GadenTools: A Toolkit for Testing and Simulating Robotic Olfaction Tasks With Jupyter Notebook Support

Pepe Ojeda[1], Jose-Raul Ruiz-Sarmiento[1], Javier Monroy[1] and Javier Gonzalez-Jimenez[1]

Machine Perception and Intelligent Robotics Group (MAPIR-UMA). Malaga Institute for Mechatronics Engineering & Cyber-Physical Systems (IMECH.UMA). University of Malaga. SPAIN

Abstract. This work presents GadenTools, a toolkit designed to ease the development and integration of mobile robotic olfaction applications by enabling a convenient and user-friendly access to Gaden's realistic gas dispersion simulations. It is based on an easy-to-use Python API, and includes an extensive tutorial developed with Jupyter Notebook and Google Colab technologies. A detailed set of examples illustrates aspects ranging from basic access to sensory data or the generation of ground-truth images, to the more advanced implementation of plume tracking algorithms, all in an online web-editor with no installation requirements. All the resources, including the source code, are made available in an online open repository.

Keywords: Robotic Olfaction, Gas Dispersion Simulation, Python, Jupyter Notebook, Google Colaborative

1 Introduction

Mobile robot olfaction (*MRO*) is the field concerned with the integration and application of the sense of smell into mobile robots. It is a widely multidisciplinary research area, involving problems such as chemical sensing and classification [4,15], dispersion modeling [5,7], optimal sampling [8] or gas source localization [10,17], among others.

There are many potential applications for autonomous, mobile robotic agents with the ability to sense the presence of gases in an environment, *e.g.* locating leaks in pipes, detecting dangerous substances, rescue missions, air quality monitoring, *etc.* As a result of these interesting prospects, the field of robotic olfaction has steadily been gaining attention over the years, and is expected to continue to do so as the available technology improves.

A key hurdle, preventing much research from reaching the degree of maturity necessary for real-world deployment, is the lack of proper datasets and tools in the field that can be used to conduct ground truth evaluations. Works dealing with these concepts are crucial, and highly popular in other fields such as

computer vision [11]. In MRO, though, performing experiments in real environments is frequently not possible, and there is a serious technological limitation to obtaining ground truth information of the spatial distribution of gases in the environment [22,16].

These reasons, alongside the fact that the robotic olfaction research community is still small, make it so that not many resources (datasets, visualization tools, *etc.*) are available to facilitate the research progress. We believe that creating these assets for robotic olfaction, along with the tools to make them easily accessible to our peers, is a meaningful contribution to the field that can boost advances, and specially will help newcomers get started on a subject that has been, up to now, somewhat obscure.

In this work we present GadenTools, a toolkit to handle data from gas dispersion simulations that aims to push in that direction. It is shipped as an easy-to-use Python [6] module, and includes an extensive tutorial[1] developed with the Jupyter Notebook [9] and Google Colab technologies. This notebook-shaped tutorial was designed to illustrate both the basic interaction with the GadenTools module's API and some more advanced use cases of how to exploit its capabilities to perform powerful visualizations, data analysis, and methods' validations. For doing so, it takes advantage of the Jupyter notebooks ability to combine the expressiveness of traditional explanations from textbooks (texts, equations, figures, etc.), with the interaction capabilities of software applications and executable code with visible outputs [9,21]. As a result, GadenTools allows for a fast prototyping and development of contents related to MRO. All the resources, including the source code, are made available in an online open repository[2].

More specifically, GadenTools works with data from Gaden simulations. Gaden [16] is a gas dispersion simulator specifically designed for mobile robotics that relies on Computational Fluid Dynamics (CFD) data. Although it has been widely used in *MRO* research already [20,3,17], we believe the somewhat steep learning curve required to use it, alongside the requirement to install additional software (*i.e.* the Robotics Operating System, *ROS*), might be making researchers who would benefit from its use to decide against it. In this way, GadenTools offers a clean and accessible entry to Gaden simulations, while removing the need for other software dependencies.

