Gas Dispersion Simulation
General Description

As shown in Figure 1, the gas dispersion simulator is composed of three ROS nodes namely “environment”, “dispersion_simulation” and “simulated_MOX”. These ROS nodes publish topics and visualization markers that are then used by conventional ROS nodes to display the results of a given simulation run. In the following sections, a brief overview of each of these ROS nodes is presented. The different roslaunch input variables are explained as well as the topics published by each of the nodes.

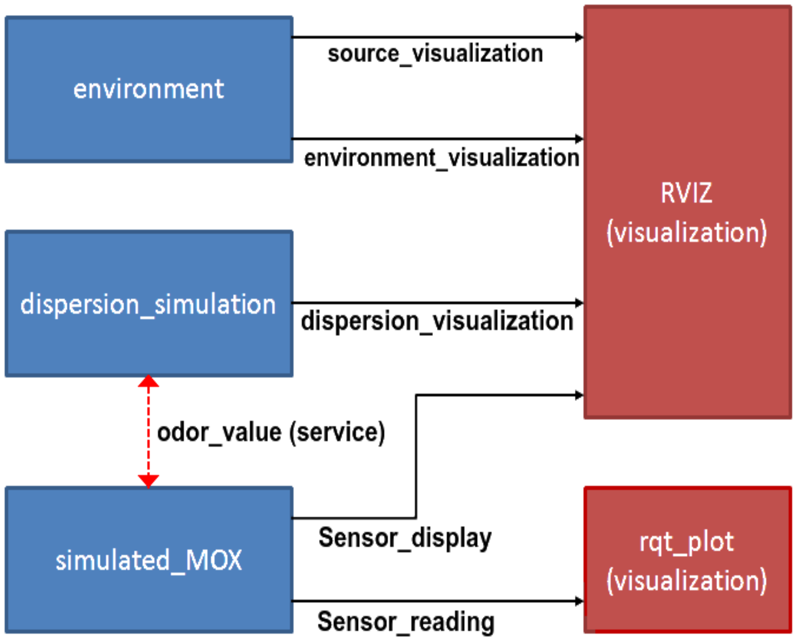

Figure 1: Block diagram of the gas dispersion simulator.

1.- Dispersion_simulation ROS Node

This node implements the gas dispersion simulator according to a set of input variables and wind flow files. The algorithms behind the implementation of the gas dispersion simulator can be consulted in [1] (see annexed file Pashami_etal_2010.pdf).

1.1.- Wind Flow Files

As a main input, **dispersion_simulation** uses a set of wind flow files generated by an external fluid dynamics simulator (e.g. OpenFoam[1]). The wind flow data generated by e.g. OpenFoam is given as a series of snapshots (periodically captured at given time intervals) that specify the wind vector components (U,V,W) at each location (x,y,z) in the simulated environment.

1 http://www.openfoam.com/

dispersion_simulation uses the following naming convention for the wind flow files is as follows:

experiment_name.0.csv_U
experiment_name.0.csv_V
experiment_name.0.csv_W
experiment_name.1.csv_U
experiment_name.1.csv_V
experiment_name.1.csv_W
…
experiment_name.n.csv_U
experiment_name.n.csv_V
experiment_name.n.csv_W

Each time snapshot (from 0 to n) is specified by three files, one for each of the wind vector components U,V,W. **dispersion_simulation** requires the following file structure:

(0,0,0)	(0,1,0)	...	(0,y,0)
(1,0,0)	(1,1,0)	...	(1,y,0)
...	...	...	...
(x,0,0)	(x,1,0)	...	(x,y,0)

;

(0,0,z)	(0,1,z)	...	(0,y,z)
(1,0,z)	(1,1,z)	...	(1,y,z)
...	...	...	...
(x,0,z)	(x,1,z)	...	(x,y,z)

In the above structure, the simulation environment is discretized in voxels of a given size. Each voxel contains the value of one wind vector component. Columns are separated by tabs ('\t'). Each column corresponds to the cell coordinate in the Y axis, rows corresponds to X and coordinates on the Z axis are separated by semicolons (;).

