Lab #14

Autonomous Collaborative Search Part II

2.S998 Unmanned Marine Vehicle Autonomy, Sensing and Communications

Contents

1 Overview and Objectives

1.1 Structure of the Lab and Goals

In this lab we return to the problem of collaborative search addressed in Lab #9. A few features of the simulation and ground rules are changed in this lab. Unlike the previous lab where participants were graded on a pass-fail basis, in this lab the grading will be partly based on (a) implementation, (b) competition results, and (c) a short report and in-class presentation held after the competition.

1.2 Preliminaries

Standard practice now before any lab should be to perform an svn update within your moos-ivp tree and rebuild if you see any updates pulled down from the SVN server. This lab does assume that you have a working MOOS-IvP tree checked out and installed on your computer. To verify this make sure that the following executables are built and findable in your shell path:

```
$ which MOOSDB
/Users/you/moos-ivp/MOOS/MOOSBin/MOOSDB
$ which uTimerScript
/Users/you/moos-ivp/bin/uTimerScript
$ which mykill
/Users/you/moos-ivp/scripts/mykill
```

If unsuccessful with the above, see the 2.S998 wiki page for help in getting started:

http://oceanai.mit.edu/2.S998

1.3 An Example Hazard File

An example hazard file may be downloaded from the web:

```
% wget http://oceanai.mit.edu/2.S998/examples/hazards_lab14.txt
```

Or if you prefer, a similar file may be auto-generated by:

```
% gen_hazards --polygon=-150,-75:-150,-400:400,-400:400,-75 --objects=12,hazard
  --objects=600,benign --exp=4
```

The hazard file used in the lab may be substantially different however in terms of the ratio of hazards to benign objects, the distribution of hazard resemblance factors, the number of overall objects, or all of the above.

1.4 More Resources on Sensing and Inter-Vehicle Comms for Today's Lab

The one document that may be most useful to the discussion in this lab is the documentation for the uField Toolbox linked below. See also the slides from today's lecture.

- Today's lecture notes.

- The uField Toolbox Documentation
  http://oceanai.mit.edu/moos-ivp-pdf/moosivp-ufield.pdf

  This contains the documentation for the `uFldHazardSensor`, `uFldMessageHandler`, and `uFldNodeComms` applications. All three applications are used in this lab for the first time.

- The IvP Helm documentation.
  http://oceanai.mit.edu/moos-ivp-pdf/moosivp-helm.pdf

- The moos-ivp.org website documentation.
  http://www.moos-ivp.org

2 Collaborative Search

2.1 Overall Mission Description and Tips for Getting Started

The goal of this mission is to deploy two marine vehicles collaborating to detect and classify shallow-water hazards. Each vehicle will be able to use a simulated hazard sensor, `uFldHazardSensor`. The autonomy mission, in terms of waypoints, sensor settings, strategies for inter-vehicle communications, and maximizing mission reward is completely up to each team. You are encouraged to work in teams of two.

The Hazard File

The hazard field is a file generated by the `gen_hazards` command line application built as part of the moos-ivp tree. You can generate output by:

```
% gen_hazards --polygon=-150,-75:-150,-400:400,-400:400,-75 --objects=5,hazard --objects=8,benign, --exp=3
// Proper polygon specified: -150,-75:-150,-400:400,-400:400,-75
hazard = x=64.1,y=-358.2,type=hazard
hazard = x=2.5,y=-153.3,type=hazard
hazard = x=145.9,y=-352.9,type=hazard
hazard = x=267.8,y=-250,type=hazard
hazard = x=37.7,y=-394.8,type=hazard
hazard = x=-118.7,y=-155.3,hr=0.22051,type=benign
hazard = x=222,y=-312.4,hr=0.00051,type=benign
hazard = x=31.7,y=-365.3,hr=0.02351,type=benign
hazard = x=76.8,y=-332.9,hr=0.45121,type=benign
hazard = x=-78.1,y=-381,hr=0.67831,type=benign
hazard = x=242.6,y=-340.9,hr=0.31111,type=benign
hazard = x=-63.8,y=-156.2,hr=0.24988,type=benign
hazard = x=223.2,y=-255.4,hr=0.65666,type=benign
```

You may redirect the above output to a file, by appending `"> file.txt"` to the above command-line invocation.

The Baseline Mission

The recommended baseline mission for this lab is the mission folder created for `lab09two`. The fundamentals in terms of operation area, number of vehicles and region of hazards will not change in this lab.