2 On the Importance of Gas Dispersion Simulators and Datasets

The phenomenon of gas dispersion is very complex, and developing techniques to understand it and infer the state of the environment from sparse measurements is a challenging task that requires extensive experimentation. This experimentation process is also non-trivial, as setting up repeatable, consistent experiments

[1] https://colab.research.google.com/drive/1Xj7rrsmeDa_dS3Ru_UIhhzlaifGH6GS4?usp=sharing

[2] https://github.com/MAPIRlab/GadenTools

requires a way to control the airflow, temperature, source release rate, and many other factors. As a result, real-world experiments are mostly limited to simple, wind-tunnel-like environments [23,14].

Simulation is thus an essential tool for researchers, allowing them to perform perfectly repeatable experiments in environments of certain complexity. While simulations, no matter their precision, do not avoid real-world validation of the robotic techniques, they are a key tool in the early stages of development and testing.

However, because CFD simulations with state-of-the-art numerical solvers are difficult to set up and offer no integration with robotics software, many researchers end up opting for custom-made simulation tools that are more appropriate for robotic olfaction [2,13]. Since these tools are not commercial software meant to be used by the non-expert public, even when they are available online they are often difficult for new users to interact with and lack proper documentation. This ends up leading to a situation where each research group uses their own internal simulation tool, which is incompatible with those used by everyone else and hampers the fair comparison between developments. And so, getting started in the field of robotic olfaction requires learning how these complex tools work, or implementing one's own.

Monroy *et al.* attempted to solve this integration issue by implementing Gaden, a gas dispersion simulator, under the umbrella of the well-known Robot Operating System (ROS) [19]. This way, Gaden was developed as a ROS package [16], thus allowing any other ROS process to communicate with it through topics and messages. However, while this undoubtedly makes access to the simulations more convenient for those researchers who use ROS to develop their own software, it also makes the data inaccessible to those who do not. The tools presented in this work intend to help bridge that gap, making the simulation data easily accessible in multiple formats by directly querying an API through standalone Python code.

Recently, the authors of this work also released the VGR Dataset [18], a large repository of already computed simulations in complex environments, made public online[3]. Figure 1 shows some snapshots of the CFD-generated airflows for two different indoor environments (included in the dataset) that can be accessed with GadenTools. This dataset was intended to eliminate the necessity for researchers to design and create their own simulations, which is in itself a time-consuming process that requires a degree of familiarity with the simulation tools and additional knowledge of other fields (*e.g.* 3D modelling). Moreover, the data are provided in different formats, including human-readable CSV files, enabling their usage outside Gaden at the expense of developing a toolkit for managing them.

GadenTools aims to palliate this issue by reducing the barrier of entry to the field of robotic olfaction, allowing researchers to focus on developing their algorithms and techniques, rather than on implementing simulation tools and designing testing environments.

[3] https://mapir.isa.uma.es/mapirwebsite/?p=1708

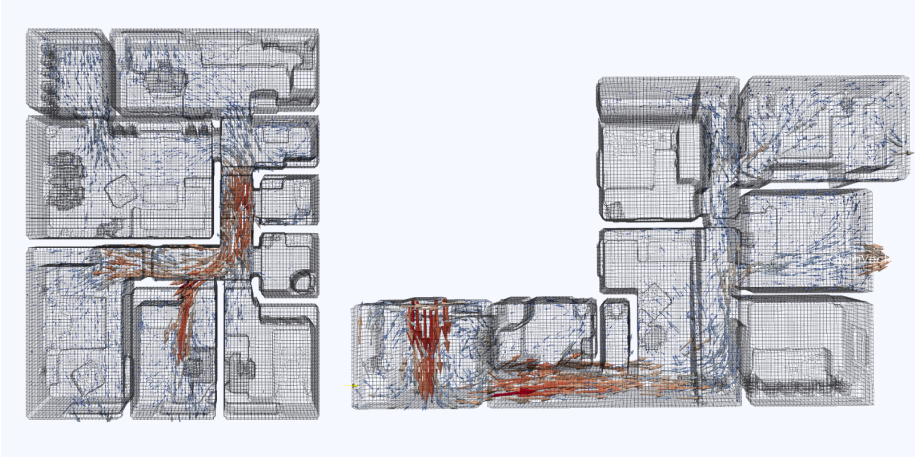

Fig. 1: Examples of airflows generated with CFD tools in two different environments. Having multiple inlets and outlets, as well as many obstacles, tends to produce flows that are more complex than would be possible in a wind tunnel.

