**Marine Plume Simulator**

Marine plume simulator is a ROS based simulator to simulate the growth of marine plumes. This document covers the basic hardware and software requirements and covers the basic necessary settings required to run the simulation.

**Citation:**

Please cite this paper if you use our work.

@article{marineplumesim,

title={ Experimentally Calibrated Gaussian Pollutant Dispersion Model to Achieve Fine Scale Structure in Marine Plumes},

author={Fahad, Muhammad and Guo, Yi and Bingham, Brian},

journal={IEEE/RSJ International Conference on Intelligent Robots and Systems},

volume={},

number={},

pages={},

year={Submitted, 2017}

}

**System requirements:**

CUDA enabled Graphics Processing Unit

Ubuntu v 14.04 or higher

ROS Indigo or later

CUDA

PyCUDA

**Installation Steps:**

The simulator doesn’t need to be installed. You can simply download and copy it to your catkin workspace ‘src’ folder. The simulator environment variable needs to be set. You can add this to the bashrc file on your Ubuntu installation by copying the below line to your bashrc file.

export PLUMESIM = $” path the simulator is copied”

**Properties File:**

First step to running your own simulation is to edit the properties file and set the simulation parameters according to your requirement. Each parameter in the properties file is explained below.

“*lx*”: The length in meters of the simulation environment along the x-axis.

“*ly*”: The length in meters of the simulation environment along the y-axis.

“*frameid*”: Sets the frame ID of the topics to be published.

*‘Src’*: This sets the ‘x’ and ‘y’ location of the source, ‘con’ concentration of the source in parts per billion (ppb). ‘mean\_time\_high’ if the mean value of the time in seconds for which this source remains high, ‘set\_time\_high’ standard deviation of the high time random variable. ‘mean\_low\_time’ is the mean value of the time in seconds for which the source remains zero, ‘set\_time\_low’ standard deviation of the low time random variable.

The line can be copied and pasted multiple times to simulation more than one source.

‘*vis\_conc*’: This sets the maximum visible concentration in ppb.

‘*cr\_num*’: Sets an upper bound on the maximum value of the Courant number.

‘*grid\_x*’: The grid resolution ∆x along x-axis in meters.

‘*grid\_y*’: The grid resolution ∆y along y-axis in meters.

‘*flow\_x*’: Sets the maximum value of the flow rate along x-axis in m/s, and whether the flow rate reverses direction.

‘*flow\_y*’: Sets the maximum value of the flow rate along y-axis in m/s, and whether the flow rate reverses direction.

‘*flow\_up\_rt*’: Flow update rate in seconds.

‘*flow\_sol\_t*’: The number of times the flow field is solved to reach a stable solution in the flow.py class.

‘*flow\_p\_init*’: The number of times the pressure field is solved to reach a stable solution in the flow.py class.

‘*sim\_duration*’: Duration of the simulation in seconds.

‘*k1\_x*’: Value of diffusion coefficient in m2/s along the x-axis.

‘*k1\_y*’: Value of diffusion coefficient in m2/s along the y-axis.

**Simulation Steps:**

After finishing editing the properties file, there are a few steps that the user must follow to run the simulation.

1. First step is to generate and save the flow field. This can be done by running the flow.py file in the package file. This will generate and save the flow field for the duration of the simulation. It can take a considerable amount of time depending on the set parameters.
2. Next, the simulator can either be run as a stand-alone simulator or can be run as part of some other simulation.
3. To run independently, first run the run\_simulator.sh script. Next, in another tab, run the plume\_node.py file. This should start the simulation and you would be able to see the simulator running.
4. To run as part of another simulation, assuming roscore is already running, the user only needs to run the plume\_node.py file.
5. The node in either case will start and publish “con\_cld” point cloud to visualize concentration, “flow\_cld” to visualize the flow field, and “src\_loc” to mark the location of the source. The node also publishes, “con\_val” topic for the concentration value at each point in the simulation environment, “flow\_x” for the flow value along x-axis, “flow\_y” for flow value along “y-axis” and “sim\_time” for the simulation time. All topics are published with frame id set in the properties file.

**Flow Field Import Procedure:**

The simulator can import flow fields generated by other generation softwares, such as OpenFOAM. The simulator expects seperate files for flow values along x and y-axis. The file naming convention is "uXXXX\_save.npy" for flow values along x-axis and "vXXXX\_save.npy" along y-axis. Here "XXXX" is the sequence number of the files and it should not be zero padded to the left for values less than 4 digits. The npy file contains a 2D numpy matrix, with ly/grid\_y rows and lx/grid\_x columns. All the data files must be stored inside the "data" folder in the simulator home folder. The flow field along x and y-axis at each point in the grid are stored in the .npy files.