Integration of the Virtual Gas Release Dataset into the Data-Driven Plume Model (DDPM)

Ayoub Ouederni

École Polytechnique Fédérale de Lausanne (EPFL)

Supervisor: Wanting Jin

Supported by the Laidlaw Foundation

September 2025

Abstract

Gas source localization and plume modeling are central challenges in mobile robotics and environmental intelligence. Traditional methods, such as Gaussian plume models or Computational Fluid Dynamics (CFD), either lack accuracy or are too computationally demanding for real-time use on resource-limited robots. The Data-Driven Plume Model (DDPM) addresses this limitation by using deep learning surrogates trained on simulated data.

The contribution of this work is the integration of the Virtual Gas Release (VGR) dataset into the DDPM pipeline. Unlike the original tunnel-based dataset, VGR provides more realistic multi-room environments with complex airflow and obstacles. This report details the preprocessing steps, dataset adaptation, and automation required to make VGR compatible with the existing DDPM architecture.

The resulting framework enables direct training on VGR data, extending the model's applicability beyond simplified environments. Future directions include combining tunnel and VGR datasets to test robustness, extending the approach to 3D reconstructions, and evaluating performance in real robotic deployments.

Acknowledgments

This research was supported by the Laidlaw Foundation and carried out at the École Polytechnique Fédérale de Lausanne (EPFL). I would like to express my sincere gratitude to my supervisor, Wanting Jin, for her continuous guidance, encouragement, and patience throughout this project. I am also thankful to the members of the DISAL laboratory for their valuable feedback and technical assistance.

Contents

0.1 Introduction

Gas source localization and plume modeling are central challenges in mobile robotics and environmental intelligence. Applications range from industrial safety and environmental monitoring to emergency response in case of hazardous leaks. Traditional approaches based on analytical dispersion models face two major limitations: first, the inherent complexity of turbulent flows makes them difficult to represent accurately; second, high-fidelity CFD (Computational Fluid Dynamics) simulations are computationally expensive, preventing their use in real-time scenarios on robots with limited resources. To address this problem, the Data-Driven Plume Model (DDPM) was developed as an alternative. The central idea is to replace analytical equations or CFD solvers with a deep learning surrogate trained directly on simulated data. This model can predict gas concentration maps quickly, thus enabling real-time use in source localization algorithms. The original DDPM pipeline combined several tools: OpenFOAM to generate wind fields, Webots and GADEN to simulate gas propagation, MATLAB to assemble and preprocess datasets, and PyTorch to train a convolutional neural network based on a U-Net architecture. However, this pipeline was restricted to simple tunnel-like environments with randomly placed obstacles. Such limited scenarios significantly constrained the model's ability to generalize to more realistic situations. The goal of the present work is to broaden this pipeline by integrating a more diverse and realistic dataset: the VGR (Virtual Gas Release) dataset. Developed within the MAPIR project, VGR offers multi-room house-like environments with complex geometries and varied obstacles. Incorporating VGR into the DDPM pipeline is a key step toward overcoming the limitations of the original setup and testing the robustness of the model in scenarios that are closer to reality.

0.2 Background & Related Work

Modeling gas dispersion has been a longstanding research topic at the intersection of fluid dynamics, robotics, and environmental sciences. Traditional approaches rely on analytical plume models, such as Gaussian approximations, or on high-fidelity CFD simulations of the Navier–Stokes equations. While analytical models offer simplicity and fast computation, they are often too coarse to capture turbulent and obstacle-driven effects. On the other hand, CFD provides accurate predictions but at a computational cost that makes it impractical for real-time robotic applications. Recent advances in machine learning have opened new avenues for data-driven surrogates of physical processes. A notable example is DeepCFD, which demonstrated that convolutional neural networks (CNNs) based on U-Net architectures can approximate steady-state flow fields orders of magnitude faster than CFD solvers, with minimal loss in accuracy. This approach inspired the development of the Data-Driven Plume Model (DDPM), designed specifically for gas dispersion prediction. The original DDPM pipeline combined physics-based simulators and data-driven learning. OpenFOAM was used to simulate wind profiles in tunnel-like environments with obstacles. These wind maps were then processed in MATLAB to then generate plume dispersions using static sensor networks. The resulting datasets consisted of pairs of environment definitions, obstacle maps, and distance-to-source features as inputs, with gas concentration maps as outputs. A U-Net model was then trained on this data to serve as a surrogate for plume predictions. Finally, the trained model was integrated into a Source Term Estimation (STE) framework, enabling robots to localize unknown gas sources by querying the surrogate instead of running costly CFD simulations. Despite its promising results, the first version of DDPM was limited to synthetic tunnel environments. These environments were relatively simple, with regular geometries and uniform boundary conditions. To address this limitation, the Virtual Gas Release (VGR) dataset was proposed by the MAPIR group. VGR provides a collection of multi-room, house-like environments generated through OpenFOAM and GADEN. It offers pre-computed plume simulations in environments with complex layouts and diverse obstacle configurations. Unlike the tunnel dataset, VGR aims to mimic realistic indoor conditions, making it particularly valuable for robotics research.

