22.79
35  [783.85]: abe::ben, cpa=18.26
36  [757.90]: cal::deb, cpa=29.16
37  [638.36]: cal::deb, cpa=27.73
38  [629.30]: ben::deb, cpa=16.85
39  [611.36]: ben::cal, cpa=28.97
40  [583.23]: ben::deb, cpa=19.33
41  [578.27]: abe::ben, cpa=17.02
```

The state is further broken down by vehicle in lines 21-29. The total number of encounters, near misses and collisions per vehicle is shown. In the final portion of the the appcast output are the 8 most recent events. Each encounter is an event, and the cpa distance for is also shown.