# Lab #16

# Behavior Construction for Autonomous Front Estimation

# Part II: In-water Collaborative Deployment

2.S998 Unmanned Marine Vehicle Autonomy, Sensing and Communications

## Contents

# 1 Overview and Objectives

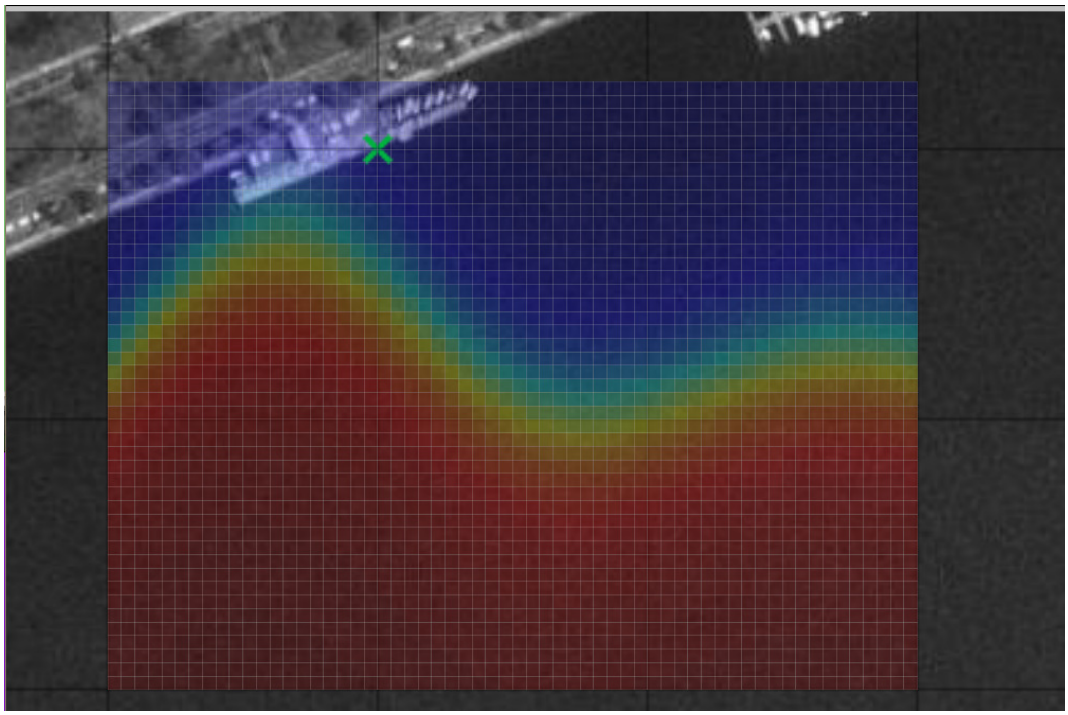

Figure 1: **Ocean Front Estimation:** An autonomous vehicle with a temperature sensor is used to estimate the dynamic properties of an ocean front, separating two water bodies of difdferent temperatures, indicated by the color shading. The front is dynamic, and the objective is to estimate the parameters in an analytical model of the temperature versus time and location

In this Lab we will continue to improve our sampling strategies for assessing the properties of an ocean front. The front is parameterized by the same analytical function as in the previous Lab, but we have moved it into the Charles River, and instead of running in pure simulation, you will implement your code and configuration on a Kingfisher ASC. Since ocean fronts are rather rare in the Charles, we will continue to simulate the temperature sensors on shoreside, and you will obviously be restricted to operate in real time, i.e. with warp=1. On the other hand you will be working with a partner, and the objective of the lab is to upgrade your sampling behaviors to take advantage of this and achieve a parameter estimate in a shorter time. Your vehicles will be able to communicating with each other if within a maximum communication range.

