UNIVERSIDAD
DE MÁLAGA

E.T.S.
INGENIERÍA
INFORMÁTICA

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

EVALUACIÓN DE ALGORITMOS DE BÚSQUEDA DE FUENTES DE GAS PARA ROBOTS MÓVILES BAJO MÚLTIPLES CONDICIONES AMBIENTALES

AN EVALUATION OF GAS SOURCE LOCALIZATION ALGORITHMS UNDER MULTIPLE ENVIRONMENTAL CONDITIONS

Realizado por
José Ojeda Morala

Tutorizado por
Javier González Jiménez
Javier González Monroy

Departamento
Ingeniería de Sistemas y Automática

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2017

Fecha defensa:

Fdo. El/la Secretario/a del Tribunal

Abstract

The term Gas Source Localization (GSL) refers to the problem of finding the point in an environment a gas is being emitted from. There are numerous practical applications for GSL techniques, such as finding gas pipe leaks or locating explosives and contraband. The use of mobile robots equipped with electronic noses offers a promising approach to solve this problem, but is still an open research field with many unsolved challenges. One circumstance that hinders the development of Gas Source Localization methods is the lack of comparative experimental studies that show the performance of the existing strategies. The present work aims to provide one such study of several widely known GSL algorithms under diverse environmental conditions, including realistic environments that feature obstacles and turbulent airflows. The experiments that comprise this study will be carried out using GADEN, a gas dispersion simulator for robotic olfaction, and robotic tools offered by ROS, one of the most widespread robotic architectures.

Keywords: Robotic olfaction, gas source localization, ROS.

Resumen

El término Búsqueda de Fuentes de Gas (GSL, del inglés Gas Source Localization) hace referencia al problema de encontrar el punto de un entorno desde el que un gas concreto está siendo emitido. Existen numerosas aplicaciones prácticas para las técnicas de búsqueda de fuentes de gas, incluyendo la localización de fugas en tuberías y la detección de sustancias explosivas o contrabando. El uso de robots móviles equipados con narices electrónicas presenta una perspectiva prometedora a la hora de abordar este problema, pero aún es un campo de investigación abierto con numerosos desafíos por resolver. Una de las circunstancias que dificulta el desarrollo de métodos de búsqueda de fuentes de gas es la ausencia de estudios comparativos que muestren la eficacia de las estrategias existentes. El presente trabajo pretende aportar un estudio de esas características sobre el rendimiento de varios conocidos métodos de búsqueda de fuentes bajo diversas condiciones ambientales, incluyendo entornos realistas con flujo de aire turbulento. Los experimentos que componen este estudio se llevarán a cabo utilizando GADEN, un simulador de dispersión de gases desarrollado para aplicaciones robóticas, y la arquitectura robótica de ROS.

Palabras clave: Robótica olfativa, localización de fuentes de gas, ROS.

Index

1

Introduction

1.1 Motivation

Gas Source Localization (GSL) refers to the problem of localizing one or multiple gas emission sources within a real-world environment [1]. This includes, for example, the localization of gas pipe leaks in industrial facilities [2], the identification and location of illegal substances for contraband interception [3][4], or the pinpoint of pollution-related emissions that may affect high density population areas [5] among others.

Traditionally, this problem has been tackled with the help of animals, deploying networks of fixed chemical sensors (e.g. pollution monitoring stations [6][7], see Figure 1), or by means of portable handheld gas sensors carried by expert operators [8]. Even though these solutions might be acceptable for some scenarios, often the scale of the problem (i.e. large search areas where a fixed network of sensors is impractical) or the dangerous environmental conditions the operators or animal assistants would be exposed to (i.e. toxic gases, high temperatures) make them unfeasible. For these reasons, it is an interesting perspective to have mobile robots equipped with chemical sensors to take the role of the operator/animal in the search process [4][9][10].

Mobile Robotic Olfaction (MRO) is the branch of robotics that revolves around the use of mobile robots capable to detect gases in the environment. It is an active research field with many important technological and scientific challenges yet to be solved [11]. Nevertheless, the recent advances in the design and manufacturing of portable gas sensing devices, usually referred to as electronic noses (e-noses) [12][13], as well as in applying and adapting signal processing techniques to this field [3][14][15] have already positioned MRO as a promising solution in many practical applications.

In general, the problems addressed by MRO can be sorted into three main topics: identification or classification of a measured volatile [16][17], estimation of the gas dispersal through the creation of distribution maps [18][19] and the localization of the

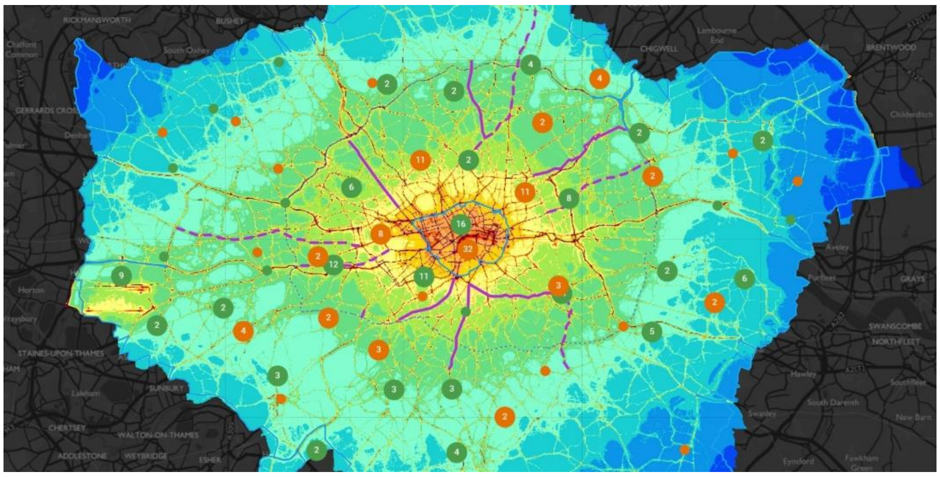


Figure 1. Air quality map of London published by the government of the city (From [1]).

gas emission source (from now on, Gas Source Localization or GSL) [20][21][22]. In this work we focus on GSL, particularly in the localization of gas sources in realistic, complex environments with the use of a mobile robot.

Gas dispersion is a complex phenomenon, particularly in realistic indoor environments, where the airflow often presents a turbulent nature [23][24]. Laminar airflows make the gas disperse in the form of a down-wind plume [25], but turbulent airflows cause the plume to break into independently moving patches (Figure 2), notably increasing the complexity of the search process.

Multiple approaches have been proposed along the last decades to tackle the problem of GSL with a mobile robot (See section 3.1 for a detailed review). Because of the complexity of the problem, many of these proposals make assumptions about the nature of the source, the environmental conditions (characteristics of the airflow, presence of obstacles) or the sensors the robot is equipped with. A common drawback in these works is for the performed experimentation to be limited to unrealistic conditions, as well as a lack of objective comparisons with other methods that try to present a solution to the same problem. This issue makes it difficult to evaluate the proposed methods, or to select the most convenient method for each specific application, as well as hindering the development of new alternative solutions.

This work contributes with an experimental evaluation of the performance of some of the most notable GSL algorithms for mobile robotics when utilised in different environmental conditions. The goal is to provide an objective comparison that may be used by the scientific community to select the most appropriate GSL methods for specific environmental conditions, as well as to provide an open-source implementation of the chosen methods to facilitate any further experimentation with them. To perform this evaluation we rely on simulation tools, which will allow us to compare the performance of the different methods under identical conditions and to set up multiple

[1] https://www.london.gov.uk/what-we-do/environment/pollution-and-air-quality/london-air-quality-map

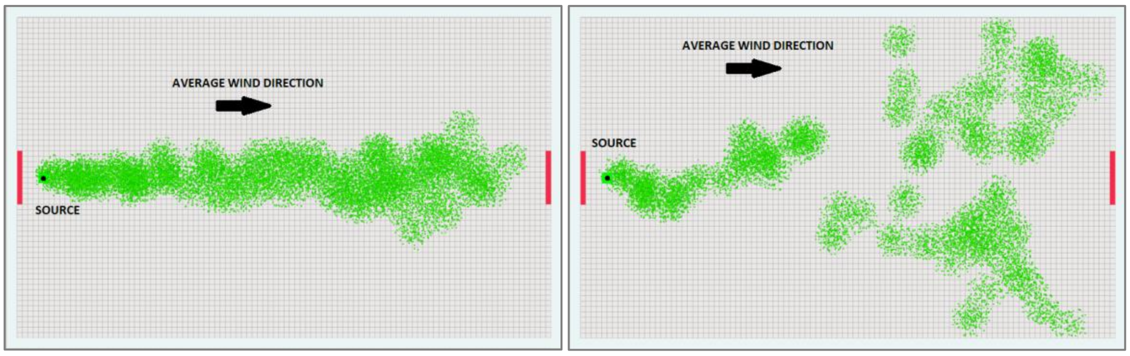


Figure 2. A laminar airflow causes the gas to disperse in the form of a plume (a), but turbulences bend the plume or break it into disconnected patches (b).

configurations that would be unfeasible with real experiments. In the next section we detail the contributions of this work, and outline the structure of the document.

1.2 Objectives

This work aims to present a comparative experimental study of some of the most widely known gas source localization algorithms, highlighting their strengths and weaknesses. These experiments will be carried out using GADEN[2] [26], an open-source gas dispersion simulator (GDS) specifically designed for the development of mobile robotic olfaction techniques.

The process of the study has been divided into three main tasks:

- **Implementing the algorithms**. Many GSL publications do not offer an open-source implementation of the algorithm they present, and some methods are developed only for specific robotic hardware, which means using the described algorithms requires implementing them first.

  As mentioned, the experiments that compose this work will be carried out using GADEN, which runs on the widely used Robotic Operating System[3] (ROS) [27]. Using ROS to make open-source implementations of the chosen algorithms will make these implementations accessible and easily adaptable to run on real-world robots. These implementations will be available in an online repository [4].

- **Adapting the algorithms to navigate environments with obstacles**. Some of the chosen algorithms have been developed to find the gas source in an empty workspace, and therefore do not include any obstacle-avoidance behaviour. In order to be able to carry out the experiments in a wider range of environmental

[2] http://mapir.isa.uma.es/work/gaden
[3] https://www.ros.org/
[4] https://github.com/MAPIRlab/Gas-Source-Localization

conditions, some new obstacle-handling behaviour will be included when necessary.

These algorithms will be adapted to work with goal-based navigation through the use of the navigation tools provided by ROS, rather than velocity commands, which will allow the robot to more effectively navigate environments that feature obstacles.

- **Experimentation**. A series of experiments will be carried out to study the performance of the algorithms in different environmental conditions. Furthermore, a thorough analysis and discussion of the results is provided to better contextualize the numerical results, trying to provide a better understanding of the applicability of each method to specific environmental conditions.

  Using a simulator for these experiments will allow a greater control of the setup, guaranteeing that the conditions are the same for each algorithm.

1.3 Structure of the Document

This document includes the following sections:

- **Used Technologies:** A description of the main tools utilized for the development of the implementations of the algorithms, along with a brief explanation of how they work. These technologies are ROS itself, along with the Stage and GADEN simulators.