0.3 Original DDPM Pipeline

The original DDPM pipeline was designed to generate training data, train a deep learning model, and deploy it as a surrogate for gas dispersion within the Source Term Estimation (STE) framework. It consisted of several sequential stages, each relying on specialized software tools.

0.3.1 Environment and Wind Field Generation

The first stage used OpenFOAM to simulate airflow within synthetic tunnel-like environments. These environments were generated either randomly, with obstacles placed inside rectangular tunnels, or manually, for controlled test cases. OpenFOAM simulations produced detailed wind fields, capturing the velocity components across the domain.

0.3.2 Preprocessing for Plume Simulation

Since Webots requires simplified inputs, the OpenFOAM wind fields were post-processed using MATLAB scripts. These scripts exported the velocity maps into a format that could be loaded into Webots, ensuring consistency across the training set. A universal 64×64 grid was defined, with coordinates aligned across all simulations, so that the resulting datasets would be compatible for machine learning training.

0.3.3 Gas Dispersion Simulation in Webots

The next step simulated gas dispersion directly in Webots, where a static network of virtual gas sensors sampled plume concentrations. For each environment, multiple gas source positions were tested, and the simulator collected the average gas concentration across the sensor grid. The outputs were stored as CSV files, each corresponding to a plume realization for a given map and source position.

0.3.4 Dataset Assembly

The resulting plume maps were combined with environment descriptors to form the final dataset. For each source position, the inputs included:

- Flow definition (`flow_def`): describing obstacle placement and free space.
- Signed Distance Function (`SDF`): representing distances to obstacles.

- Inverse distance to the source: capturing spatial proximity to the release point.

The target output consisted of the average gas concentration map, defined on the same 64×64 grid. This was encoded into MATLAB `.mat` files, which could then be read by the training scripts.

0.3.5 Model Training

The dataset was used to train a U-Net–based convolutional neural network, inspired by the *DeepCFD* approach. The model took three input channels (`flow_def`, `SDF`, inverse distance to source) and produced one output channel (average gas concentration). Training was performed in `PyTorch`, with 90% of the dataset used for training and 10% for testing. After convergence, the model was exported as a `.pt` file and could be deployed either in Python or in C++ through `libtorch`.

0.3.6 Deployment in STE

The trained model was integrated into the Source Term Estimation (STE) framework. Instead of running expensive CFD simulations at runtime, the STE algorithm queried the surrogate model to obtain plume concentration estimates at arbitrary points. This reduced computational cost drastically, enabling real-time feasibility on robots.

0.3.7 Limitations

While effective, this pipeline was restricted to synthetic tunnel environments. The geometries were simple, airflow patterns relatively regular, and source–obstacle interactions limited. As a result, the trained model risked overfitting to simplified patterns and lacked robustness in more realistic conditions. This limitation motivated the integration of the VGR dataset.

0.4 Integration of the VGR Dataset

The main contribution of this project was the integration of the Virtual Gas Release (VGR) dataset into the DDPM pipeline. Unlike the original DDPM dataset, which relied on synthetic tunnels, VGR provides precomputed gas dispersion simulations in multi-room house-like environments with complex layouts and realistic airflow patterns. While this dataset offers greater realism and diversity, it was originally stored in formats incompatible with the DDPM pipeline, requiring extensive adaptation and preprocessing.

0.4.1 Dataset Characteristics

The VGR dataset, developed by the MAPIR group, combines `OpenFOAM` wind simulations with the `GADEN` gas dispersion simulator. The results are stored in binary format, later convertible into CSV files using the `writeConcentrations` utility. Each simulation corresponds to a specific house environment, airflow configuration, and gas source location. The dataset is organized into directories, where each experiment contains multiple timesteps of plume evolution. Each timestep records concentration values and wind vectors over a 3D voxel grid with a spatial resolution of 10 cm.

This structure contrasts with the original DDPM dataset, which directly produced 2D plume maps on a fixed 64×64 grid. Therefore, integrating VGR required extracting appropriate 2D slices and reformatting them to match the DDPM pipeline.