The name you should give to this mission folder for handing in will be `lab14`.

2.2 Application Modules Important to this Lab

The following applications are important to this lab, including two that you will need to write yourself:

The uFldHazardSensor Application

This tool is your simulated sensor. It runs in the shoreside MOOS community. During in-lab demonstrations, it will be launched and configured by the TAs, but you will need to have your own

shoreside community configured during your own development and testing. It is described in the uField Toolbox documentation found on the course website. The baseline mission provided for this lab also makes use of this application.

The uFldMessageHandler Application

This application runs on each of the vehicles. It is used for unparsing inter-vehicle messages. You should be fairly familiar with this now as it was the focus of Section **??** in the first part of this lab.

The uFldNodeComms Application

This application runs on the shoreside computer. It is used for routing messages between vehicles. You should be fairly familiar with this now as it was the focus of Section **??** in the first part of this lab.

The pHandleSensorData Application

You are free to implement your solution however you see fit. However, it is likely that you will need an application that simply manages the output of your sensor information, updates it with new sensor information from subsequent passes and other vehicles, and reports it to the shoreside when the mission is complete. The recommended name for such an app or similar is `pHandleSensorData`. This will help us understand what is going on in your implementation when/if we need to help you, and when grading your handed in assignment.

Note: One function of the above app may be simply to ensure that `UHZ_SENSOR_REQUEST` is being published and bridged to the shoreside. Recall that this message must be received from the vehicle regularly in order for the simulated sensor to generate reports to that vehicle.

The pHazardPath Application

You are free to implement your solution however you see fit. However, it is likely that you will need an application that reasons about your vehicle's waypoints. The recommended name for such an app or similar is `pHazardPath`. This will help us understand what is going on in your implementation when/if we need to help you, and when grading your handed in assignment.

2.3 Ground Rules

The following ground rules are relevant to this lab.

2.3.1 Operation Area

The simulated hazard field will be within a polygon with the following four vertices:

- (-150, -75)
- (-150, -400)
- (400, -400)
- (400, -75)

The region was chosen to be rectilinear to simplify the lawnmower pattern generation for a waypoint behavior. See the documentation for this behavior. A pattern may be given to the behavior simply by specifying the characteristics of the lawnmower pattern without having to explicitly determine all the points.

2.3.2 Time and Top Speed

You mission will will have a maximum duration of 150 minutes, or 9000 seconds. The plan is to test at 30x real time in the labs. So we should be able to see each team perform its full mission in 5 minutes of real time.

The top vehicle speed for each vehicle will be 2.0 meters per second. This will be monitored on the shoreside. If a vehicle is noted to go over the maximum speed a penalty may be invoked in the form of cutting off access to the sensor simulator, or inter-vehicle messaging, or both.

2.3.3 Sensor Configuration Frequency

You will be allowed to reset your sensor configuration once every 20 minutes, per vehicle. Attempts to switch configurations more frequently will simply be ignored.

Your sensor configuration options will be:

- sensor_config = width=10, exp=6, class=0.93, max=1
- sensor_config = width=25, exp=4, class=0.80
- sensor_config = width=50, exp=2, class=0.60

See the documentation for `uFldHazardSensor` and the class notes on what this entails in terms of the ROC curve performance of the sensor, and how the vehicle may request a particular configuration.

2.3.4 Scoring Metrics

The scoring metrics are as follows (three categories eliminated from lab 09):

- +10: Hazard identified correctly as a hazard
- -50: Hazard not report as a hazard (reported as benign or not reported at all)
- -25: Benign guessed incorrectly as a hazard

Your job is to report, before the end of the mission time, a single hazard report message of the following form:

```
HAZARD_REPORT = "source=archie # x=45,y=88,type=hazard,label=04 #
                x=15,y=18,type=benign,label=16"
```

Note there is no incentive to report benign objects. For the purpose of maximizing your score, you need only report objects you believe are hazards.

There is a class called `XYHazardSet` that may be used as follows for easily constructing this report:

```
#include "XYHazardSet.h"

XYHazardSet my_hazard_set;

my_hazard_set.setSource("archie");
my_hazard_set.addHazard(45,88,"hazard","04");
my_hazard_set.addHazard(15,18,"benign","16");

string my_report = my_hazard_set.getSpec();

m_Comms.Notify("HAZARD_REPORT", my_report);
```

Messages reported late are ignored. You must ensure this variable is bridged to the shoreside community with an entry to `uFldNodeBroker`.