- **State of the art:** An overview of some the most popular approaches to gas source localization and their general advantages and weaknesses, highlighting some specific algorithms and introducing some of the concepts that will be further explored when explaining the algorithms implemented in this work.

- **Implemented algorithms:** A detailed explanation of the strategy and concepts utilized by each of the algorithms that have been chosen to be implemented in this work. Any modifications done to the algorithms (mostly in obstacle handling behaviours), as well as any relevant implementation details, will be explained in this section.

- **Experiments and results:** In order to study the performance of each of the algorithms, several experiments will be carried out using the aforementioned simulation tools. This section includes a description of the setup of each of the experiments, as well as the values assigned to the most important parameters and a study of the results obtained.

- **Conclusions and future work.** The analysis of the results of the previously mentioned experiments will be used to justify conclusions about the applicability of the different GSL methods to specific environmental conditions. Furthermore, some related lines for future research will be proposed based on these conclusions.

2

Used Technologies

2.1 ROS

2.1.1 Description

The Robotic Operating System [27], usually referred to as ROS, is a bundle of packages and libraries that provide numerous tools for developing software for robotic devices. It not only provides a programming framework, but also a way to easily establish communication between robotic hardware and computers.

ROS is a good choice for developing robotic software because it provides tools for the many functionalities required by such innately multi-tasking systems, on top of being compatible with the most popular robotic hardware and having the trustworthiness and reputation of having been used for many academic and commercial projects.

Robotic systems are different from other multi-process systems in that they need to interact with the real world. This means that a robotic framework requires the development of diverse and complex functionalities, which makes it necessary for many experts to work on the project. This would be very costly and inefficient for each business or group to do independently.

ROS is fundamentally open-source, allowing any advances made in any aspect of robotic software to easily be shared between experts. This not only speeds up the development of the framework itself, but it also facilitates the development of software for specific tasks, allowing access to the already existing libraries and packages that handle basic robot behaviour and low-level device control.

2.1.2 Structure and Main Concepts

ROS utilizes a modular architecture that allows different processes to run independently or communicate through messages. Some of the most important concepts to understand this system are[5] (Figure 3):

- **Node:** Comparable to the concept of "process" on most operating systems. A given node will usually handle one specific functionality (sensing, planning), and the development of complex robotic methods will require intercommunication among different nodes. A node is identified by a name, ID and port.

- **Master node:** The master node handles the registration of other nodes and organizes their communication by allowing nodes to find each other.

- **Package:** Packages are directories that contain a specific file (*package.xml*) that identifies them for the master node and gives a description of the code contained in it. They are used to organize the code and establish dependencies for the compilation and execution of the code. They usually contain nodes or libraries.

- **Message:** Messages are the way ROS nodes communicate. They are data structures that contain typed data, and natively support primitive types (integers, floating point numbers) and arrays, although they can be defined for more complex data structures. Messages can delivered either through TCP or UDP, which means it is possible for ROS nodes that run on different devices to communicate with each other.

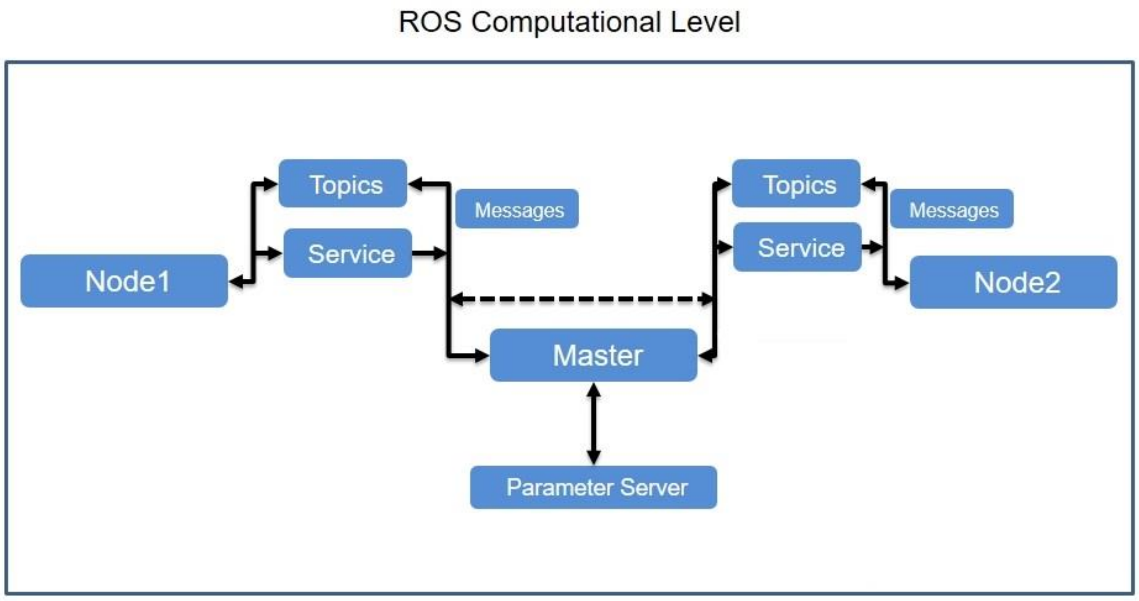


Figure 3. Graph that shows the relations among the most important ROS concepts. (Adapted from [28]).

[5] http://wiki.ros.org/ROS/Concepts

- **Topic:** Topics provide the structure for messages to be shared. They define a specific kind of message and allow nodes to choose to receive them. A node can **subscribe to** or **publish on** a specific topic. Whenever a node publishes a message on a topic, the message will be sent to all nodes that are subscribed to it . With the use of topics, the user does not have to handle inter-node communication directly. The master node keeps track of all topic subscriptions and communicates them to publisher nodes so that they can engage in peer to peer communication.

- **Service:** Services offer an alternative to Topics for inter-node communication. Since Topics are inherently one-way one-to-many forms of communication, they are not appropriate for request-reply dynamics. Services offer that functionality, defining a type of message for the requests and another for the replies.

- **Action Server:** Action servers, similarly to Services, offer a form of communication based on requests and replies, but are geared to be used for the execution of preemptable tasks. Action servers receive from the client a **goal**, which determines the task that needs to be executed, and sends back **feedback** (during the execution) and **result** (when completed or interrupted) messages.

2.1.3 Notable Packages

Some ROS packages play an important role in the present work. The most notable ones include:

- **Move_base:** The *move_base* package[6] functions as an Action Server that receives a message specifying a goal pose in a given map and sends velocity commands to the robot to guide it there.

  To do this, *move_base* divides the given map into a grid of cells and assigns each cell a "cost" (that depends on how close it is to an obstacle), creating what is known as a *global costmap*. This costmap is used to plan the optimal path to the goal, and, while the robot is moving, the sensory information is used to create a *local costmap* that allows to reactively modify the planned path in case of unexpected obstacles or faulty odometry.

  By using this node, navigation is reduced to deciding a goal, delegating the low-level path planning and obstacle handling logic to the *move_base* server, which facilitates the implementation of algorithms such as the ones included in this work.

- **Stage:** Stage[7] is a widely used robotic simulator. Originally part of the Player/Stage [29] project, it is currently available as a ROS package[8]. It is the tool used in GADEN simulations to handle the robotic localization and movement. It

[6] http://wiki.ros.org/move_base
[7] http://rtv.github.io/Stage/
[8] http://wiki.ros.org/stage

is a powerful tool that allows for the creation on a navigable environment and provides the sensing tools (lasers, cameras) necessary for navigation.

- **GADEN:** GADEN is a gas dispersion simulator for robotic olfaction. Since it is one of the most important tools for this work, it will be explained in detail in the next section (Section 2.2).

2.2 GADEN

GADEN is a gas dispersion simulator designed for the development of robotic olfaction algorithms. As was mentioned before, the use of a simulator provides a common framework for the implementations of algorithms to be shared by the scientific community, which is one of the main objectives of this work.

Amongst the other options for the choice of a simulator (see State of the Art, Section 3.2), GADEN is selected due to presenting several advantages:

- **Environment definition:** GADEN permits the definition of three-dimensional environments with complex shapes through the use of CAD models. Since we intend to study the performance of the algorithms in different environmental

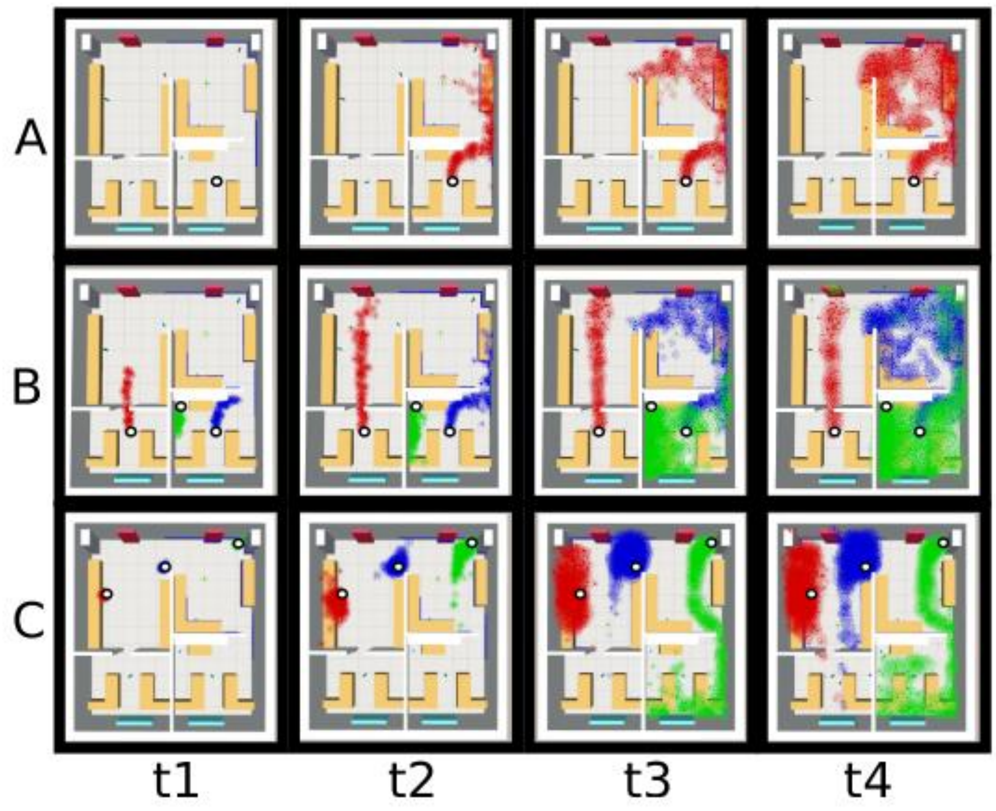


Figure 4. Snapshots of gas dispersion in with different setups (A-C) in several time instants(t1-t4) that shows how GADEN supports gas dispersion simulation in complex three dimensional environments with multiple simultaneous sources and gas types. (From [21]).