3 Technologies Employed

In this section, we will briefly discuss the choice of technologies used for the implementation and deployment of the GadenTools Python's module and interactive tutorial.

3.1 Python

The Python programming language [6] was chosen for this project due to multiple reasons, one of the most prominent of which is that the main aim of the project is being easily accessible to everyone, regardless of their previous experience on the field or their technical skill. Python is a very popular language with an easy-to-learn syntax and a great deal of existing learning resources and useful libraries that help reduce the barrier of entry.

Another important reason why Python was a natural choice is that it is fully compatible with ROS. While GadenTools does not need to directly interact with the existing ROS implementation of Gaden (and, indeed, in its current state it is meant to be used in its stead), keeping open the possibility for their interaction makes it so that future updates to the GadenTools project may add extra functionality and create a more user-friendly way to interact with Gaden, while still leveraging the speed and stability of the existing C++ Gaden ROS nodes.

A drawback of Python is that it is not appropriate for highly performant code. While most of the operations carried out in GadenTools do not require great speed, there are some types of queries (see the visualization examples in section 4.2) that may be inconveniently slow when reading simulations of big size.

For this reason, the implementation of GadenTools uses Cython [1], an extension of the Python language that is statically compiled and thus allows for greatly optimized execution. Cython modules can be used from standard Python code seamlessly, and users of the module do not need to modify their own program in any way to account for this.

3.2 Jupyter Notebooks

The Jupyter Notebook technology [9] has become the de facto choice in the scientific community for producing reproducible computational workflows. It is of special relevance in this work since it permits us to build up an interactive tutorial describing GadenTools and its usage. This type of resources are also called *computational stories* or *live documents*. This minimizes the tool's learning curve and illustrates how to perform different robotic olfaction task with it. Jupyter Notebook consists of three main components:

- **Notebook**. Text document (JSON-based) used to define the Jupyter notebook itself. It consists of a list of text or code cells. Text cells contain markdown-formatted text, which enables the inclusion of math formulas, tables, lists, rich media (images, videos, audios), etc. If needed, HTML can also be used. In its turn, code cells contain source code in the kernel language that can be executed, being their output (text, plots, etc.) printed below them.
- **Web application**. Web-based interactive development environment to create, edit and run notebooks. It allows for the hosting of the notebooks in a server in order to work with them, given that the application is also installed. Local installations are also possible using package managers such as pip or Anaconda. The classic notebook interface evolved into *JupyterLab*, which fixed different usability issues and expanded its scope.
- **Kernel**. Backend application in charge of interacting with the Jupyter application by executing code cells and making readily available their output. Although there exist kernels for more than 50 programming languages[4], here we resort to the *IPython* kernel, which executes Python code.

Instead of JupyterLab, we have relied on Google Colaboratory[5] (Colab for short) for implementing the tutorial. This application has some advantages for the design of an illustrative notebook: it provides free cloud resources so the tutorial can be executed in any browser with internet connection, it enables IDE-like features like autocompletion, function declaration inspection from any point of the notebook, a variables-watcher, etc., and other features like showing a table of contents to the user. Using Google Colab, the user can launch the tutorial in a non-writable fashion and play with it without making any local installation. Morever, they could make a local copy (just of the tutorial) to modify it according to their needs.

[4] https://github.com/jupyter/jupyter/wiki/Jupyter-kernels
[5] https://colab.research.google.com/

4 GadenTools

GadenTools can be seen as a set of tools containing two main components: a Python module for managing datasets, and a Jupyter Notebook-based tutorial for illustrating how to operate with it, which also serves as a starting point for further developments. Next sections describe them in further detail.