## 1.1 Structure of the Lab and Goals

This lab with stretch over three lab sessions. In the first session, you will work with your partner on developing a collaboration strategy and modify your sampling behaviors accord-

ingly. We will then in the second session work on implementing your autonomy system on the Kingfishers, and in the last session we will finish with a competition. Instead of scheduling Lab presentations later, we will instead ask each team to narrate the mission while it is running. We will set up a projector for your presentation slides.

## 1.2 Preliminaries

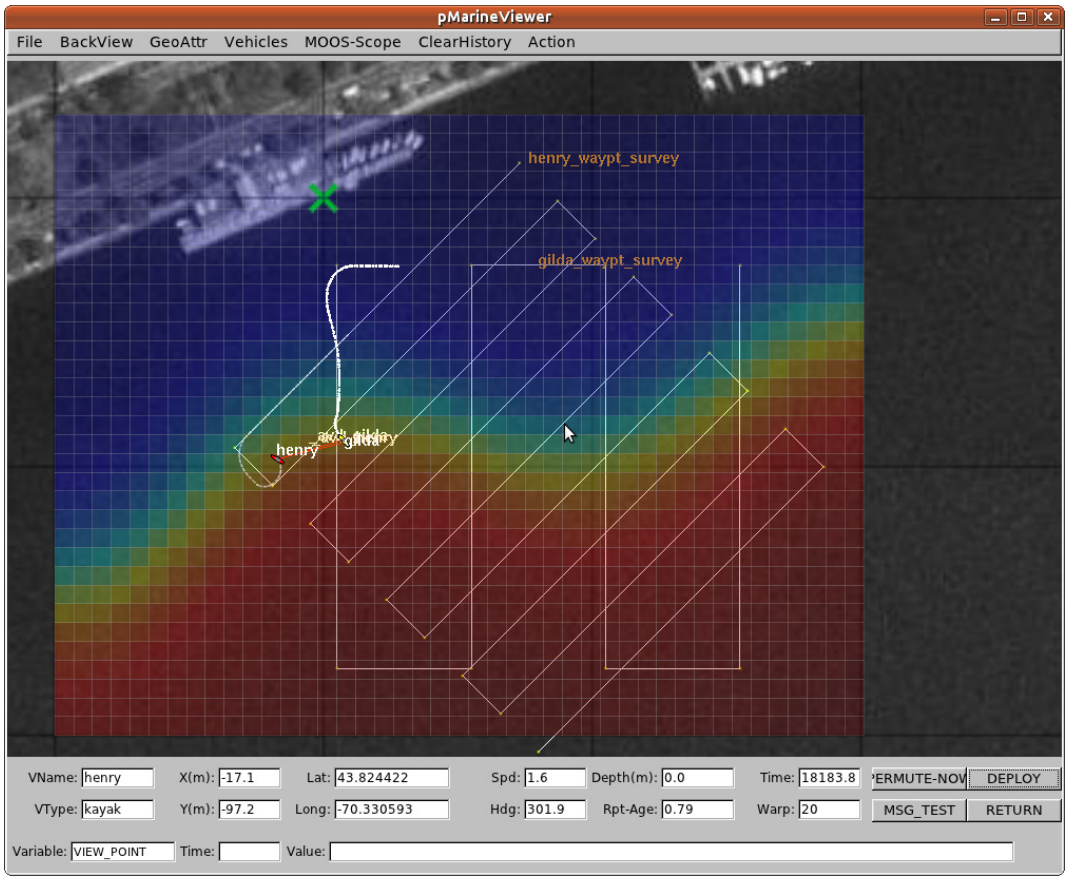


Figure 2: **Dual Vehicle Survey:** Two surveying vehicles with communication and collision avaoidance

The shoreside and vehicle configuration you should use as template for the frontal estimation problem is available in the class repository:

```
moos-ivp-12.2mit/ivp/missions/u7_beta
```

To execute the template lawnmower survey missions with two vehicles, shown in Fig. 2 and the shoreside community, use the command:

```
> ./launch.sh --warp=15 -c -a
```

where the switches represent:

