pSearchGrid: Using a 2D Grid Model for Track History

June 2018

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

1 Overview **1**

2 Using pSearchGrid **2**
2.1 Basic Configuration of Grid Cells . . . 2
2.2 Cell Variables . . . 3
2.3 Serializing and De-serializing the Grid Structure . . . 3
2.4 Resetting the Grid . . . 4
2.5 Viewing Grids in pMarineViewer . . . 4
2.6 Examples . . . 4

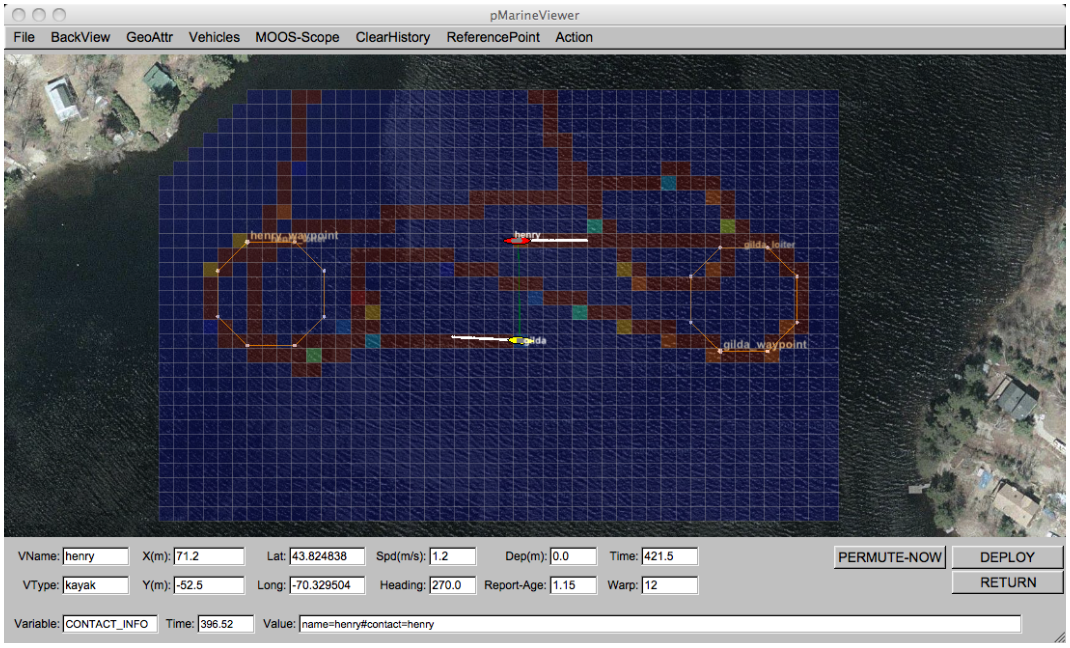
3 Configuration Parameters of pSearchGrid **4**

4 Publications and Subscriptions for pSearchGrid **5**
4.1 Variables Published by pSearchGrid . . . 5
4.2 Variables Subscribed for by pSearchGrid . . . 5
4.3 Command Line Usage of pSearchGrid . . . 6

1 Overview

The `pSearchGrid` application is a module for storing a history of vehicle positions in a 2D grid defined over a region of operation. This module may have utility as-is, to help guide a vehicle to complete or uniform coverage of a given area, but also exists as an example of how to use and visualize the `XYConvexGrid` data structure. This data structure may be used in similar modules to store a wide variety of user specified data for later use by other modules or simply for visualization. Here the structure is used in a MOOS application, but it could also be used within a behavior. The `pSearchGrid` module begins with a user-defined grid, defined in the MOOS configuration file. As a vehicle moves and generates node reports, `pSearchGrid` simply notes the vehicle's current position increments the cell containing vehicle's current position. After time, the grid shows a cumulative history of the most commonly traveled positions in the local operating area.

The `pSearchGrid` module is an optional part of the Charlie example mission discussed later in this section. When running and grid viewing is enabled in `pMarineViewer`, the viewer may show something similar to Figure 1.