2.3.5 Inter-Vehicle Communications

In this lab, you will have the choice between the comms restrictions available in Lab 09, and an alternative configuration. The restrictions in Lab 09 were:

- **Range:** Communications limited to 100 meters.
- **Frequency:** Unlimited.
- **Bandwidth:** Unlimited.

The alternative configuration available in this lab is:

- **Range:** Communications limited to 250 meters
- **Frequency:** Communications will be limited to once per 20 seconds
- **Bandwidth:** Each message will be limited to 100 characters

These configurations are enforced in the `uFldNodeComms` configuration. This module is on the shoreside and will be configured by the TAs during testing, but you should configure your own `uFldNodeComms` module for this restriction during your development. Remember that a message that fails to be sent, due to one of the above criteria, will not be queued for later re-sending. You should handle the possibility that message may be dropped. Prior to in-class competition, you should inform the TA which comms configuration you prefer.

2.3.6 Collision Avoidance

We have not yet covered the issue of inter-vehicle collision avoidance, or means for ensuring operation without hitting land. You will not be penalized for either type of collision in this lab.

2.3.7 Depth

Depth is not a factor in this sensor simulator, so you can assume the simulation of a surface vehicle for this lab. Your helm need not be configured for depth operations.

2.4 Due Date and In-Lab Testing

This lab is due in two parts. The first part is due Thursday April 12th:

- Demonstrating your mission to a TA in lab, with the TA providing the shoreside.
- Uploading your source code.

During class on April 19th, teams will each make a short in-class presentation describing their strategy, lessons-learned, ideas for continuation. A separate hand-in is due on this date:

- Upload a PDF of your in-class presentation.

Your assignment will not be considered submitted until you have completed all three. The PDF file handed in should be named lab14-slides-smith-jones.pdf, containing both partner names.

3 Substantial Simulation Changes Since Lab 09

There are three substantial changes to the way simulation is conducted in this lab compared to Lab 09:

- **Classification Processing:** The classification algorithm is not automatically (and instantaneously) invoked in each hazard report. The user/vehicle must decide, for which detections, to invoke the classification algorithm. And this algorithm takes some time for each request.
- **Hazard Resemblance Factor:** Each benign object in the simulator's hazard file has an associated *resemblance factor*. A lower factor is more likely to be passed over during a detection algorithm.
- **One High-Resolution Vehicle:** Only one vehicle per team will be able to configure its hazard sensor to the highest resolution. This is to encourage cooperation and communication between vehicles.

These issues are explained more fully below.

3.1 Classification Processing

In the previous lab, each sensor response to the vehicle consisted of both a detection and a classification. Driving the vehicle through the hazard field, repeatedly publishing to the sensor `UHZ_SENSOR_REQUEST`, the sensor may generate a reply for each object in the field that comes into the sensor swath. A typical response would be something like:

```
UHZ_HAZARD_REPORT = x=45,y=88,type=hazard,label=04
```

In this lab, the `uFldHazardSensor` has been modified to only return the detection information initially. The initial responses by the sensor will instead look something like:

```
UHZ_DETECTION_REPORT = x=45,y=88,label=04
```

Note that the response variable name is `UHZ_DETECTION_REPORT`, and it contains no classification information.

To access classification information, a further request must be made to the sensor for this information by posting the following (which also needs to be bridged to the shoreside):

```
UHZ_CLASSIFY_REQUEST = vname=archie,label=04
```

Only after the sensor receives this request will it perform the classification algorithm on the detected object with the given label. classification will take some time, in this lab 30 seconds, after which the results will be immediately posted to the vehicle in the form of a hazard report:

```
UHZ_HAZARD_REPORT = x=45,y=88,type=hazard,label=04
```

Re-setting the Queue

The only way to affect the queue at run-time is to clear it of all prior classification requests. This may be done by posting to the variable:

```
UHZ_SENSOR_CLEAR = vname=henry
```

Motivation

The motivation for this change is that it may more properly reflect the realities of an on-board classification algorithm which may be CPU intensive. The idea was conveyed in the lecture notes.