4.1 The Python Module

In order to better understand the module's functionalities, it is convenient to first dive into the primitives it works with: data from Gaden simulations. The result of one of these simulations comprises a series of files in an internal binary format, which separately hold information about: i) the airflow vectors through the environment, ii) the gas distribution, as well as iii) a CSV file that describes the geometry of the 3D environment in which the simulation takes place. These binary representations are intended to speed up both reading and writing the simulation results, and to reduce their memory footprint, but regrettably they also make it difficult to manually parse the files.

This is the most immediate problem solved by GadenTools, which implements the necessary steps to decode this information and make it easily accessible through simple programmatic queries. These queries have been designed to be straightforwardly incorporated to Jupyter notebooks workflows. Moreover, as commented, the implementation has been made efficient by leveraging the advantages of Cython.

The module's core is the `Simulation` class, which permits the user to instantiate a simulation by just specifying the path to its files in the device file system. Behind this instantiation is the reading of Gaden's binary files and their parsing, so the information they encode becomes accessible through the module's API. A `Vector3` class is also provided, which in addition to basic arithmetical operations with 3-dimensional vectors also supports some extra utility functions.

This way, the API permits the MRO user to get information like the current gas concentration or the wind vector at a given location in a certain simulation iteration (expressed in *ppm* and *m/s*, respectively). The API also offers the option to query multiple positions at once. For example, one can extract the gas concentration or wind data of 2D slices of the environment, formatted as multidimensional arrays, with a single query. This functionality has multiple applications (*e.g.* obtaining the ground truth of a concentration map to compare to those generated by an algorithm that is being tested), but probably the most notable is the option to visualize the extracted data as images. The next section further explores this option.

Beyond the ability to manually query specific points of the simulation, as shown earlier, GadenTools can also replicate the traditional behavior of the ROS version of Gaden, playing back the simulation in real time. This is useful when one wants to simulate a robotic agent that takes time to move through the environment and must deal with the fact that gas dispersion is a time-dependent phenomenon. When using this mode, new snapshots of gas dispersion

```
sim = Simulation("/content/DATA/Exp_C/FilamentSimulation_gasType_0_sourcePosition_1.75_8.60_0.50", \
                 "/content/DATA/Exp_C/OccupancyGrid3D.csv")
pointToQuery = Vector3(2.5, 4.0, 0.1)
iteration = 500
print("Gas Concentration at selected point:" + str( round(sim.getConcentration(iteration, pointToQuery), 2) ) + " ppm")
print("3D Wind Vector at selected point:" + str( vector3Round( sim.getWind(iteration, pointToQuery) ) ) + " m/s")
```

```
Gas Concentration at selected point:2.74 ppm
3D Wind Vector at selected point:[0.2, -0.25, -0.0] m/s
```

Fig. 2: Excerpt from the Jupyter Notebook tutorial that shows how to load simulation data and query for single-point measurements of the gas concentration and the airflow vector.

Fig. 3: Illustration of GadenTools accessing the gas concentration and 3D wind vector from a given Gaden simulation. By simply updating the robot position, the API provides the desired measurements.

and airflow information are continually loaded in the background at the specified rate without any further input from the user, and the API provides methods to access the data that pertains to the currently active iteration, rather than to a manually specified one. This functionality was designed to be Jupyter Notebook-friendly, as discussed below.

The GadenTools module also contains a "Utils" submodule, which includes several auxiliary tools which are not strictly required for the core functionality of the module. Besides the `Vector3` class, which has already been discussed, several functions are included to aid in the generation of visualizations, from simply marking the environment's obstacles in the generated images, to encoding the airflow's direction as colors. Section 4.2 explores the topic of generating visualizations in more depth.