0.4.2 Conversion from Binary to CSV

The first challenge was that the VGR data was provided in a proprietary binary format. Using the `writeConcentrations` program included in the dataset utilities, the binary files were converted into CSV files. Each CSV contains columns representing cell indices (x, y, z), gas concentration in parts per million (ppm), and wind velocity components (U, V, W).

During this process, a large number of timesteps were generated for each simulation, leading to significant storage requirements. To reduce redundancy, only one timestep out of ten was retained. This subsampling preserved the temporal evolution while keeping the dataset size manageable.

0.4.3 Extraction of 2D Slices

Since the DDPM model operates on 2D concentration maps, it was necessary to extract a horizontal slice from the 3D voxel grid. After analysis, the slice corresponding to $z = 0.3$ m

was selected, as it provides a representative cross-section consistent with the height used in the original DDPM setup. This decision ensured compatibility between the two datasets.

0.4.4 Interpolation onto the Universal Grid

The extracted slices from VGR did not align directly with the universal 64×64 grid used in the DDPM pipeline. To resolve this, the concentration values were interpolated using a scattered data interpolation method. The interpolation mapped the irregular cell coordinates from VGR onto the fixed grid defined in the original pipeline (x from 2.75 m to 12.74 m, y from 0.01 m to 3.99 m). This ensured that all inputs and outputs would share a consistent format.

0.4.5 Feature Construction

For each source position, the dataset was restructured to match the three input channels expected by the DDPM model:

- Flow definition (`flow_def`): derived from the occupancy grid of the house, indicating obstacles versus free space.
- Signed Distance Function (`SDF`): computed from the occupancy grid, representing distances to obstacles.
- Inverse distance to source: calculated from the known source coordinates embedded in the simulation folder names.

The output was defined as the average gas concentration map on the 64×64 grid, consistent with the targets used in the original dataset.

0.4.6 Automation and Implementation

The entire preprocessing pipeline was automated through a dedicated Python script. The script:

- Listed and read all CSV files generated from VGR simulations.
- Extracted the required 2D slice at $z = 0.3$ m.
- Interpolated concentration values onto the universal 64×64 grid.
- Computed `flow_def` and `SDF` from occupancy grids.

- Calculated the inverse distance-to-source map.
- Packaged all inputs (`py_x`) and outputs (`py_y`) into `.mat` files, exactly mirroring the structure produced by the original DDPM pipeline.

The resulting files — `my_x_for_torch_VGR.mat` and `my_y_for_torch_VGR.mat` — can be used directly for training the U-Net model without modifying the existing `PyTorch` code.

0.5 Contributions

The core of this project was the adaptation and extension of the DDPM pipeline to incorporate the VGR dataset. This process required solving several practical challenges, implementing multiple conversion and preprocessing steps, and ensuring strict compatibility with the existing framework. Below is a detailed account of the work carried out.

0.5.1 Handling VGR Data Formats

The VGR dataset was originally stored in a binary format, which could not be directly processed by the DDPM pipeline. The first step was to use the provided `writeConcentrations` utility to convert the binary outputs into CSV files. Each CSV contained the following information for each cell of the simulation grid:

- x, y, z indices (10 cm resolution),
- Gas concentration in ppm,
- Wind velocity components (U, V, W).

Since each simulation consisted of thousands of timesteps, the conversion process produced very large amounts of data. To manage this, I implemented a filtering step that retained one file out of ten, thereby reducing the dataset size while preserving its variability.

0.5.2 Directory Management and Automation

The dataset structure included numerous simulation folders for different environments, airflow conditions, and source positions. Managing this volume of data manually would have been infeasible. Therefore, I created Bash scripts to automate the processing workflow. These scripts performed the following tasks:

- Iterated through simulation folders and launched the binary-to-CSV conversion.
- Moved and renamed output files systematically to maintain consistency.
- Filtered timesteps to retain only the required subset.
- Removed intermediate files when necessary to save storage space.

This automation was crucial to make the dataset manageable and reproducible.

0.5.3 Extraction of 2D Slices

Unlike the DDPM dataset, which was natively 2D, the VGR dataset is three-dimensional. The model, however, requires 2D concentration maps on a fixed 64×64 grid. To address this, I selected a horizontal slice at $z = 0.3$ m, corresponding to a representative height within the simulation environments.

For each CSV file, I extracted the subset of rows where $z = 0.3$ m. This produced a 2D map of concentration values and corresponding coordinates.