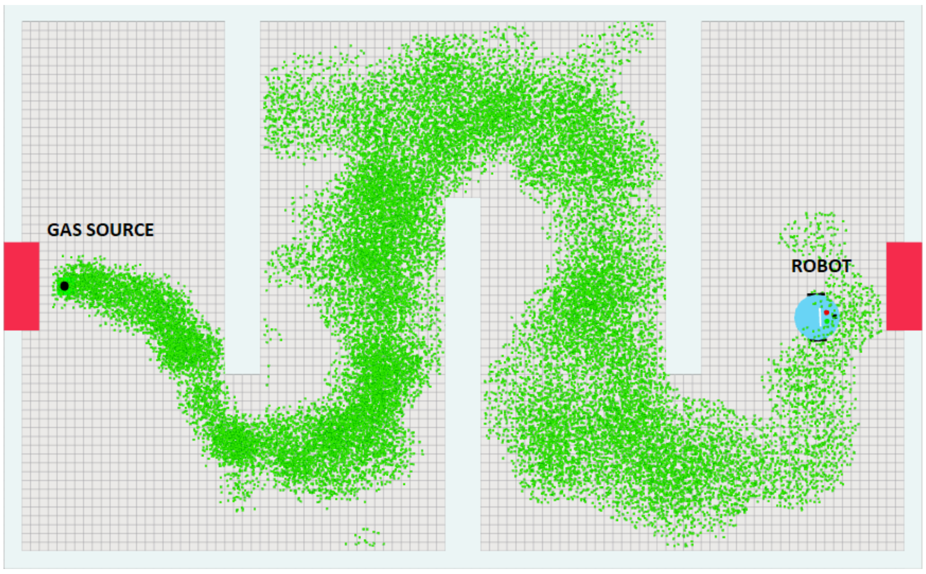


Figure 5. Execution of a simulation using GADEN. As can be observed, it allows integration of the gas dispersion simulation with robotic tools.

conditions, being able to easily create multiple environments with different characteristics is an important advantage.

- **Integration:** GADEN is one of the few GDS that has been specifically designed to be used in the development of MRO methods (Figure 5). For that reason, it innately provides an integration of the gas dispersion simulation and robotic tools that the more specialized CFD tools lack. Being implemented in ROS, the code of algorithms implemented for GADEN can mostly be reused for deployment on real robots.

- **Dispersion model:** GADEN uses the filament dispersion model [30], which represents the gas released as a series of puffs, each of them consisting of several filaments. A filament, in turn, represents a 3D gaussian dispersion of gas particles. The filaments move and expand independently, which allows a higher degree of realism in dealing with obstacles and turbulences than would be possible using a more simplistic model, such as the gaussian plume [25]. This more sophisticated model also means that GADEN is capable of handling complex simulations with multiple release points or different types of gas (
- Figure 4).

GADEN simulations follow a three-steps structure (Figure 6): (I) firstly, the airflow data is generated using an external CFD tool, (II) then that wind data is used to perform the gas dispersion simulation, (III) and lastly the results of the gas simulation are played back in a manner they can be utilized by a robot.

Dividing the simulation in these separate steps is convenient since GADEN uses a complex gas dispersion model with a high computational cost, and therefore the gas simulation is very time consuming. This way, the gas simulation only needs to be computed once in advance, the results are stored in log files that record the airflow vectors and gas concentration values for each

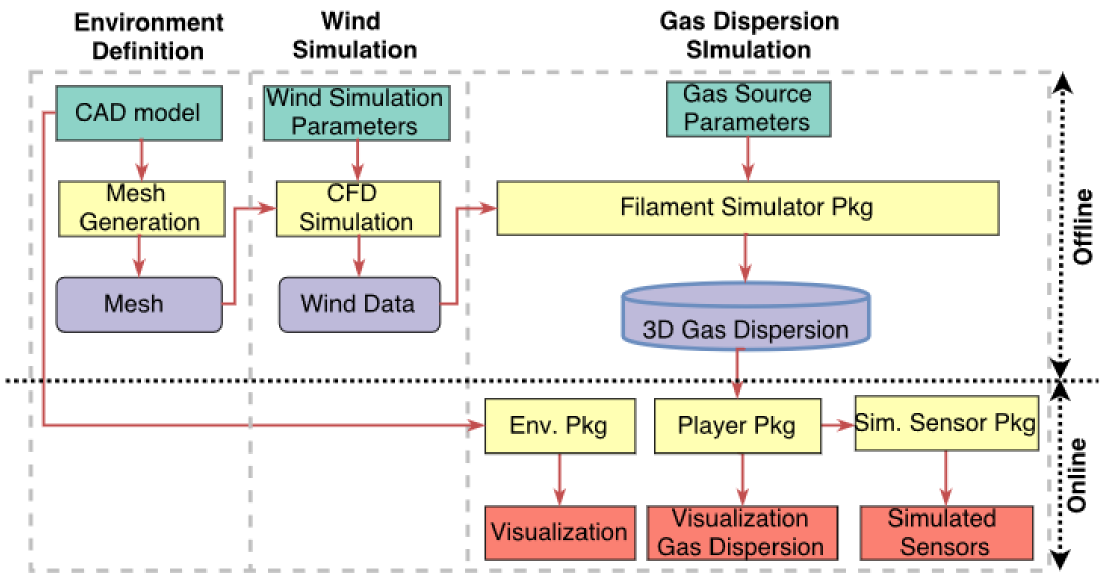


Figure 6. Block diagram that represents the simulation process for GADEN. (From [21])

point of the environment, and the generated data can be used as many times as necessary at a low computational cost while running any robotic experiments.

As well as the gas simulation itself, GADEN offers simulated versions of several gas sensors, such as MOX or PID, and an anemometer. The implementations of these sensors simulate the sensor noise, which permits a greater degree of realism in the experiments.

3 State of the Art

In this section we will present some popular approaches to gas source localization to better contextualize the chosen algorithms, as well as a brief overview of the available simulation tools that might be applicable for this work.

3.1 Gas Source Localization Algorithms

The localization of a gas source is not a simple task. Gases are distributed in the environment through diffusion (dominant when the airflow is weak) and advection (dominant when the airflow is strong). The airflow can be time-variant and/or turbulent and depends greatly on the shape of the environment and the temperature and humidity conditions.

When the environment presents a laminar airflow the gas is distributed in the form of a plume [25], but turbulences break the plume into "patches" that move independently (Figure 2). This means gas does not necessarily disperse in straightforward patterns, so it is possible for a gas patch measured far from the source to have followed an unpredictable path, particularly in complex environments that feature obstacles [24][31]. This makes it difficult to estimate where the gas measured at a given location might have been released from.

GSL algorithms can be broadly classified as belonging to one of two main branches: probabilistic or reactive (for a more detailed taxonomy, see [1]).

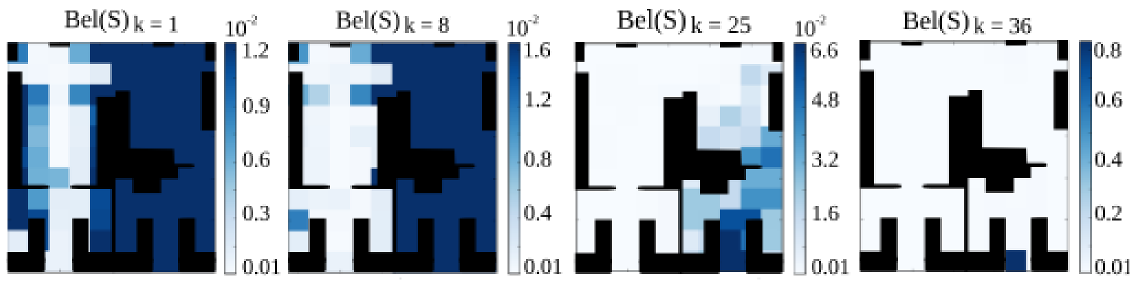


Figure 7. Example of the progression of a probabilistic algorithm. Different sections of the map are assigned a probability of containing the source (represented, in this figure, by the darkness of the blue colour), and those probabilities are updated with every new observation. (Adapted from [33])

3.1.1 Probabilistic Methods

Due to the mentioned difficulties, algorithms based on statistical inference to estimate the location of the source usually rely on previous assumptions that simplify the environmental conditions (such as laminar/homogeneous/ constant airflow) [22][32].

By assuming these conditions are met, it becomes possible to develop analytical models for the dispersion of the gas, permitting probabilistic searches. However, even though these assumptions might perform well for certain specific applications, but they are not generally realistic.

Another strategy for probabilistic gas source localization methods consists in using databases of gas and wind measurements to extrapolate the conditions of other points in the environment from the measurements of the current robot position [33]. This approach eliminates the need to simplify the environment, allowing for the application of these algorithms in more realistic conditions, but requires a previous acquisition of experimental data for that specific environment, which can be impractical.

In most cases, probabilistic methods will use these mentioned strategies to create a Probability Density Function (PDF) that gives the likeliness of a given point of the environment being the actual location of the source, and then rely on some statistical inference technique (Bayesian inference [33], a particle filter [32]) to modify this probability function after each measurement (Figure 7).

3.1.2 Reactive Methods

Because of the difficulty to analytically solve the problem, many proposed algorithms do not try to estimate the location of the source, but rather to devise a movement strategy that allows the robot to reactively navigate towards it.

The behaviour of certain types of insects has been an important inspiration for these reactive strategies [34][35][36]. These bioinspired algorithms can be divided into chemotactic and anemotactic strategies.

Chemotactic strategies use the measured gas concentration to guide the movements of the robot. They are usually designed to be utilized in environments where there is no strong airflow and gas is mostly dispersed through diffusion, which creates a concentration gradient. Some of the most utilized chemotactic strategies include [35]:

- The movement pattern of the E.coli bacteria [37], which is a biased random walk. Whenever a gas measurement is taken, the robot makes a random rotation and advances a random distance. When the gas concentration measured is larger than the previous one, the rotation is small and the distance advanced is large, and vice versa.

- Spiral [34], a strategy that uses a growing spiral movement to get closer to the source, and restarts the spiral when the gas measurements indicate it is closer to the source than it was before (see Section 4.1.2).

- Gradient climbing [37], which has several variants, but revolves around the use of two different gas sensors to measure the gas concentration in several points at each time instant to determine the direction of the concentration gradient, and then moving following it.

Anemotactic strategies use wind information to guide the movements of the robot, and therefore are appropriate for environments with a strong, measurable airflow, where advection is a main component of the gas dispersion.

The most notable subcategory of anemotactic algorithms is plume-tracking [38], which assumes the existence of a downwind gas plume and tries to find the source by moving through it (Figure 8). Some examples of plume-tracking algorithms include:

- The Silkworm Moth [21] algorithm, which combines short straight movements when detecting odour with upwind zig-zagging and circular movements when not detecting it.