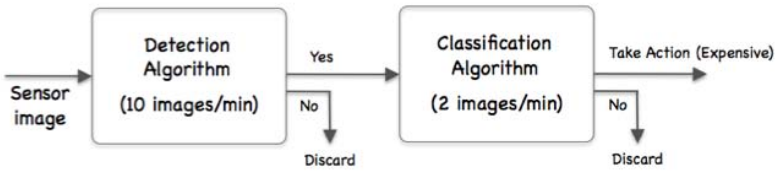

Figure 1: **Detection and Classification algorithms**

Notes

- A classification report will be handled by the sensor for each detection made. If you request N classification reports for D detections, you will receive at most D replies. The remaining $N-D$ requests will simply be ignored (but they will not consume any time from the classification handling queue).
- The sensor simulator will queue successive classification queries. The replies will be sent to the requesting vehicle as they become ready.
- There is no way to adjust, re-prioritize, or get a status of the queue dynamically. If you flood the sensor with classification requests early, your only option is to clear the queue completely.
- The simulated sensor is implemented to simulate multiple sensors for multiple vehicles simultaneously, thus it maintains a distinct classification queue for each vehicle.

3.2 Hazard Resemblance Factor

Hazard field objects may be configured a bit differently in this lab. Benign objects may contain an additional *resemblance factor* in the range of $[0, 1]$, where zero indicates the object shares very little resemblance to a hazard, and vice versa for values of one. Each benign object i in the hazard file has an associated resemblance factor R_i.

The `uFldHazardSensor` app publishes visual markers as before indicating hazards and benign objects. It renders a low resemblance factor for an object by posting the marker with higher transparency. The rendering may look something similar to that in Figure 2.

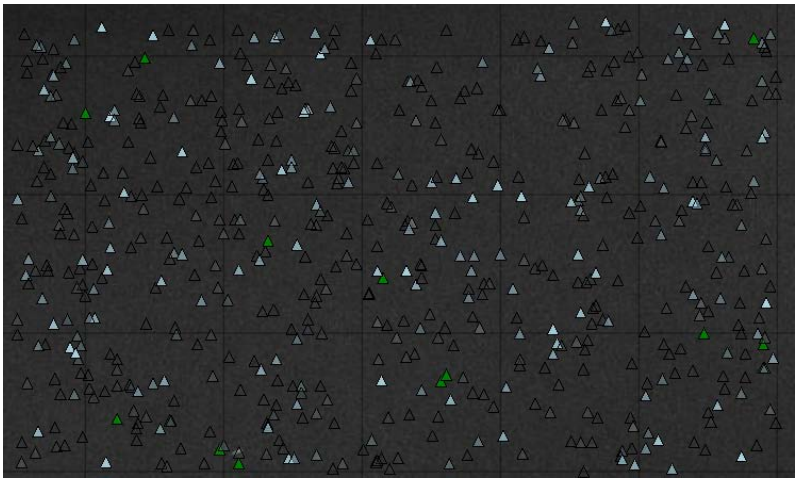

Figure 2: **A simulated hazard field with resemblance factors:** Green markers represent hazards. Light blue markers represent benign objects. The more transparent the markers, the lower the resemblance factor. The resemblance factor ranges from $[0, 1]$.

Influence of Resemblance Factor on P_{FA}

The probability of a detection declared on a benign object (a false alarm), is normally given by P_{FA}. The hazard resemblance factor is further applied to provide a unique probability of false alarm for a given object i.

$$P_{FA}(i) = (P_{FA} + R_i)/2$$

The exception is when $P_{FA} = 1$. In this case $P_{FA}(i)$ is also 1.

Influence of Resemblance Factor on P_C

The probability of a correct classification on a given object is normally given by P_C, in the range of $[0.5, 1]$. The hazard resemblance factor is further applied to provide a unique probability of correct classification for a given object i.

$$P_C(i) = P_C + (1 - P_C)(1 - R_i)$$

In short, $P_C(i)$, at worst, is given by P_C for objects closely resembling a hazard ($R_i = 1$), and P_C approaches 1, i.e., perfection, for objects that look nothing like a hazard ($R_i = 0$).

3.3 Optional One High Resolution Sensor Vehicle

In this lab, teams have the option of utilizing a high-resolution sensor setting *for one vehicle*. This is to promote collaboration and or cueing between vehicles. This optional sensor setting has a swath width of 5 meters, an exponent factor (defining its ROC curve) of 8, and a probability of correct classification of 0.98. It may be enabled for your testing by adding the following line in the `uFldHazardSensor` configuration block of your shoreside MOOS file:

```
sensor_config = width=5,  exp=8, class=0.98, max=1
```

The last component of this configuration line simply indicates that at most one vehicle connected to the simulator may have this sensor setting.

2.S998 Marine Autonomy, Sensing and Communications

Spring 2012