4.2 The Notebook Tutorial

The purpose of the Jupyter notebook within GadenTools is to interactively describe its functionalities while illustrating how to use it in different MRO use cases. It starts by describing GadenTools and its API in a similar way as in Section 4.1, but taking advantage of code cells to exemplify its usage. For example, Figure 2 shows a code cell that illustrates how to create a `Simulation` object, also using the API to query the ground-truth gas concentration and wind vector at a certain position and simulation iteration. We can see how the result

of executing this cell code is reported just below. In the notebook, the access to existing simulation data is made available through selected downloadable examples, but one can indeed opt for any other existing resource, including the previously mentioned VGR dataset, or even newly created simulations with custom environments or source parameters.

Once the basic usage is covered, the tutorial guides the user through different GadenTools use cases. It starts by presenting different ways to extract simulation data as images. The ability to visually represent the ground truth data not only helps to gain an insight into the patterns of gas dispersion, and so help guide the development of new techniques and algorithms, but is also a convenient tool when troubleshooting methods of source localization or concentration maps creation that are not working correctly in a given scenario.

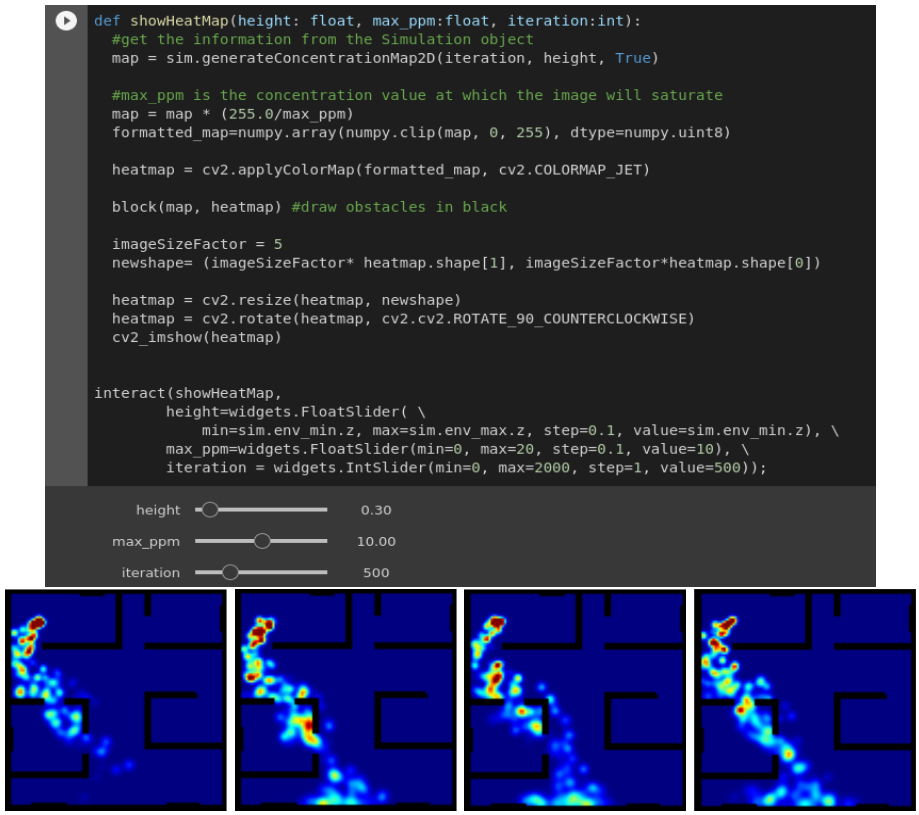

Fig. 4: (Top) Excerpt from the Jupyter Notebook tutorial that shows how to generate a heatmap image of gas concentration on a 2D slice of the environment. In the notebook, a series of interactive sliders lets the user control multiple parameters regarding which information is sampled and how it is shown. (Bottom) Example of gas concentration heatmaps for different iterations (40, 100, 300 and 500).