**warp** Time warp factor as usual. You should be able to do at least 10 here.

**-c** Perform the annealing concurrently on both vehicles

**-a** Launch two vehicles, *henry* and *gilda*

### 1.2.1 Scoring

The scoring will be based on the RMS error and the elapsed time of the survey, using the following algorithm:

$$\text{score} = 100/(\gamma_t \epsilon) \tag{1}$$

where $\epsilon$ is the RMS error of the model parameters, and $\gamma_t$ is a time penalty factor,

$$\gamma_t = \begin{cases} 1\ , & t \leq t_0 = 500 \\ 1 + (t - t_0)/t_0\ , & t > t_0 \end{cases} \tag{2}$$

# 2 Application Topology, Comms and Operation Area

In this lab, although we will be working from the MIT Sailing Pavilion with physical platforms, the application topology is identical to the in-class simulations and competitions. A single shoreside computer will serve to both broker the intervehicle communications, and provide the simulated environmental sensor.

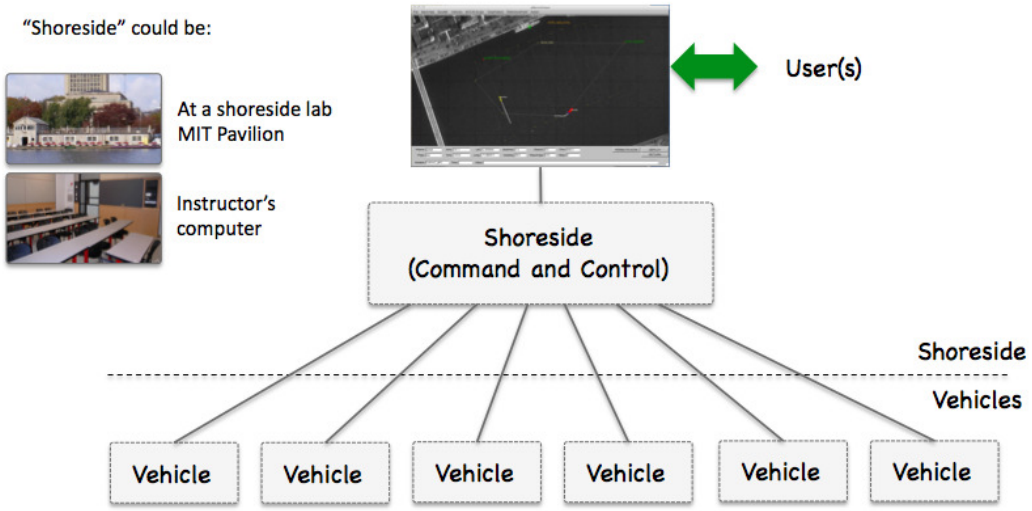


Figure 3: **The Vehicle / Shoreside Topology**

The baseline mission in `moos-ivp-12.2mit/ivp/missions/u7_beta` is configured with three vehicle communities in simulation. Part of your assignment includes the conversion of this mission to a vehicle mission in your own tree(s). This will include at least the following;

- Replacing the `uSimMarine` simulator with the components needed for navigation and control. On the Kingfishers this includes using instead, `iGPS_5hz` for position localization, `iOS5000` for compass/heading information, and `iActuationKF` for access to the Kinfisher actuation (thrusters).
- Use of the shoreside IP address rather than your laptop localhost IP address. For this lab you can assume that the shoreside machine will be on the local wireless router at the Pavilion (SSID of CPRVIP) with an IP address of 10.25.0.5 and port 9000. This may change, and Alon will have the changes if so.
- Remove all references to MOOS Time Warp.
- Modify your launch script such that it is appropriate for running on the vehicle. That is, it does not try to launch the shoreside, and launches the appropriate vehicle.
- Your vehicles should utilize the BHV_Region behavior to keep the vehicles from straying out of the operation area. See the documentation for this behavior. It also provides information about how far the vehicle presently is from the op region in terms of time

and distance. It is recommended that if your vehicle gets too close to the op region boundary that you take corrective action. The op region behavior is implemented to shut down the helm (sit dead in the water) when/if it violates the op region constraint.