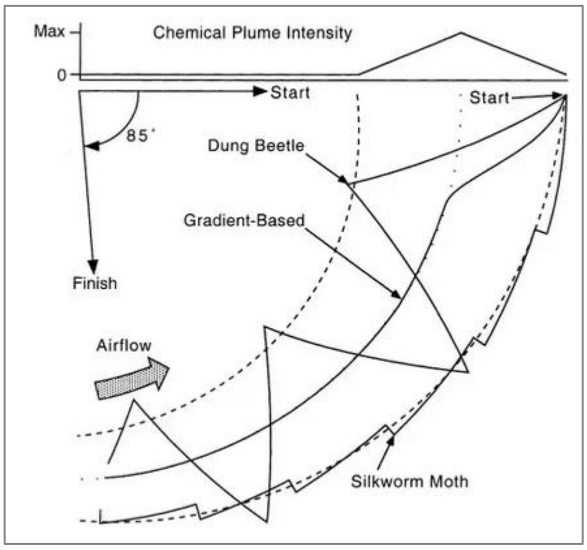


Figure 8. Diagram that shows the paths followed by the mentioned anemotactic algorithms when moving within a gas plume. (From [37])

- The Dung Beetle algorithm [39], which similarly uses upwind zig-zag movements, but uses them to track the plume by moving all the way to its limit and then returning towards the centre, rather than to recover it.

- Surge-Cast [38] (see Section 4.1.1), and Surge-Spiral [40] (see Section 4.1.3), which rely on the use of upwind movements (surges) to move within the plume and try to quickly recover it when lost.

Although many such bioinspired algorithms have been demonstrated to perform well in simplified test environments, it is currently unclear whether they are appropriate for the more complex environments required for real applications [41].

3.2 Gas Dispersion Simulators

Simulation tools play an important role in the development of most scientific and technical fields, and robotic olfaction is no exception.

Real-world experiments are costly and difficult to set up or replicate, since there are many environmental variables that can influence the behaviour of the airflow and thus change the way gas is dispersed. On top of this, it is difficult to obtain ground-truth values for gas dispersion [42], which greatly complicates the evaluation of the results.

Because of this, the use of simulation tools is a great help in simplifying the initial steps of experimentation. Also, and relevantly for the present work, the use of a simulator provides a common framework for the implementations of MRO algorithms to be shared by the scientific community, avoiding the issues that come from the use of different robotic hardware.

Amongst the available options for simulating gas dispersion phenomena, many are not designed to be used for robotic olfaction. Such is the case of tools like ANSYS[9] or OpenFoam[10], which provide advanced computational fluid dynamics (CFD) models that are able to achieve great accuracy even in complex environments [24], but do not allow for easy integration with robotics simulation. Also, the complexity of these simulators makes them difficult to use for anyone who lacks expertise in CFD. For those reasons, the application of these simulators to robotic olfaction is limited.

On the other hand, we can find tools like PlumeSim [43], which was designed for the Player/Stage framework [29], or the gas dispersion simulator based on OpenMORA[11] proposed by Monroy *et al.*[44], which are actually designed for robotic olfaction, but present some other relevant limitations, such as only permitting bidimensional simulations, relying on simplistic gas dispersion models, or lacking integration with the most recent robotic tools.

[9] https://www.ansys.com/
[10] https://www.openfoam.com/
[11] https://github.com/OpenMORA

For this work, we chose to use GADEN (see section 2.2). GADEN solves many of these issues, allowing us to define complex three-dimensional environments through the use of CAD models, using the filament dispersion model and working with most currently used robotics tools due to being implemented on ROS.

4

Implemented Algorithms

4.1 Description of the Algorithms

Amongst the algorithms mentioned in the previous section (section 3.1), four particular ones have been chosen to be implemented and used for the comparative study. The next sections explain in detail how they work.

4.1.1 Surge-Cast Plume Tracking

Plume tracking [38] is one of the most utilized GSL reactive strategies. It is based on the idea that, if gas is being distributed in the form of a plume, once the plume has been found the robot has only to follow it in order to find the source.

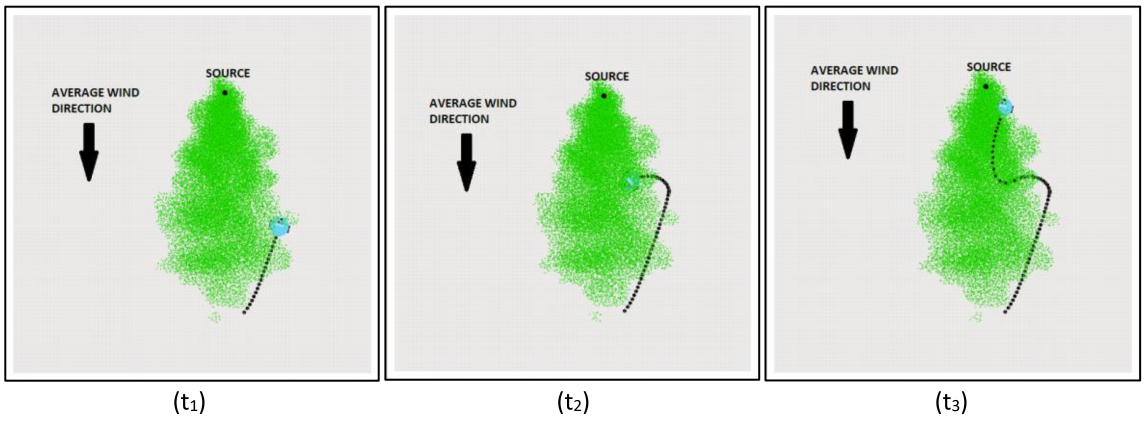


Figure 9. Execution of the Surge-Cast algorithm. The robot moves directly upwind when it measures gas, and crosswind when it does not. Each of the subfigures shows a different time instant.

As was discussed earlier, in complex environments plumes are often broken into different patches, which can pose a problem for this strategy [45].

It is common for plume tracking algorithms to use a state machine (Figure 10) consisting of at least two states: one to move within the plume, and one to recover the plume if the robot has lost it. In the case of Surge-Cast (Figure 9), those two core states are defined as follows:

- **Surge:** This state is used when the robot is inside of the plume, that is, gas has been detected. The movement direction will always be directly upwind, since that is the direction the gas is likely coming from. The robot stops after a fixed distance to resample the wind direction, or if it loses the plume.

- **Cast:** The state for recovering the plume after losing it. The robot performs a swipe perpendicular to the measured wind to try and find the plume. The movement stops if the robot measures gas.

For this specific implementation of the algorithm, which is based in the one used by Monroy *et al.* [45], some extra states have been defined in order to make the search more robust:

- **Stop and measure:** Whenever a choice needs to be made with respect to which state comes next, the robot stops in place for a fixed length of time and takes wind and gas measurements. In order to make the next decision, the averages of the measurements are used. This helps reduce the impact of sensor noise and get around small airflow turbulences that would otherwise disrupt the strategy.

- **Wander:** When the robot measures gas but the wind speed is too low it cannot determine which direction to move towards. To solve this, the robot moves to several nearby positions and repeats the measurement phase, trying to find a more helpful measurement.

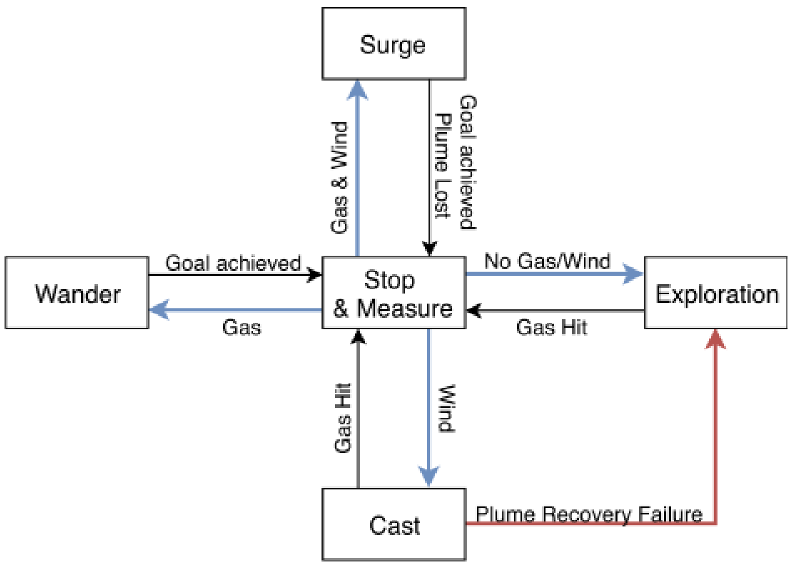


Figure 10. State machine used by the Surge-Cast algorithm. (From [45])

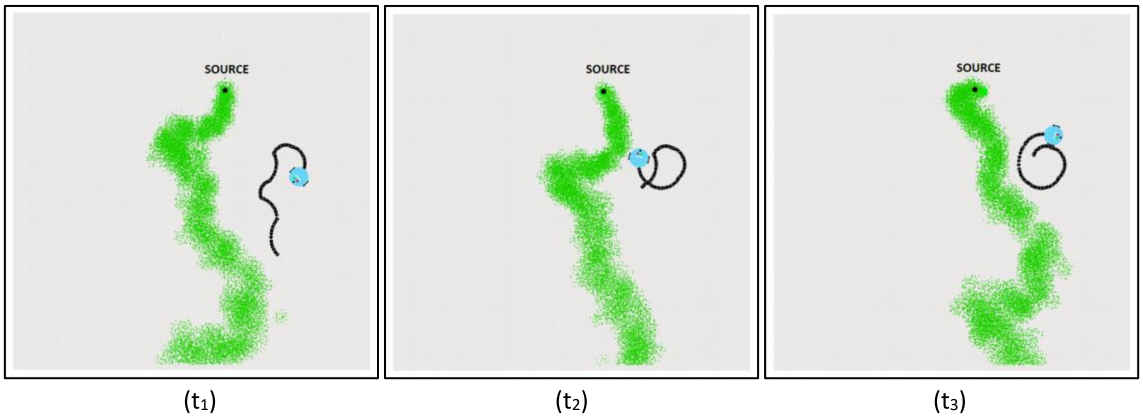


Figure 11. Execution of the Spiral algorithm. The robot moves in following a spiralling pattern until it infers from the gas measurement that it is closer to the source than it was when it began moving, and then it restarts the spiral. Each of the subfigures shows a different time instant.

The strategy of this algorithm is to have the robot never leave the plume, with the upwind surge movement being interrupted as soon as the robot stops measuring gas, and using a crosswind cast to quickly regain the plume.

Other parameters, such as gas concentration and wind speed thresholds, will be explained in more detail in the experiments section (Section 5).

4.1.2 Spiral

The Spiral GSL method, presented by Ferri *et al*. [34] in 2009, is a reactive GSL algorithm. Of the algorithms implemented for this work, Spiral is the only purely chemotactic one, and therefore does not require the robot to be equipped with an anemometer.