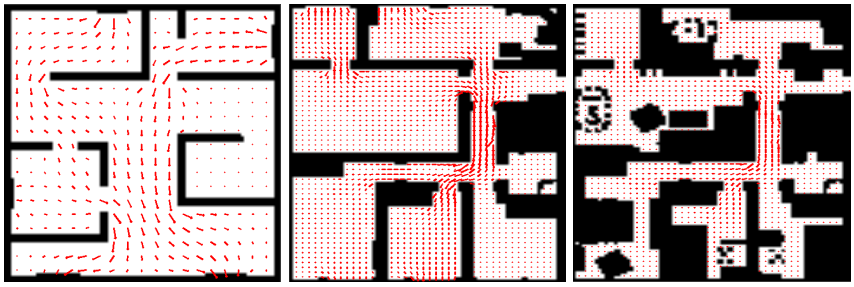

Fig. 5: Visualization of the airflow through the environment as a vector field, generated from the "2D map" method in the GadenTools API. The second and third pictures correspond to different vertical slices of the same environment, and show the variation of the geometry and airflow with height.

In the same way, Figure 3 shows images built upon the provided API with an occupancy map of a house, and gas dispersion at different iterations. The user can simulate a mobile robot (marked as a red point in the images) to inspect said environment by sampling it to get gas concentration and wind vector measurements at different locations. The notebook also challenges the user to find the gas source by guessing its location, giving feedback on the accuracy of their guess. Figure 4 shows a notebook excerpt with a cell code illustrating how to create images with an horizontal slice of the gas concentration as a heatmap, while Figure 5 does the same with the airflow's vectors. In both cases, a minimal GUI gives the user the possibility to set the value of relevant parameters, which helps to gain insight into the simulation behavior.

Once these basic images are obtained, the notebook also exemplifies how to apply simple image processing techniques to them in order to get additional information. Concretely, it illustrates how to retrieve the boundaries of the gas plume in order to know the area affected by the gas dispersion, for which binarization and contour finding techniques are used. A second illustrative example shows how to get concentration gradients to visualize how said concentration changes over space. Figure 6 shows the resulting images after executing these examples.

As discussed in Section 4.1, the API provides the required functionality to play back a simulation in real time. This is also described in the notebook, including an example of its usage to build a video of the gas dispersion at time intervals of 0.5 seconds. This functionality is also included in the concluding use case of the notebook: an implementation of the classic Surge-Cast source localization algorithm [12] (see Figure 7). The Surge-Cast algorithm has the robot move through the gas plume by alternating two movement patterns, an upwind surge and a crosswind cast, according to whether it is currently sensing gas or not. By tracking the plume in this way, the robot eventually reaches the source.

This last use case is particularly illustrative of how a Jupyter notebook can be used for the fast prototyping and validation of algorithms in general, and MRO

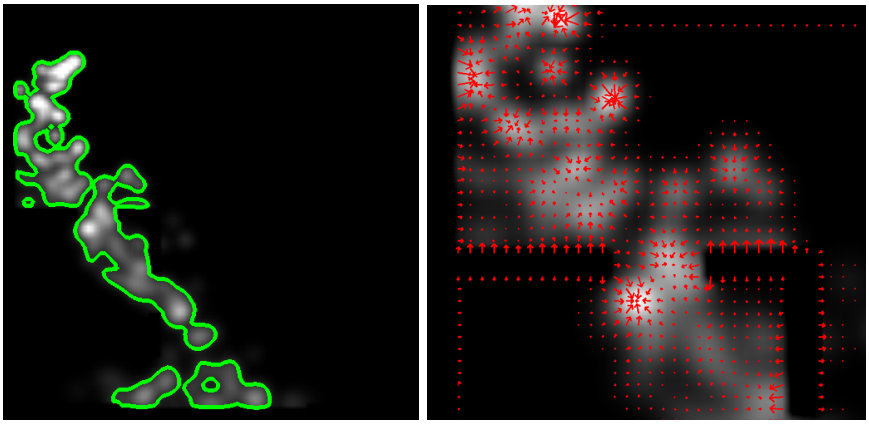

Fig. 6: Examples of using image processing techniques to extract information from the visualizations of the simulation data. The first image shows the contours of the gas plume, while the second image shows the local concentration gradients.