### Operation Area

Due to Wi-Fi coverage constraints, the operation area depicted in Figure 4 should be used. In addition to using this information for your mission planning, you may configure your BHV_OpRegion behavior with these coordinates.

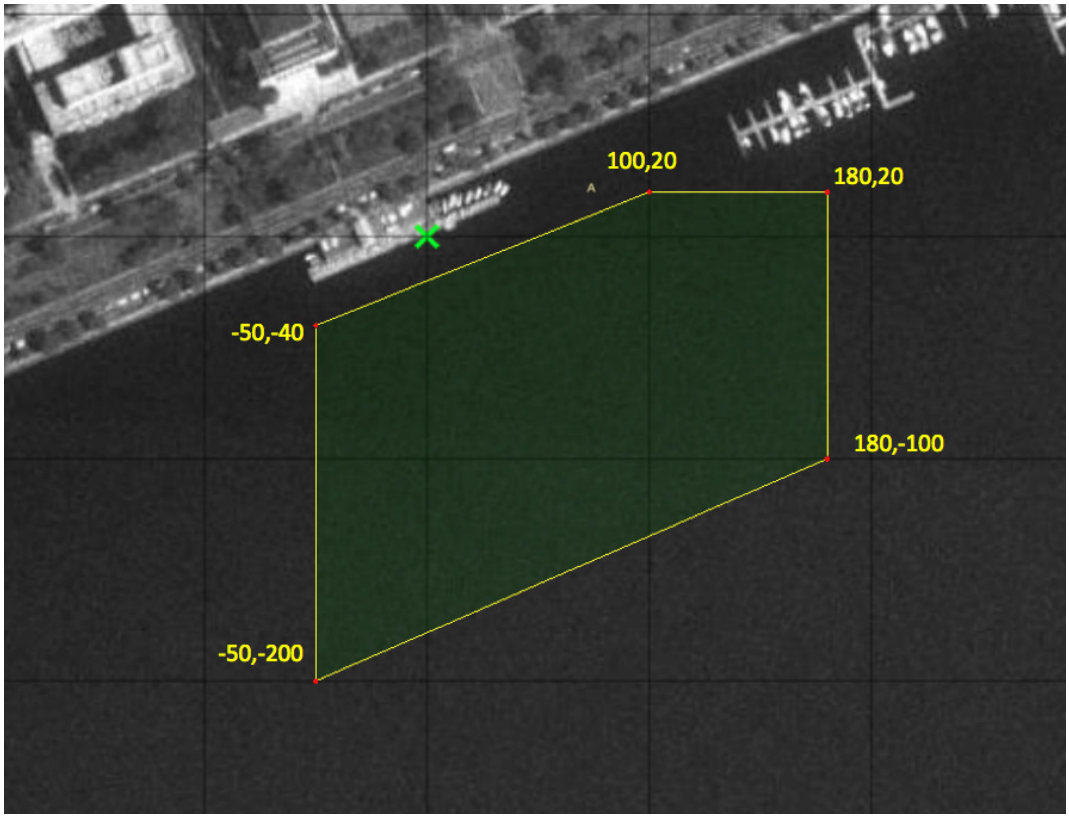


Figure 4: **The Pavilion Op-Area:**

# 3 Front Estimation

## 3.1 Front Model

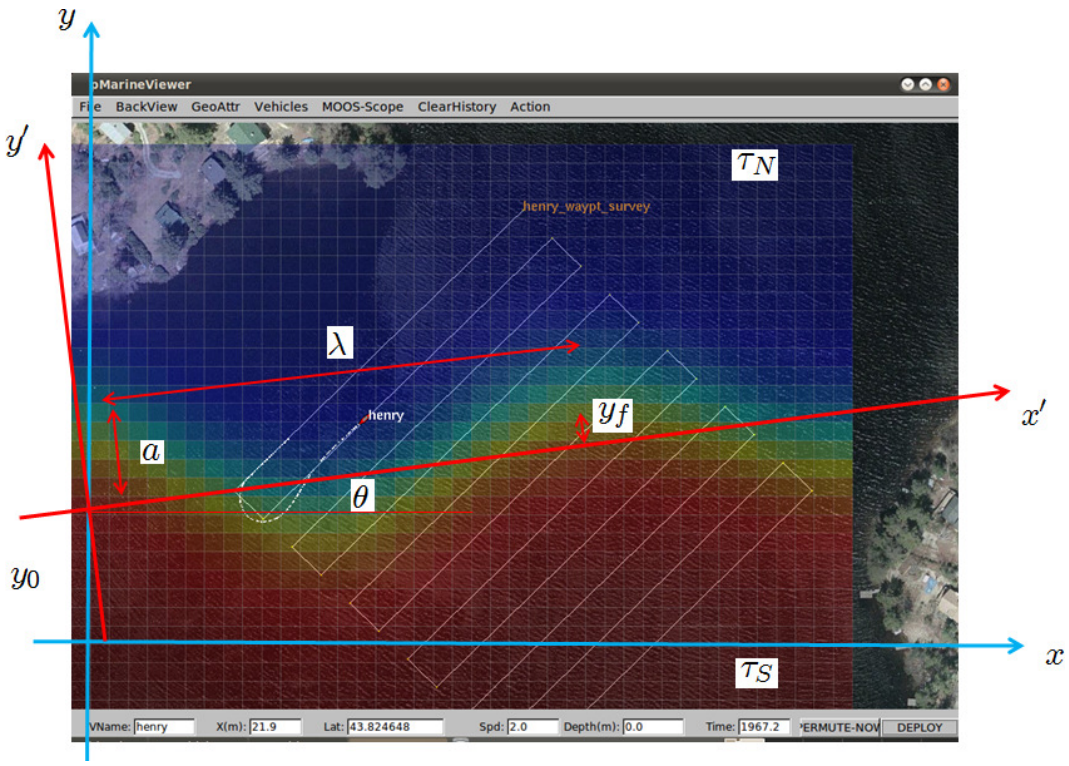


Figure 5: **Ocean Front Estimation:** The temperature field is parameterized by an analytical model with 9 parameters. In addition to the geometrical parameters shown here, the parameters include the temporal period $T$, the length scale of the frontal gradient, $\beta$, and the propagation decay $\alpha$

The temperature field is modeled by an analytical function, defined as follows. The coordinate system aligned with the front, shown in Fig. 5 is defines as

$$x' = x\cos\theta + (y - y_0)\sin\theta \quad (3)$$
$$y' = (y - y_0)\cos\theta - x\sin\theta. \quad (4)$$

The position of the center of the front in the front coordinate system is

$$y_f = a\exp(-x'/\alpha)\sin(kx' - \omega t), \quad (5)$$

where $k = 2\pi/\lambda$ is the wavenumber and $\omega = 2\pi/T$ is the radial frequency. The temperature field is then given by

$$\tau = \tau_0 + d\tau\tan^{-1}(y' - y_f)/\beta, \quad (6)$$

where

$$\tau_0 = (\tau_N + \tau_S)/2 \quad (7)$$
$$d\tau = (\tau_N - \tau_S)/\pi. \quad (8)$$