1.2.- Roslaunch Input Variables

- **source_position_x, source_position_y, source_position_z :** The coordinates (in m) of the emitting gas source.
- **gas_type:** Integer input variable that specifies the gaseous substance being released. The following are the set of substances that are already implemented in the code:
  (0) Ethanol, (1) Methane, (2) Hydrogen, (3) Propanol, (4) Chlorine, (5) Fluorine, (6) Acetone, (7) Neon, (8) Helium, (9) Hot air
- **delta_t:** This variable determines the refresh interval (in seconds) at which the gas dispersion model is re-computed.
- **wind_data:** The path were the set of wind files is located.
- **snapshots:** Number of snapshots to simulate.
- **filaments:** Number of filaments used by the simulation engine [1].
- **area_size_x, area_size_y, area_size_z:** The simulation area size (in number of cells).
- **area_cell_size:** The size of the cells in the simulation grid (in meters).
- **fixed_frame:** The reference frame (i.e. fixed frame) of the TF tree.

1.3.- Published Topics

- **dispersion_visualization:** A set of markers that allows to visualize the gas dispersion pattern on RVIZ.

1.4.- Services

- **odor_value** *x y z*: allows to query the gas dispersion model at an specific postion (x,y,z in meters). This service returns a point concentration value in PPM.
  Example: *rosservice call odor_value 3 3 1*(returns the concentration at location 3,3,1)

2.- Environment ROS Node

This ROS node publishes a set of markers to visualize the position of the gas source and the obstacles in the environment, as shown in Figure 2.

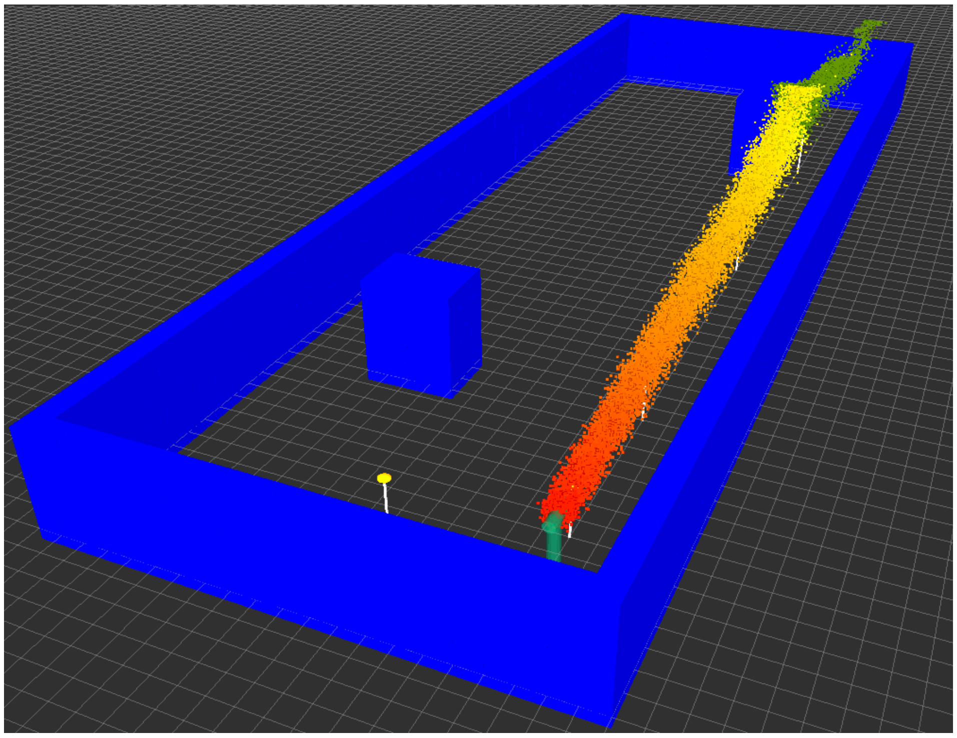

Figure 2: Environment visualization in RVIZ.

2.1.- Environment file structure

In order to visualize the configuration of the simulated environment, a environment file is needed. An environment file is a binary structure that specifies the occupancy of the voxels. A value equal to 1 denotes that the voxel is occupied by an obstacle, while 0 denotes that the voxel is unoccupied. The structure of an environment file is as follows:

(0,0,0)	(0,1,0)	...	(0,y,0)
(1,0,0)	(1,1,0)	...	(1,y,0)
...	...	...	...
(x,0,0)	(x,1,0)	...	(x,y,0)

;

(0,0,z)	(0,1,z)	...	(0,y,z)
(1,0,z)	(1,1,z)	...	(1,y,z)
...	...	...	...
(x,0,z)	(x,1,z)	...	(x,y,z)

Similarly to the wind flow files, columns are separated by tabs ('\t'). Each column corresponds to the cell coordinate in the Y axis, rows corresponds to X and coordinates on the Z axis are separated by semicolons (;).