It defines a simple strategy by which the robot always follows a spiralling movement pattern, and the only decision to be made is whether to continue the current spiral or

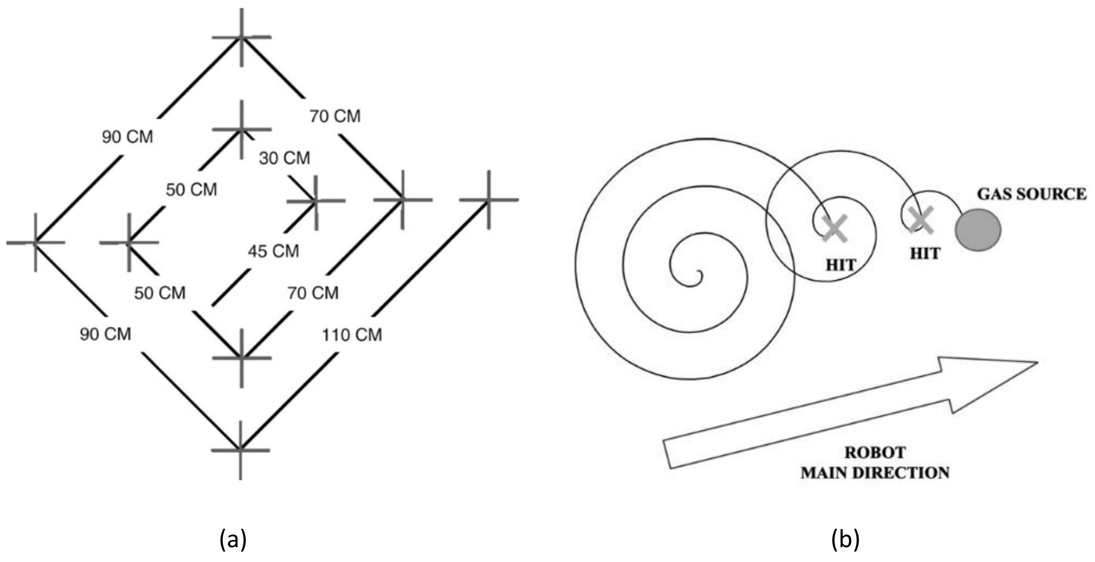


Figure 12. (a) Path followed by the robot with the Spiral method. The crosses represent measurement points. (b) Restarting the spirals when the Proximity Index in one of those points is larger than when the spiral was started creates a general movement towards the source. (From [34])

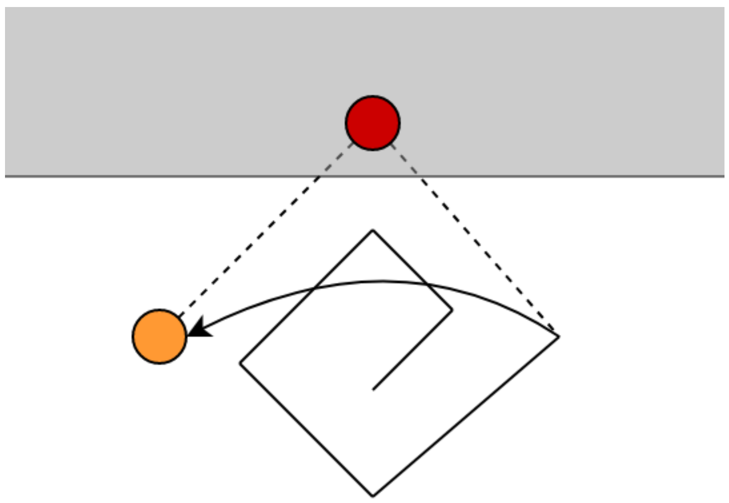


Figure 13. New obstacle-handling behaviour implemented for Spiral. When the next point is unreachable, the spiral is continued after skipping it.

to start a new one. In order to decide the next step, the robot stops to measure gas at four fixed points for every turn of the spiral (Figure 12a). The spiral is reset in case of a gas hit, and continues otherwise.

For this algorithm, a "gas hit" is not defined in terms of a fixed concentration threshold, but rather a dynamic parameter called Proximity Index, or PI, that also factors in the intensity of concentration peaks (see [34] for details on how it is calculated) and serves as a heuristic representation of how close the robot is to the source. A gas hit occurs when the current PI is larger than the one calculated when the spiral was initiated. The absence of a hit is called a "miss".

The algorithm includes also a mechanism for lowering the PI after several consecutive misses to prevent the robot from getting stuck in an ever increasing spiral after an unusually high gas measurement.

The strategy of the algorithm is that, while inside of a plume, the restarting spirals create a general movement in the direction of increasing PI values, which moves the robot towards the source (Figure 12b); and when the plume is lost or broken into patches, the spiralling movement covers the area around the robot allowing it to find gas again.

The implementation of Spiral presented in this work modifies the obstacle-handling behaviour included in the original publication to adapt it to the higher-level goal based navigation utilized. Since it is possible to calculate it beforehand, whenever the next point in the spiral falls in an unreachable position (within or too close to an obstacle), that point is skipped and the robot is instead sent to the point that would follow it (Figure 13).

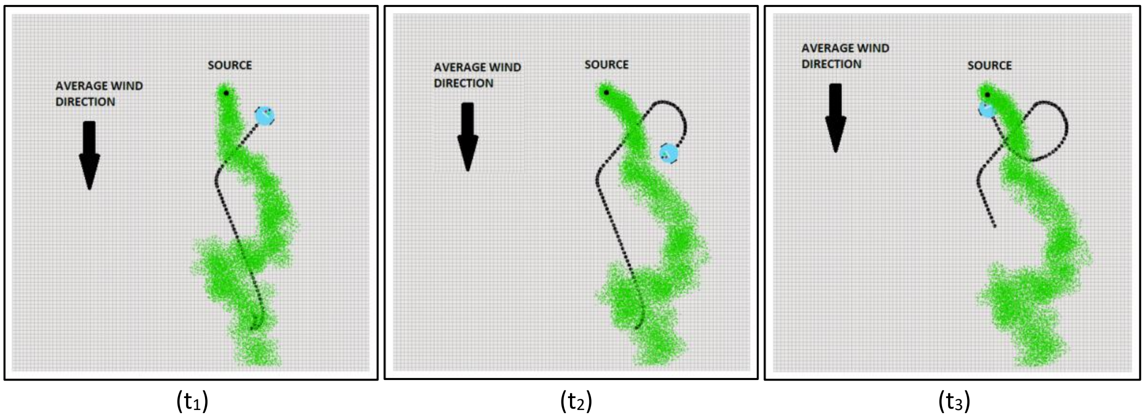


Figure 14. Execution of the Surge-Spiral algorithm. The robot moves upwind and maintains the same direction while it continues to measure gas. When it loses the gas, it performs a spiral to try and recover it. Each of the subfigures shows a different time instant.

4.1.3 Surge-Spiral Plume Tracking

The Surge-Spiral method, presented by Hayes *et al.* [40] is another version of the plume tracking strategy that combines the previous algorithms. In this particular case, the Spiral method is not used to move the robot towards the source, but merely to regain the plume once lost (Figure 14).

This method is more similar to Surge-Cast than it is to Spiral, letting go of the Proximity Index concept in favour of a gas concentration threshold, and returning to the idea of a state machine.

The implementation of this algorithm follows the specifications in [40]. Differing from the previous implementation of Surge-Cast, the surge is not interrupted as soon as the robot stops measuring gas and, during a surge, any successive hits will reset the surge distance without resampling the wind direction. This not only speeds up the process of moving within the plume (by getting rid of the wind resampling step), but also makes for a less strict tracking of the plume, meaning the robot is allowed to move outside of the plume more easily than in the case of the Surge-Cast algorithm.

This makes it easier for the robot to overcome points where the plume is bent or broken into patches due to turbulences, but in turn complicates the task of regaining the plume once lost, since the surge movement might end further away from it. Therefore, the spiralling movement is chosen to substitute the crosswind cast, since it is a more robust, albeit slower, method to regain the plume compared to casting[36].

Another difference from the previous versions of the involved algorithms is that the spirals no longer have fixed stop-and-measure points, but instead are interrupted whenever the measured gas concentration is above a given threshold.

Since the implementation made for this work is going to be used in environments that feature obstacles, the algorithm has been modified to also stop the surge movement and resample the wind direction if the current surge cannot continue due to the presence of obstacles in the path.

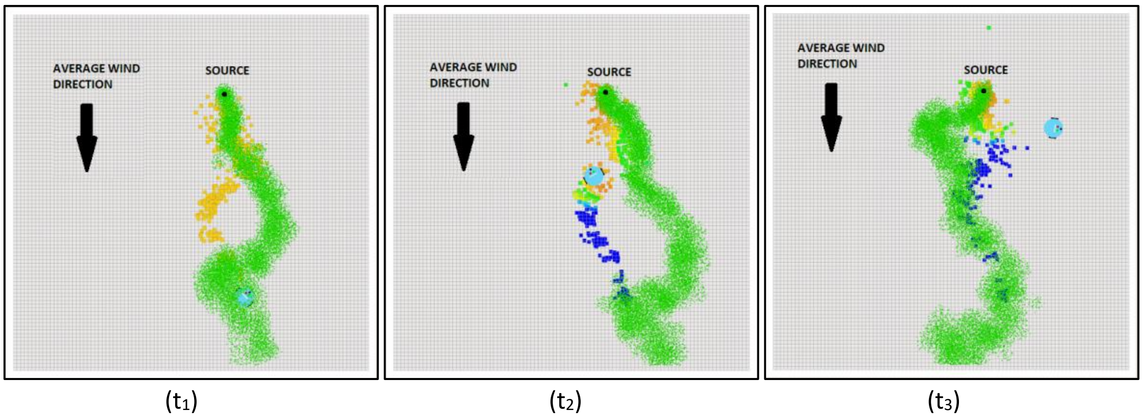


Figure 15. Example of the execution of the particle filter algorithm. The dots represent particles, and their colour represents their weight, red being the highest and blue the lowest. Each of the subfigures shows a different time instant.

4.1.4 Particle Filter Based GSL

This algorithm, presented by Li *et al.* [32] in 2011, is the only statistical inference method to be included in this work. It is based on the use of a particle filter to estimate the location of the source (Figure 15).

In order to sample and weigh the particles, a probability density function for the source location that depends on the gas and wind measurements is defined. As was explained previously, defining such a function for a complex environment with obstacles and turbulent wind is still an open question, and for that reason it is necessary to make some kind of simplification to the environment.

In this case, the PDF is calculated assuming that the wind can change over time, but is homogenous in all the environment. This assumption is reasonable for outdoors environments without obstacles, where air currents are large and can encompass the entire workspace of the search. This is the use the algorithm was devised for, but it will also be tested in more complex environments to study whether it performs well in conditions where the assumptions about the airflow do not hold.

As presented by the authors, the particle filter algorithm does not influence the movement of the robot. Instead, a reactive algorithm (Surge-Spiral, see Section 4.3) is utilized to guide the movements while the particle filter is executed.

In order to be able to make estimations, a record of the last wind observations is kept. When the first gas observation occurs, particles are generated inside the "Observation Window" (the area that the gas could have been released from, given it was measured in the current position), which is calculated using the history of wind to estimate the path followed by the gas patch (Figure 16). Since the implementation presented in this work is designed to be applicable to complex environments, it is also taken into account that particles do not fall inside of an obstacle.