The sensor model used by the vehicles is `uFldCTDSensor` which simulates the front and takes virtual measurements of the associated temperature field using the class `CFrontSim`. The sensor simulator is operated in the shoreside community. A vehicle requests the measurement by publishing the variable `UCTD_SENSOR_REQUEST` to the MOOSDB with the value `vname=my_auv`. In the template configuration the measurement request are scheduled by the process `uTimerScript`, with the sampling period set in the plug `plug_uTimerScript.moos`. When you develop your behavior you may use the scheduler or directly issue the requests from your behavior. The measurement is returned in the MOOS variable `UCTD_SENSOR_REPORT` with the content

```
vname=my_auv,utc=123456789.0,x=123.45,y=345.67,temp=22.34
```

You control the dynamics of the frontal simulator by setting the ground truth parameters in the plug `plug_uFLDCTDSensor.moos`:

```
//--------------------------------------------------
// uFldCTDSensor configuration block from plugin

ProcessConfig = uFldCTDSensor
{
  AppTick   = 3
  CommsTick = 3

   // Configuring Model of Dynamic Front
  xmin  = 0;
  xmax  = 500;
  ymin  = -400;
  ymax  =  0;

  offset = -100;              // y_0
  angle  = 12;                // front angle theta
  amplitude = 20;            // spatial amplitude a
  period = 350;              // temporal period T
  wavelength = 200;          // spatial wavelength lambda
  alpha = 500;               // spatial 1/e length scale alpha
  beta = 20;                 // length scale of frontal gradient beta
  temperature_north = 20;    // temperature North of front
  temperature_south = 24;    // temperature South of front
  sigma = 0.01;              // standard deviation of gaussian sensor noise
}
```

## 3.2 Parameter Estimation

On your vehicle you will run the parameter estimation process *pFrontEstimate* which subscribes to the sensor report. It must be configured with the minimum and maximum of all the frontal model parameters in the plug *plug_pFrontEstimate.moos*:

```
ProcessConfig = pFrontEstimate
{
  AppTick    = 4
  CommsTick  = 4

  vname = $(VNAME)
  temperature_factor = $(COOL_FAC)
  cooling_steps = $(COOL_STEPS)
  min_offset = -120;
  max_offset = -60;
  min_angle = -5;
  max_angle = 15;
  min_amplitude = 0;
  max_amplitude = 50;
  min_period = 200;
  max_period = 600;
  min_wavelength = 100;
  max_wavelength = 500;
  min_alpha = 500;
  max_alpha = 500;
  min_beta = 10;
  max_beta = 30;
  min_T_N = 15;
  max_T_N = 25;
  min_T_S = 20;
  max_T_S = 30;
  concurrent = true           // Flag controlling whether the
                              // annealing is performed concurrently
                              // with survey.
}
```

The cooling parameters are set in the launch script or on the command line. Note the parameter *concurrent* which is used to control whether the annealing is performed concurrently with the survey. This feature can be used to save time, but it has to be used with care, in particular if combined with fast cooling (cooling_steps small), because the parameters may 'freeze' before the survey reaches the areas which provide the most information.

Note also that you may 'fix' one or more variables by simply setting the min and max values equal to the true value. This is useful for determining the utility of a survey strategy for determining a specific parameter or the coupling between a couple of parameters, as you will be ask do do in the assignments.

The measurement collection and the parameter estimation is initiated when the survey flag is set in the MOOSDB:

```
SURVEY_UNDERWAY = true
```