0.5.4 Interpolation onto the Universal Grid

The coordinates provided by VGR did not align with the universal grid defined in the original DDPM pipeline (x from 2.75 m to 12.74 m, y from 0.01 m to 3.99 m, 64×64 resolution). To ensure strict compatibility, I implemented an interpolation step:

- The extracted 2D concentration values were interpolated using scattered data interpolation.
- The output was resampled exactly on the DDPM universal grid.
- Missing values resulting from interpolation gaps were replaced by zeros.

This step guaranteed that the final dataset matched the same dimensions and indexing scheme as the original one.

0.5.5 Occupancy Grid Processing

Each VGR house environment includes an `OccupancyGrid3D.csv` file that encodes the geometry of the environment (walls, obstacles, free space). These files contained metadata lines and irregular formatting, making them difficult to parse directly. To solve this:

- I wrote a robust parser in Python capable of ignoring metadata and handling irregular rows.
- From the occupancy grid, I constructed the flow definition (`flow_def`) used as an input channel.
- I also computed the Signed Distance Function (`SDF`), representing the distance from each free cell to the nearest obstacle, using morphological operations.

These two maps, together with the inverse distance-to-source map, completed the three input features expected by the model.

0.5.6 Source Position Extraction

The coordinates of the gas source were encoded in the simulation folder names (e.g., `sourcePosition_-0.90_3.30_-0.50`). I implemented a regular expression parser to automatically extract the source x and y positions from these strings. From these values, I computed the inverse distance-to-source map for each environment, aligned with the universal grid.

0.5.7 Assembly into Training Data

After generating the three input channels (`flow_def`, `SDF`, inverse distance to source) and the output channel (concentration map), I assembled them into arrays:

- `py_x`: shape $[N, 3, 64, 64]$,
- `py_y`: shape $[N, 1, 64, 64]$,

where N is the number of processed plume maps. To ensure randomness and avoid ordering biases, the dataset was shuffled before saving.

0.5.8 Export to `.mat` Format

Finally, the datasets were exported into MATLAB `.mat` files to replicate exactly the structure of the original DDPM dataset:

- `my_x_for_torch_VGR.mat` containing all inputs.
- `my_y_for_torch_VGR.mat` containing all outputs.

These files are fully compatible with the existing `PyTorch` training code (`DeepCFD.py`), meaning the model can be retrained or fine-tuned on VGR data without any code modifications.

0.5.9 Implementation Challenges and Solutions

Throughout the project, several technical issues had to be solved:

- **Inconsistent file formats:** Some occupancy grids contained headers and irregular rows, requiring custom parsing rather than standard CSV loading.
- **Large data volume:** Processing thousands of timesteps quickly exceeded storage limits. Subsampling (1/10 files) solved this issue while maintaining dataset diversity.

- **Grid mismatch:** VGR's 3D voxel grids did not align with DDPM's 2D universal grid. Interpolation routines were implemented to enforce strict compatibility.
- **Automation:** Manual handling of simulations was impractical; Bash and Python scripts ensured reproducibility and efficiency.

0.6 Conclusion and Perspectives

This project advances the Data-Driven Plume Model (DDPM), designed to replace analytical gas dispersion models with data-driven surrogates. The original pipeline, though effective, was restricted to simplified tunnel environments and lacked generalization to realistic scenarios.

The main contribution was integrating the Virtual Gas Release (VGR) dataset into the pipeline. This involved:

- Converting `GADEN` binary outputs to CSV,
- Automating filtering and renaming with Bash scripts,
- Developing a Python preprocessing tool,
- Extracting 2D slices at $z = 0.3$ m,
- Interpolating onto the universal 64×64 grid,
- Exporting `.mat` files compatible with the `PyTorch` framework.

These steps transformed an initially incompatible dataset into one fully aligned with the existing architecture.

With this extension complete, the next step is to compare training on tunnels, on VGR, and on both combined. This will test whether added diversity improves generalization or creates conflicts between datasets. Such conflicts are possible: models trained on complex multi-room layouts may underperform in simple tunnels, while tunnel-trained models may fail in realistic conditions.

For robotic deployment, balance is critical. Since DDPM acts as a surrogate within the Source Term Estimation (STE) framework, dataset bias could cause inconsistent behavior. Conversely, a balanced integration could greatly enhance robustness.

Future work includes:

- Extending the pipeline to multi-slice or 3D reconstructions,
- Injecting realistic noise to reduce the simulation–reality gap,
- Testing fine-tuning strategies,
- Evaluating performance not only with reconstruction metrics but also in end-to-end STE experiments (simulations and wind-tunnel tests).