VName: henry X(m): 71.2 Lat: 43.824838 Spd(m/s): 1.2 Dep(m): 0.0 Time: 421.5
VType: kayak Y(m): -52.5 Long: -70.329504 Heading: 270.0 Report-Age: 1.15 Warp: 12
Variable: CONTACT_INFO Time: 396.52 Value: name=henry#contact=henry

Figure 1: **An Example pSearchGrid Scenario:** A grid is configured around the operation area of the Berta example mission. Higher values in a given grid cell represents a longer noted time of any vehicle passing through that cell.

2 Using pSearchGrid

The present configuration of pSearchGrid will store values in its grid cells proportional to the time a vehicle is noted to be within a grid cell. One may use this application as written or regard this as a template for writing a new application that stores some other value with each grid cell, such as bathymetry data, water velocity data, or likelihood of there being an object of interest in the region of the cell for further investigation. The below discusses the general usage of managing the grid data within a MOOS application.

2.1 Basic Configuration of Grid Cells

Basic grid configuration consists of specifying (a) a convex polygon, and (b) the size of the grid squares. From this the construction of a grid proceeds by calculating a bounding rectangle containing the polygon, generating a set of squares covering the rectangle, and then removing the squares not intersecting the original polygon. The result is a non-rectilinear grid as shown in Figure 1. The basic configuration of grid cells is done with the `grid_config` parameter specifying the points and the cell size as the following example:

```
grid_config = pts={-50,-40:-10,0:180,0:180,-150:-50,-150}, cell_size=5
```

2.2 Cell Variables

The grid is primary used for associating information with each grid cell. In pSearchGrid, the grid is configured to store a single numerical value (a C++ double) with each cell. The grid structure may be configured in another application to store multiple numerical values with each cell. In pSearchGrid, the cell variables are declared upon startup in the MOOS configuration block, similar to:

```
grid_config = cell_vars=x:0y:100
grid_config = cell_min=x:0
grid_config = cell_min=x:100
```

This configuration associates two cell variables, x and y, which each cell. The cell variable x is initialized to 0 for each cell, and the cell variable y is initialized to 100. The first variable has a minimum and maximum constraint of 0 and 100, and the second variable is unconstrained. The above configuration could also have been made on one line, separating each with a comma.

2.3 Serializing and De-serializing the Grid Structure

The grid structure maintained by pSearchGrid is periodically published to the MOOSDB for consumption by other applications, or conceivably by a behavior with the IvP Helm. The structure may be serialized into a string by calling the `get_spec()` method on an instance of the `XYConvexGrid` class. Serializing a grid instance and posting it to the MOOSDB may look something like:

```
#include "XYConvexGrid.h"
....
XYConvexGrid mygrid;
....
string str = mygrid.get_spec();
m_Comms.Notify(VIEW_GRID, str);
```

Likewise a string representation of the grid may be de-serialized to a grid instance by calling a function provided in the same library where the grid is defined. De-serializing a string read from the MOOSDB into a local grid instance may look something like:

```
#include "XYFormatUtilsConvexGrid.h"
#include "XYConvexGrid.h"
....
string grid_string_spec;
....
XYConvexGrid mygrid = string2ConvexGrid(grid_string_spec);
```

The same function used to de-serialize a string is used by pSearchGrid to configure the initial grid. In other words, the components from the various `GRID` configuration parameters are concatenated into a single comma-separated string and passed to the de-serialization function, `string2ConvexGrid`, to form the initial instance used by pSearchGrid.

2.4 Resetting the Grid

The grid may be reset at any point in the operation when pSearchGrid receives mail on the variable PSG_GRID_RESET. If this variable's string value is the empty string, it will reset all cell variables for all cell elements to their given initial values. If the string value is non-empty, it will interpret this as an attempt to reset the values for a named cell variable. Thus PSG_GRID_RESET="x" will reset the x cell variable and no other cell variables. If "x" is not a cell variable, no action will be taken.