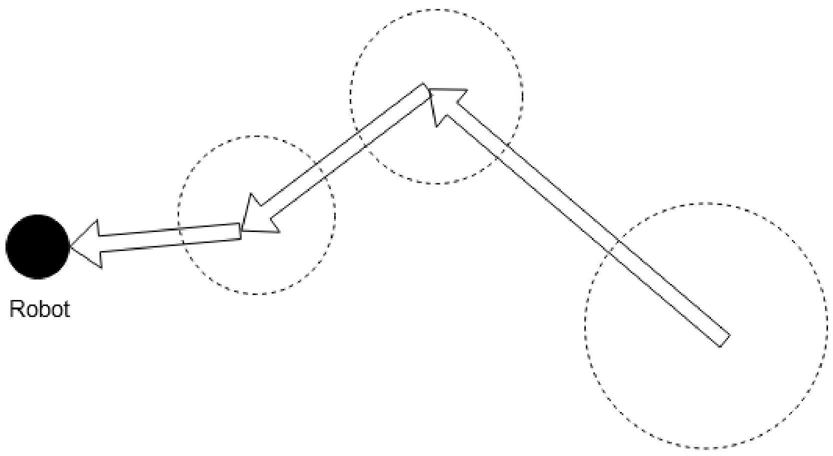


Figure 16. The arrows represent the path followed by the gas, as estimated from the measured wind vectors, and the dashed circles mark the Observation Window. (Adapted from [32])

From that point on, the weights of the particles are updated recursively with every new observation by multiplying the previous weight by the value of the PDF for the location of the particle and the new measurement.

Whenever the concentration of weights exceeds a given threshold a resampling step is performed, where particles with low weights are removed and new ones are generated around the particles that have higher weights.

As with any particle filter algorithm, mechanisms to deal with degeneracy of the distribution and to generate new particles when necessary are defined (read [32] for the details).

The algorithm declares the final estimation of the source position whenever a given number of consecutive estimations (calculated as the weighted average of the locations of the particles) converge. Due to change of weights being too small whenever a miss (non-detection of gas) occurs, only estimations that follow a hit are taken into account.

4.2 Implementation

All the algorithms included in this work have been implemented using pre-existing ROS tools for navigation, such as the *move_base* package, and the simulated sensor models provided by GADEN. Therefore, the implementations of the algorithms only include the logic of the algorithms themselves, and leave the low-abstraction path planning and sensing to the mentioned external tools.

The implementations have been developed using C++, and the processes that compose each of the algorithms have been defined as methods on a different class. Given that, as was mentioned, both the Surge-Cast and the Surge-Spiral algorithms are particular cases of the general Plume Tracking strategy, a non-instantiable Plume Tracking base class has been defined that provides the general state machine structure and common processes for Plume Tracking strategies, and the two implemented versions extend it.

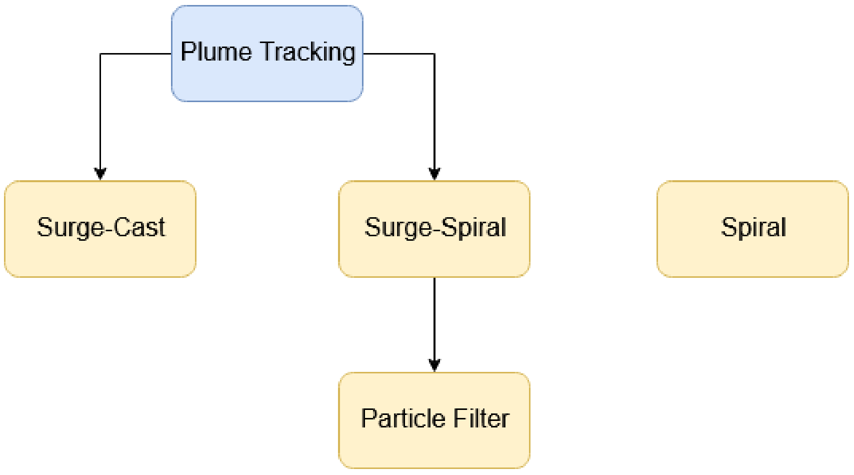


Figure 17. Class diagram of the implementation of the GSL algorithms.

The Particle Filter class, on the other hand, extends the Surge-Spiral class, since it retains all the movement functionality of the Surge-Spiral algorithm and adds the particle filter logic to it.

Despite including the spiral movement pattern, the Surge-Spiral and the Spiral classes are not directly related, since the abundant differences in all other aspects make little of the code re-usable. Figure 17 shows the resultant class structure.

To ease the use of these implementations, a central ROS node which functions as a server is provided. The server waits for a message containing the name of the algorithm, which is delivered through a specific ROS topic using the publisher-subscriber structure (See section 2.1). When the message is received, the server instantiates the corresponding class and starts the main loop of the selected algorithm.

Therefore, in order to use one of the implemented algorithms in a program of their one, one needs only to launch the server node and publish a message containing the name of the chosen algorithm.

5 Experiments

Using the implementations of the algorithms presented in the previous section, a series of experiments has been carried out in order to present a comprehensive comparative study of their performance in different environments.

The characteristics of these test environments have been chosen to highlight the strengths and weaknesses of the different methods, keeping in mind that each of them is designed for different specific environmental conditions.

It should be noted that, for the purposes of these experiments, the problem of source declaration (discerning when the source has been found) has not been taken into consideration. Therefore, the search will be considered a success whenever the robot gets within a set distance (in this case, 0.5 meters) of the source, or, for the case of the Particle Filter algorithm, whenever the average of the estimations is within said distance. The search will be considered a failure, on the other hand, if the source has not been found after a given time (set, for these experiments, to 600 seconds).

The problem of finding the plume in the first place has equally not been considered, since none of these methods define any particular exploration behaviours. Therefore, the robot will in all cases be initially positioned where it can measure gas.

The robot is equipped with a PID gas sensor and a 2D anemometer for taking the measurements necessary for the GSL algorithms and a laser scanner for navigation.

5.1 Weak Airflow in Empty Environment

This experiment is based on the conditions the Spiral algorithm was designed for [34]. In an environment without obstacles (10m x 10m), gas is released from a given point and allowed to distribute through the effect of a low speed, unstable airflow.

It is trivial that, in the absence of a measurable airflow, all methods other than Spiral will fail, since they require wind measurements to decide the direction of movements. Therefore, even though the wind speed is set to values low enough that sensor noise could pose a problem for anemotactic strategies, it has been kept high enough for them to function.

As can be observed in Figure 18, the gas disperses in the form of a plume in the average wind direction, but the low speed values and the instability of the airflow (modelled as gaussian noise in the movement of the filaments) make the plume grow wide and move slowly.

5.1.1 Parameters

All algorithms, except Spiral, use wind speed and gas concentration thresholds to determine whether a given measurement is too low to be used. Given the characteristics of the airflow, the threshold at which a wind measurement is considered to be valid has been set to 0.1 m/s, while the gas concentration threshold has been set to 0.3 ppm.

Another shared parameter is the length of the measurement step, which is used to get an average wind or gas value that is more reliable than an individual measurement. After some preliminary tests, the stop-and-measure phase has been set to be 5 seconds long for both plume tracking algorithms. Spiral uses a longer measurement phase, of 10

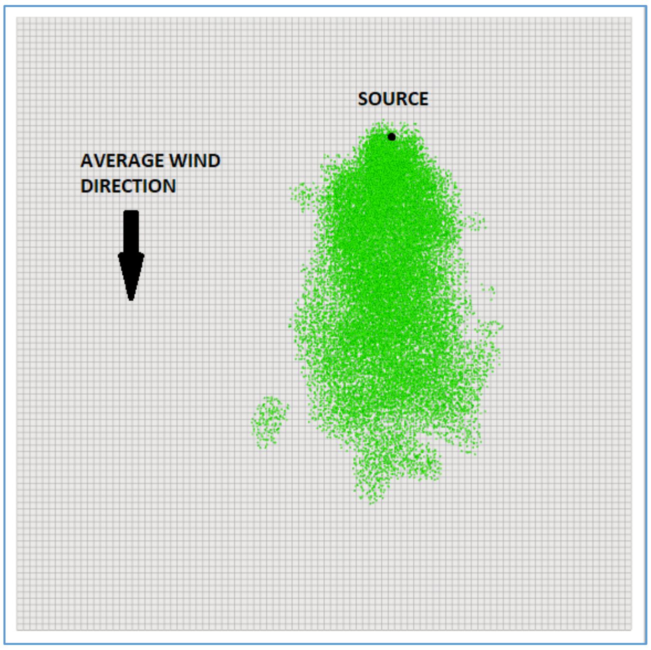


Figure 18. Gas plume formed in this experimental setup. The weak airflow chosen for this experiment causes gas to be dispersed as a wide, slow moving plume.

seconds divided into 2 seconds intervals (see [34]), since the preliminary tests showed that shorter measurement steps made the Proximity Index too unreliable.

For the Particle Filter algorithm, the last 30 seconds of wind measurements are taken into account each step (interval length of 1 second, see [32]), and the average of the last 10 estimations of the location of the source is the one used for the termination condition.

The robot starts at position (6.4, 1.8), and the gas source is in position (5.8, 7.7). The initial distance is 5.93 m, and the maximum speed at which the robot can move is 0.4 m/s.

5.1.2 Results

The experiment has been run five times for each of the algorithms, and the results are shown in Table 1. Figure 19 shows an example of the path followed by the robot for each of the algorithms.

As can be observed, all algorithms had a perfect success rate, with Surge-Cast having the best results. It is reasonable that plume tracking algorithms perform well in this case, since there is a clear, unbroken plume. Even though these algorithms rely on wind measurements to decide the direction of movement and the airflow is very unstable, the relatively long stop-and-measure time allows them to obtain the main wind direction through averaging despite the instantaneous instability.

Surge-Spiral is, on average, slower than Surge-Cast, despite their similarities, because maintaining the direction of the first surge rather than resampling makes it more likely to lose the plume, and not stopping the movement as soon as the plume is lost allows the robot to move further away from the gas, which forces it to spend more time in