2.2.- Roslaunch Input Variables

- **source_position_x, source_position_y, source_position_z :** The coordinates (in m) of the emitting gas source.
- **environment_data:** The path of the environment file.
- **fixed_frame:** The reference frame (i.e. fixed frame) of the TF tree.
- **area_size_x, area_size_y, area_size_z:** The simulation area size (in number of cells).
- **area_cell_size:** The size of the cells in the simulation grid (in meters).

2.3.- Published Topics

- **source_visualization:** A set of markers that denote the position of the emitting gas source
- **environment_visualization:** A set of markers that denote the position of obstacles/walls in the environment.

3.- Simulated_MOX ROS Node

This node simulates different models of metal oxide MOX sensors. **simulated_MOX** sends a service request to **dispersion_simulation** to query for the concentration level at the sensor's position and publishes the sensor's resistance according to the concentration, gas type and sensor model.

3.1.- Roslaunch Input Variables

- **gas_type:** Integer input variable that specifies the gaseous substance being released. The following are the set of substances that are already implemented in the code:
  (0) Ethanol, (1) Methane, (2) Hydrogen, (3) Propanol, (4) Chlorine, (5) Fluorine, (6) Acetone, (7) Neon, (8) Helium, (9) Hot air.
- **sensor_model:** The model of the MOX sensor to simulate. (0) TGS-2620, (1) TGS-2600, (2) TGS-2611.
- **fixed_frame:** The reference frame (i.e. fixed frame) of the TF tree.
- **sensor_frame:** The TF frame of the simulated sensor.

3.2.- Published Topics

- **Sensor_reading:** The sensor resistance in Ohms.
- **Sensor_display:** A set of markers that denote the position of the sensor in the simulation area.

3.3.- Defining Multiple Sensors

Multiple instances of simulated_MOX can be launched. This is useful when e.g. a sensor array or multiple robots are simulated. In order to launch multiple instances, simulated_MOX should be launched in different name spaces as follows:

```
<group ns="Mox01">
<node pkg="simulated_MOX" type="simulated_MOX" name="fake_MOX" output="screen">
        <param name="/sensor_model" value="2" />
        <param name="/sensor_frame" value="sensor01_frame" />
        <param name="/fixed_frame" value="$(arg FixedFrame)"/>
        <param name="/gas_type" value="$(arg GasType)"/>
</node>
</group>
```

```
<group ns="Mox02">
<node pkg="simulated_MOX" type="simulated_MOX" name="fake_MOX" output="screen">
		<param name="/sensor_model" value="2" />
		<param name="/sensor_frame" value="sensor02_frame" />
		<param name="/fixed_frame" value="$(arg FixedFrame)"/>
		<param name="/gas_type" value="$(arg GasType)"/>
</node>
```

3.- Simulated_TDLAS ROS Node

This node simulates a remote gas sensor. **simulated_tdlas** sends a service request to **dispersion_simulation** to query for the concentration level along one laser and publishes the sensor's resistance according to the concentration and gas type .

3.1.- Roslaunch Input Variables

- **gas_type:** Integer input variable that specifies the gaseous substance being released. The following are the set of substances that are already implemented in the code:
  (0) Ethanol, (1) Methane, (2) Hydrogen, (3) Propanol, (4) Chlorine, (5) Fluorine, (6) Acetone, (7) Neon, (8) Helium, (9) Hot air.
- **sensor_model:** The model of the MOX sensor to simulate. (0) TGS-2620, (1) TGS-2600, (2) TGS-2611.
- **fixed_frame:** The reference frame (i.e. fixed frame) of the TF tree.
- **sensor_frame:** The TF frame of the simulated sensor.

3.2.- Published Topics

- **Sensor_reading:** The sensor resistance in Ohms.
- **Sensor_display:** A set of markers that denote the position of the sensor in the simulation area.
- **visualization_marker**: for visualization purpose (laser...)

References

[1] *Sepideh Pashami, Sahar Asadi, and Achim J. Lilienthal*, Integration of OpenFOAM Flow Simulation and Filament-Based Gas Propagation Models for Gas Dispersion Simulation. *Proceedings of the Open Source CFD International Conference, 2010.*