2.5 Viewing Grids in pMarineViewer

The pMarineViewer will display grids by subscribing for postings to the variable VIEW_GRID. As with other viewable objects such as polygons, points, etc., the viewer keeps a local cache of grid instances, one for each named grid, based on the grid label. For example if two successive grids are received with different labels, the viewer will store and render both of them. If they have the same label, the second will replace the first and the viewer will just render the one.

The rendering of grids may be toggled on/off by hitting the 'g' key, and may be made more transparent with the CTRL-g key, and less transparent with the ALT-g key. The color map used for the grid is taken from the typical MATLAB color map; red is the highest value, blue is the lowest.

2.6 Examples

Example usage of pSearchGrid may be found in both the Charlie and Berta example missions distributed with the moos-ivp tree. For example, in the Charlie example mission, edit charlie.moos and uncomment the line beginning with Run = pSearchGrid. Likewise for the Berta mission and the file meta_shoreside.moos.

3 Configuration Parameters of pSearchGrid

The following parameters are defined for pSearchGrid. A more detailed description is provided in other parts of this section. Parameters having default values indicate so.

Listing 3.1: Configuration Parameters for pSearchGrid.

grid_config:	A portion or all of a grid configuration description. See Listing 2.

An Example MOOS Configuration Block

An example pSearchGrid configuration block is given in Listing 2 below, and may also be seen from the command line invocation of:

```
$ pSearchGrid --example or -e
```

Listing 3.2: Example configuration of the pSearchGrid application.

```
===============================================================
pSearchGrid Example MOOS Configuration
===============================================================

ProcessConfig = pSearchGrid
{
  AppTick   = 4
  CommsTick = 4

  GRID_CONFIG = pts={-50,-40: -10,0: 180,0: 180,-150: -50,-150}
  GRID_CONFIG = cell_size=5
  GRID_CONFIG = cell_vars=x:0:y:0:z:0
  GRID_CONFIG = cell_min=x:0
  GRID_CONFIG = cell_max=x:10
  GRID_CONFIG = cell_min=y:0
  GRID_CONFIG = cell_max=y:1000
}
```

4 Publications and Subscriptions for pSearchGrid

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

```
$ pSearchGrid --interface or -i
```

4.1 Variables Published by pSearchGrid

The pSearchGrid application publishes the following variables:

- APPCAST: Contains an appcast report identical to the terminal output. Appcasts are posted only after an appcast request is received from an appcast viewing utility.
- VIEW_GRID: A full description of a grid format and contents.

A typical string may be:

```
VIEW_GRID = pts={-50,-40:-10,0:100,0:100,-100:50,-150:-50,-100},cell_size=5,
            cell_vars=x:0:y:0:z:0,cell_min=x:0,cell_max=x:50,cell=211:x:50,
            cell=212:x:50,cell=237:x:50,cell=238:x:50,label=psg
```

4.2 Variables Subscribed for by pSearchGrid

The pSearchGrid application subscribes to the following MOOS variables:

- APPCAST_REQ: A request to generate and post a new apppcast report, with reporting criteria, and expiration.
- NODE_REPORT: A node report for a given vehicle from pNodeReporter.
- NODE_REPORT_LOCAL: A node report for a given vehicle from pNodeReporter.
- PSG_GRID_RESET: A request to reset the grid to its original configuration.

4.3 Command Line Usage of pSearchGrid

The pSearchGrid application is typically launched with pAntler, along with a group of other modules. However, it may be launched separately from the command line. The command line options may be shown by typing "pSearchGrid --help":

Listing 4.3: Command line usage for the pSearchGrid tool.

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

Options:
  --alias=<ProcessName>
      Launch pSearchGrid with the given process
      name rather than pSearchGrid.
  --example, -e
      Display example MOOS configuration block
  --help, -h
      Display this help message.
  --version,-v
      Display the release version of pSearchGrid.
Note: If argv[2] is not of one of the above formats
      this will be interpreted as a run alias. This
      is to support pAntler launching conventions.
```