which must be set by the survey behavior. You can see how this is done with the *active_flag* in the meta file for the vehicle helm, *meta_vehicle.bhv*:

```
//--------------------------------------------------
// Helm Behavior file

initialize   DEPLOY  = true
initialize   RETURN  = false
initialize   STATION_KEEP = false
initialize   SURVEY  = true
initialize   AVOID   = true
initialize   SURVEY_UNDERWAY = false

set MODE = ACTIVE {
  DEPLOY = true
} INACTIVE

set MODE = RETURNING {
  MODE = ACTIVE
  RETURN = true
}

set MODE = SURVEYING {
  MODE = ACTIVE
  SURVEY = true
  RETURN = false
}
//--------------------------------------------
Behavior = BHV_Waypoint
{
  name      = waypt_survey
  pwt       = 100
  condition = MODE==SURVEYING
  perpetual = true
  updates   = SURVEY_UPDATES
  activeflag   = SURVEY_UNDERWAY = true
  inactiveflag = SURVEY_UNDERWAY = false
  endflag   = RETURN = true

//     cycleflag = SURVEY = false
//        repeat = 1
          lead = 8
   lead_damper = 1
         speed = 2.0   // meters per second
        radius = 8.0
        points = format=lawnmower, label=dudley_survey, x=$(SURVEY_X), y=$(SURVEY_Y), \
                 width=$(WIDTH), height=$(HEIGHT), lane_width=$(LANE_WIDTH),          \
                 rows=north-south, degs=$(DEGREES)
  visual_hints = nextpt_color=red, nextpt_lcolor=khaki
```

```
  visual_hints = vertex_color=yellow, line_color=white
  visual_hints = vertex_size=2, edge_size=1
}
```

### 3.2.1 Adding Measurements from Collaborator

In this lab you have the possibility of adding measurements made by another vehicle. The class used for each measurement has the structure

```
class CMeasurement
{
 public:
  double t;
  double x;
  double y;
  double temp;
};
```

where $t$ is the time of the measurement, $x, y$ is the location, and temp is the temperature. When a measurement report is received in the moos variable *UCTD_SENSOR_REPORT*, it is parsed into a measurement by the function

```
CMeasurement CSimAnneal::parseMeas(string report)
```

returning an associated member of the measurement class. The process *pFrontEstimate* uses this function whenmever a new measurement is coming in, and is then added to the current measurement vector by the function

```
void CSimAnneal::addMeas(CMeasurement new_meas)
```

To add measurements made by another vehicle, you will have to create a message which transmits one or more measurements and pass them on to the parameter estimation process. It is up to you how to do that. One way is to pass the measurements as *UCTD_SENSOR_REPORT* to the collaborator vehicle, but you may instead want to make new message with a vector of measurements, and then code them each into the *UCTD_SENSOR_REPORT* format on the receiving vehicle.

## 3.3 Lab Assignment

### 3.3.1 Vehicle Name

First you configure your autonomy system such the the name of your vehicle is your first name. This will avoid conflicts in the dual-vehicle survey, and allow us to keep better track of your scores.

### 3.3.2 pHelmIvP Configuration

As a first step modify your behavior configuration from the previous Lab to incorporate the bounding box shown in Fig. 4, so we ensure you don't run one of our Kingfishers through the locks and out into the Atlantic, or run it into the dock or the Longfellow Bridge.

Secondly, we need you to configure the *BHV_AvoidCollision* behavior, so your two vehicles don't run into each other.

### 3.3.3 Configure Parameter Estimation

As a next step, configure the parameter estimator process *pFrontEstimate*. As we noted in the previous lab, the parameter *alpha* is very difficult to estimate with the small survey area we have, and the scores were highly dependent on the mismatch in the estimate of this parameter. For this Lab we will fix the value to *value = 500*. Make a few trial runs with the two vehicles to familiarize yourself with the performance.

### 3.3.4 Simultaneous Surveying

Next implement your adaptive sampling behavior, or if you have an improved version, use that. Then work with your partner to launch both of your vehicles with a common shoreside and run simultaneous missions to ensure that the bounding-box and collision avaoidance behaviors are proerly configured.

### 3.3.5 Measurement Data Sharing

Modify the *pFrontEstimate* process to allow the occational addition of measurement data and incorporate them in the annealing. Test this out with the template surveys.

### 3.3.6 Adaptive Collaborative Survey

As the final step, modify you behaviors and processes to execute a collaborative survey. You will be able to communicate with unlimited frequency and bandwidth when the two vehicles are within communication range.

# 2.S998 Marine Autonomy, Sensing and Communications

Spring 2012