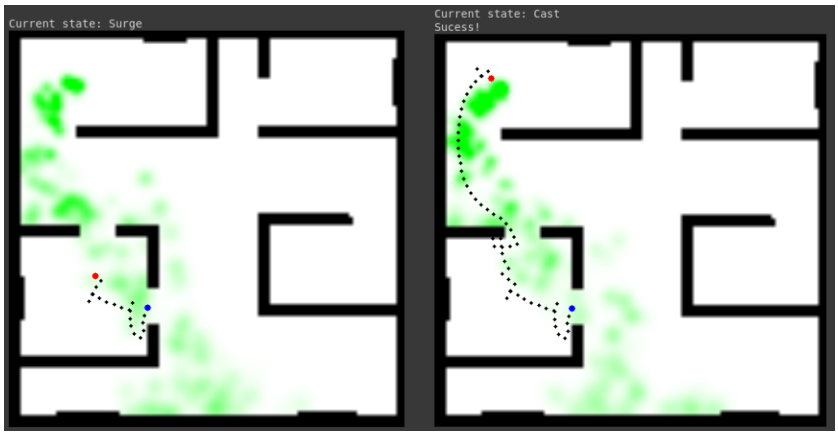

Fig. 7: An example of the proposed toolkit being used to implement an MRO algorithm. Pictured here is the Surge-Cast source localization algorithm. Both the algorithm implementation and the visualization are done fully in-notebook, and the code is presented to help users develop their own methods.

techniques in particular. GadenTools users have the chance the opportunity to continue editing and evolving the notebook according to their needs, or create new notebooks using it, since as shown, the integration of the provided Ptyhon module and Jupyter Notebook is straightforward.

5 Conclusions

This paper has presented the components of GadenTools, which pursue the boosting of research in mobile robot olfaction. For doing so, it contributes an

esay-to-use Python module with a clean API that handles data from gas dispersion simulations. Specifically, the works with simulations produced by Gaden, which include data about the environments where they were carried out (*e.g.* layout and obstacles), as well as airflow and gas dispersion simulations. The paper shows how these binary data, which previously required the installation and configuration of different software dependencies for their manipulation, becomes easily accessible through the module's API. GadenTools also includes a Jupyter notebook, designed with Google Colab, that demonstrates the claimed easy-to-use and accessibility features, and also illustrates a number of use cases showing how it could be a powerful partner in the fast prototyping and validation of MRO techniques. GadenTools is an active project open to comments and contributions from the MRO community.

Acknowledgements. This work was funded by the research projects *HOUNDBOT* (P20_01302 and UMA20-FEDERJA-056), both funding by Andalusia Regional Government and the European Regional Development Fund*ERDF* and by the grant fpr the formation of pre-doctoral researchers in Andalusia (24653).

References

1. Cython. Available Online: https://Cython.org/ (accessed 6 July 2022).
2. Martin Asenov, Marius Rutkauskas, Derryck Reid, Kartic Subr, and Subramanian Ramamoorthy. Active localization of gas leaks using fluid simulation. *IEEE Robotics and Automation Letters*, 4(2):1776–1783, 2019.
3. Joseph R Bourne, Matthew N Goodell, Xiang He, Jake A Steiner, and Kam K Leang. Decentralized multi-agent information-theoretic control for target estimation and localization: finding gas leaks. *The International Journal of Robotics Research*, 39(13):1525–1548, 2020.
4. Sang-Il Choi, Taekyu Eom, and Gu-Min Jeong. Gas Classification Using Combined Features Based on a Discriminant Analysis for an Electronic Nose. *Journal of Sensors*, 2016:9634387, 2016-10-13.
5. Jay A. Farrell, John Murlis, Xuezhu Long, Wei Li, and Ring T. Cardé. Filament-based atmospheric dispersion model to achieve short time-scale structure of odor plumes. In *Environmental Fluid Mechanics*, volume 2, pages 143–169, 2002.
6. Python Software Foundation. Python language reference, version 3.7.x. Available Online: https://python.org/ (accessed 6 July 2022).
7. Andres Gongora, Javier Monroy, and Javier Gonzalez-Jimenez. Joint estimation of gas & wind maps for fast-response applications. *Applied Mathematical Modelling*, 2020.
8. Michael Hutchinson, Cunjia Liu, and Wen Hua Chen. Information-Based Search for an Atmospheric Release Using a Mobile Robot: Algorithm and Experiments. *IEEE Transactions on Control Systems Technology*, pages 1–15, 2018.
9. Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter development team. Jupyter notebooks: a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press, 2016.