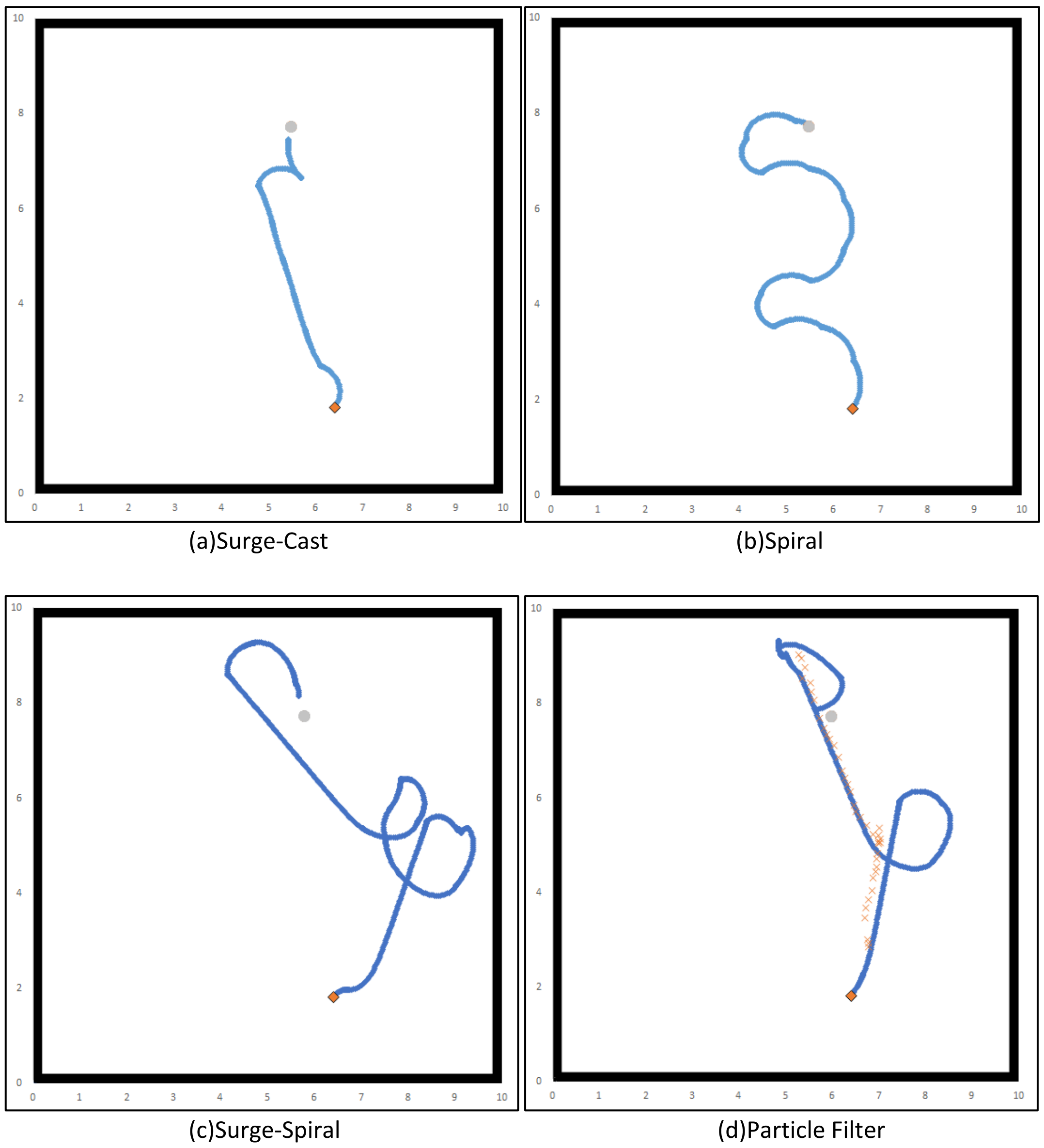
Table 1. Results for the four algorithms in an empty environment with weak, unstable airflow.

Algorithm	Success rate	Average time ± stdev(success)	Average distance travelled ± stdev (success)
Surge-Cast	5/5	82.60 ± 11.59 s	9.42 ± 2.60 m
Spiral	5/5	264.78 ± 151.63 s	20.28 ± 10.31 m
Surge-Spiral	5/5	128.52 ± 85.23 s	23.78 ± 13.76 m
Particle Filter	5/5	125.47 ± 44.81 s	23.09 ± 7.63 m

plume-recovery behaviours. Because of this, the "cost" of losing the plume is greater, which in turn increases the variance by making the runs in which the plume is lost several times much slower.

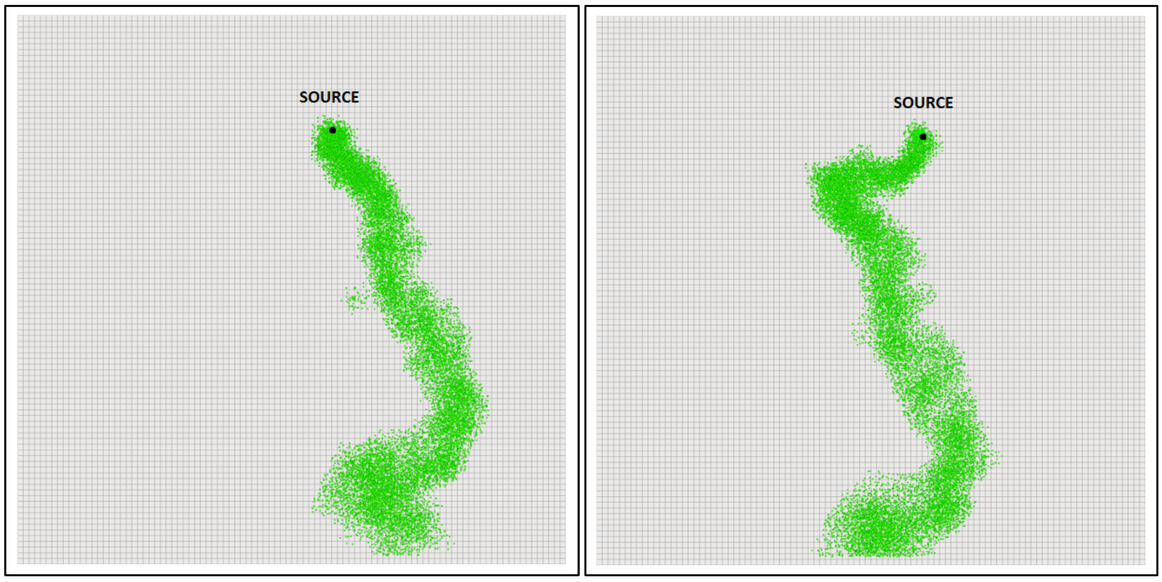
The particle filter algorithm, because of the low speed of the airflow, tends to make estimations near the current position of the robot, and therefore it can only effectively estimate the location of the source once the robot has found it through the Surge-Spiral algorithm. This problem is mitigated by the extensive wind record used for the predictions (30 seconds), but this is not enough to nullify it.

Spiral, despite making the robot travel a shorter distance than Surge-Spiral and the particle filter algorithm, is the slowest algorithm, because of the lengthy stop-and-measure step. As was mentioned before, this step needs to be longer than on other algorithms because the information gathered from the measured gas is more complex


(a)Surge-Cast (b)Spiral

(c)Surge-Spiral (d)Particle Filter

Figure 19. In blue, the path followed by the robot from the starting point (orange) to the gas source (grey) when the gas is being dispersed by a weak airflow. The crosses in (d) mark the estimations made in each iteration.



(t1) (t2)

Figure 20. Position and shape of the plume in two different time instants. As can be observed, a homogeneous airflow that changes over time causes the plume to bend and move from side to side.

(a Proximity Index, rather than a single gas concentration value) to make up for the lack of wind information.

5.2 Homogeneous Time-Dependent Airflow in Empty Environment

This experiment replicates the conditions described in [32] for the particle filter algorithm. The same empty 10m x 10m environment from the previous experiment is used, but with a strong airflow that is homogenous in the entire workspace and changes over time. This is designed to be an admissible simplification of the way the wind affects the gas source
localization process in a completely open outdoors environment.

As can be observed in Figure 20, the characteristics of this airflow cause the gas to disperse in the form of a narrow, bent plume that gradually changes shape and position over time.

5.2.1 Parameters

For this experiment, the gas concentration threshold to determine when a hit occurs has been kept at 0.3 ppm, unchanged from the previous experiment. The wind speed threshold, on the other hand, has been increased to 0.3 m/s, since the stronger airflow eliminates the necessity to accept unclear measurements.

The duration of the stop-and-measure step has also been adjusted, reducing it to 1 second for Surge-Cast and Surge-Spiral. This change is to account for the fact that, since the plume is constantly changing shape, and Surge-Cast only allows the surge movement to continue while the robot is sensing gas, a long measurement phase can cause the robot to lose the plume before it begins moving, preventing movement altogether.

For Spiral, the measurement phase has been kept 10 seconds long, since having fixed measurement points rather than stopping upon coming into contact with gas creates a higher variance in the reliability of the measurements, and a long observation phase helps compensate for it by gathering a larger number of measurements.

The particle filter algorithm takes into account the last 15 seconds of wind measurements (0.5s intervals, see [32]) since the higher wind speed means the measured particles are likely to cover the distance that separates the source from the robot faster than in Experiment 1. Again, the average of the 10 most recent observations is used for checking the termination condition.

The robot starts at position (9, 4.5), and the gas source is in position (0.5, 3.0). The maximum speed at which the robot can move is 0.4 m/s.

5.2.2 Results

The experiment has been run five times for each of the algorithms, and the results are shown in Table 2. Figure 21 shows an example of the followed path for each of the algorithms.

As can be observed, Surge-Spiral presents the best results, both in terms of average performance and in terms of reliability. The better results compared to Surge-Cast can be explained by considering the fact that a bending plume will cause the robot to lose ang regain contact with the plume often, and, whenever that happens, Surge-Cast stops the robot for a new measurement phase, while Surge-Spiral continues moving,

Table 2. Results of the algorithms for an empty environment with a weak airflow.

Algorithm	Success rate	Average time ± stdev (success)	Average distance travelled ± stdev (success)
Surge-Cast	5/5	114.40 ± 43.17 s	18.19 ±7.75 m
Spiral	3/5	203.90 ± 34.65 s	14.91 ± 3.64 m
Surge-Spiral	5/5	50.50 ± 7.21 s	8.87 ± 1.33 m
Particle Filter	5/5	75.42 ± 49.95 s	13.75 ± 8.30 m

completing the current upwind movement and potentially extending the surge with a new gas hit.

The particle filter algorithm also presents good results, although the variance is much larger than that of Surge-Spiral. This is mostly because using the average of several estimations to check the termination condition makes it so that any significant error in one of the estimations drags the average away from the source, slowing down the process.

Spiral shows the worst results, being the only algorithm that did not manage to find the source on every occasion. Given that Spiral is a purely chemotactic strategy designed to be used in diffusion-dominated environments, it is not unexpected for it to have a worse performance than anemotactic strategies in an environment that features a strong airflow.