10. Tyrell Lewis and Kiran Bhaganagar. A comprehensive review of plume source detection using unmanned vehicles for environmental sensing. *Science of the Total Environment*, 762:144029–144029, 2021.
11. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755. Springer International Publishing, 2014.
12. Thomas Lochmatter and Alcherio Martinoli. Tracking odor plumes in a laminar wind field with bio-inspired algorithms. In Oussama Khatib, Vijay Kumar, and George J. Pappas, editors, *Experimental Robotics*, pages 473–482, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
13. João Macedo, Lino Marques, and Ernesto Costa. A comparative study of bio-inspired odour source localisation strategies from the state-action perspective. *Sensors*, 19(10), 2019.
14. João Macedo, Lino Marques, and Ernesto Costa. Locating odour sources with geometric syntactic genetic programming. In Pedro A. Castillo, Juan Luis Jiménez Laredo, and Francisco Fernández de Vega, editors, *Applications of Evolutionary Computation*, pages 212–227. Springer International Publishing, 2020.
15. Javier Monroy and Javier Gonzalez-Jimenez. Gas classification in motion: An experimental analysis. *Sensors & Actuators: B. Chemical*, 240:1205–1215, 2017.
16. Javier Monroy, Victor Hernandez-Bennetts, Han Fan, Achim Lilienthal, and Javier Gonzalez-Jimenez. GADEN: A 3D gas dispersion simulator for mobile robot olfaction in realistic environments. *MDPI Sensors*, 17(7):1–16, 2017.
17. Pepe Ojeda, Javier Monroy, and Javier Gonzalez-Jimenez. Information-Driven Gas Source Localization Exploiting Gas and Wind Local Measurements for Autonomous Mobile Robots. *IEEE Robotics and Automation Letters*, 6(2):1320–1326, 2021.
18. Pepe Ojeda, Javier Monroy, and Javier Gonzalez-Jimenez. VGR dataset: A CFD-based gas dispersion dataset for mobile robotic olfaction. 2022 - To appear.
19. Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, may 2009.
20. Ronnier Frates Rohrich, Luis Piardi, José Luis Lima, and André Schneider de Oliveira. Bio-inspired distributed sensors to autonomous search of gas leak source. In *2020 International Conference on Manipulation, Automation and Robotics at Small Scales (MARSS)*, pages 1–6, 2020.
21. Jose-Raul Ruiz-Sarmiento, Samuel-Felipe Baltanas, and Javier Gonzalez-Jimenez. Jupyter notebooks in undergraduate mobile robotics courses: Educational tool and case study. *Applied Sciences*, 11(3):917, 2021.
22. Yuta Wada, Marco Trincavelli, Yuichiro Fukazawa, and Hiroshi Ishida. Collecting a Database for Studying Gas Distribution Mapping and Gas Source Localization with Mobile Robots. In *The Abstracts of the International Conference on Advanced Mechatronics : Toward Evolutionary Fusion of IT and Mechatronics : ICAM*, volume 2010.5, pages 183–188, 2010.
23. S. Waphare, D. Gharpure, A. Shaligram, and B. Botre. Implementation of 3-nose strategy in odor plume-tracking algorithm. In *2010 International Conference on Signal Acquisition and Processing*, pages 337–341, 2010.