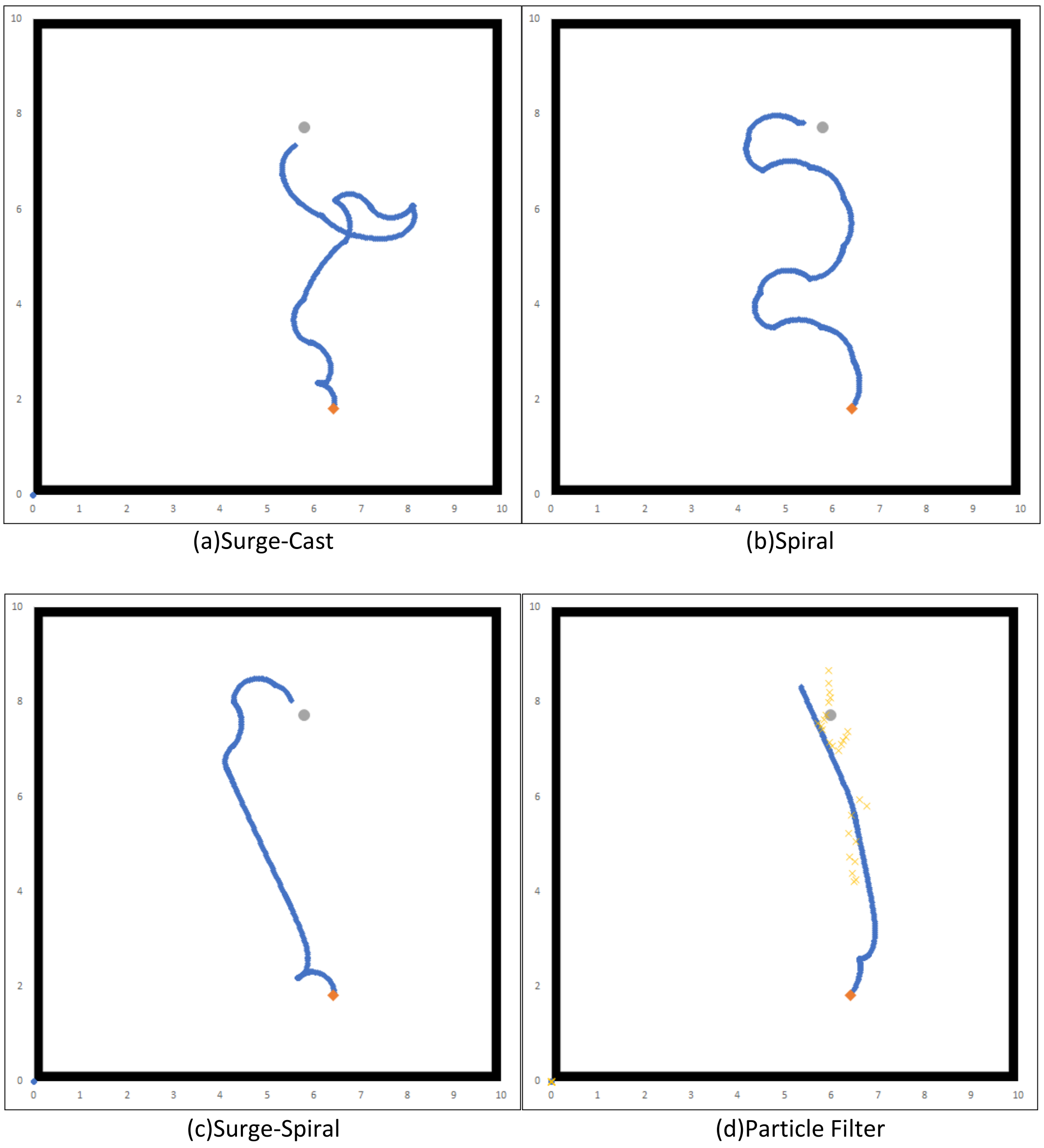


Figure 21. In blue, the path followed by the robot from the starting point (orange) to the gas source (black) when the gas is being dispersed by a homogenous airflow. The crosses in (d) mark the estimations made in each iteration.

5.3 Steady Airflow With a Central Obstacle

This experiment introduces the presence of an obstacle in the environment. As can be observed in Figure 22, the disruption of the airflow caused by the obstacle makes for a more complex problem where the plume is broken and moves close to the walls. This poses a problem for all chosen algorithms: the turbulences caused by the presence of the obstacle can cause anemotactic strategies to choose the wrong direction to move, and the fact that gas is only present in the close vicinity of obstacles greatly disrupts the movement pattern that is the main component of the Spiral strategy.

5.3.1 Parameters

Most parameters are kept to values similar to those in Experiment 2. Both gas concentration and wind speed thresholds remain unchanged at 0.3 ppm and 0.3 m/s respectively.

The duration of the stop-and-measure phase has been increased from 1 second to 1.5 for the plume-tracking algorithms, since the position of the gas plume is more stable than in the previous experiment (eliminating the problems caused by a long measurement phase), and averaging a larger number of measurements reduces the impact of sensor noise. Spiral still uses a long measurement phase (10 seconds) to maximize the reliability of the Proximity Index.

The particle filter algorithm remains unchanged as well, with a wind measurement record of 15 seconds divided into 0.5 seconds intervals, and using the average of the last 10 source position estimations to check the termination condition.

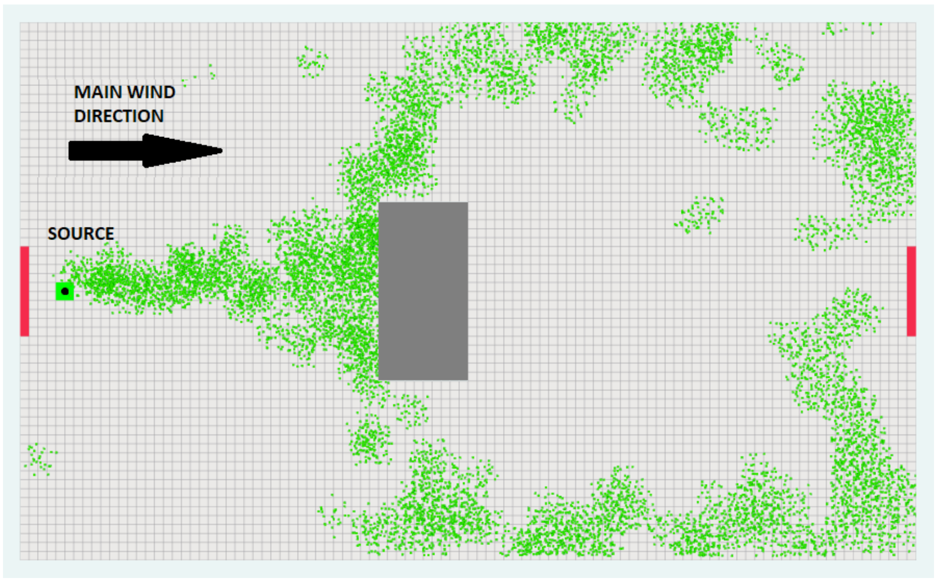


Figure 22. Environment chosen for Experiment 3. The presence of obstacles in the environment disrupts the formation of a plume, creating a more complex dispersion pattern, as well as posing a difficulty for robot navigation. The red rectangles indicate the areas where gas can enter or leave the environment.

The robot starts at position (9.2, 2.1), and the gas source is in position (0.5, 3.0). The maximum speed at which the robot can move is 0.4 m/s.

5.3.2 Results

The experiment has been run five times for each of the algorithms, and the results are shown in Table 3. Figure 23 shows an example of the followed path for each of the algorithms.

As can be observed, the introduction of an obstacle does not necessarily hinder the ability of plume-tracking algorithms to find the source. Surge-Cast shows the best results, managing to track the broken plume around the obstacle and find the main body of the plume.

Surge-Spiral has similar results, but is on average slightly slower. As was observed in Experiment 1, the lack of mechanisms to interrupt the surge means that the robot is allowed to more freely leave the plume, forcing it to spend more time in recovering the gas.

The particle filter algorithm shows similar results to those of Surge-Spiral, but is on average slower. Given that the assumption about homogenous wind is not met, the estimations made during the first part of the search are often meters away from the source, making it so that the average of the last 10 estimations only coincides with the location of the source after the robot has physically reached it.

Spiral, in this case, is unable to find the source. Being guided only by the gas measurements in an environment where advection is the phenomenon that dominates the dispersion of gas makes it so that the Proximity Index is too unreliable, and having

Table 3. Results for the four algorithms in an environment that features a central obstacle and a steady airflow.

Algorithm	Success rate	Average time ± stdev (success)	Average distance travelled ± stdev (success)
Surge-Cast	5/5	77.76 ± 17.65 s	13.93 ± 3.30 m
Spiral	0/5	-	-
Surge-Spiral	5/5	83.88 ± 21.15 s	18.73 ± 4.74 m
Particle Filter	5/5	112.52 ± 5.23 s	17.51 ± 1.27 m

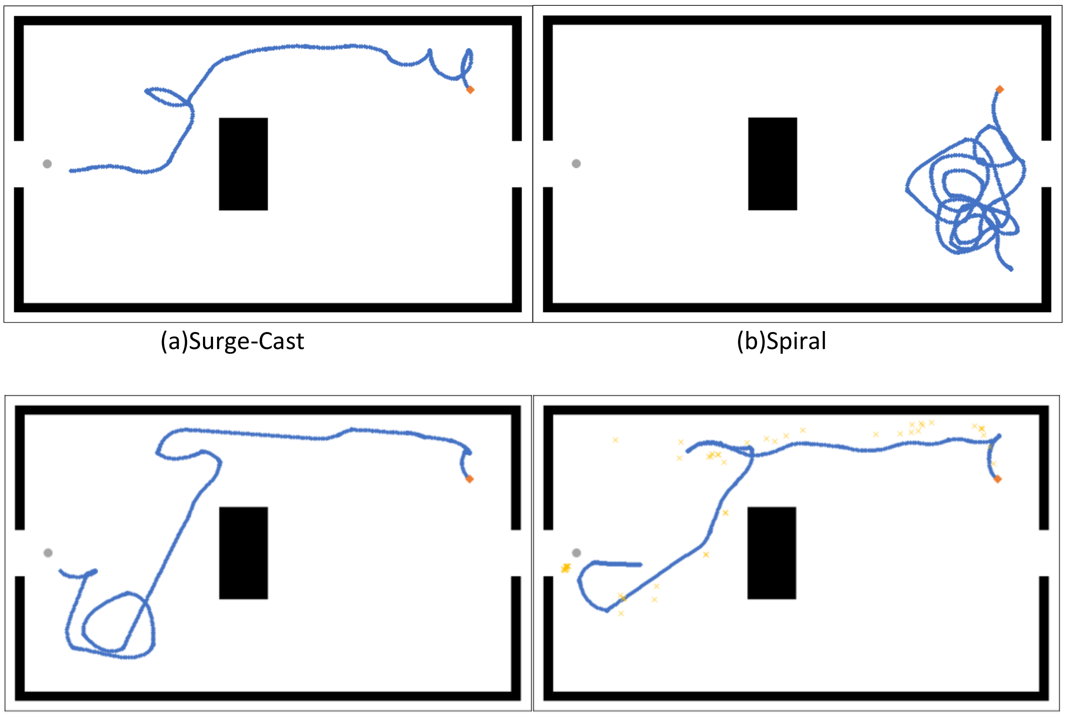


Figure 23. In blue, the path followed by the robot from the starting point (orange) to the gas source (grey). The crosses in (d) mark the estimations made in each iteration.

the plume be narrow and close to walls makes it difficult for the fixed measurement points to coincide with the interior of the plume.

5.4 Steady Airflow in Maze-Like Environment

This experiment is set in a more complex environment that resembles a maze (Figure 24), with several interior walls forming a narrow zig-zagging path from the gas source to the initial location of the robot.

The shape of this environment makes it difficult for the robot to move outside of the gas plume (the corners being the only points where sometimes there is not enough gas for a hit to occur), but the direction changes and blocked areas can pose a problem for this algorithms, since they do not use any map information to set the navigation goals and instead rely on relatively simple movement patterns.

5.4.1 Parameters

This experiment has been carried out using the same values used for Experiment 3 for all parameters: the gas concentration threshold is 0.3 ppm, the wind speed threshold is 0.3 m/s; the stop-and measure duration is set to 1 s for plume-tracking algorithms and to 10 s for Spiral; and the particle filter algorithm used 15 s of wind measurements divided into 0.5 s intervals to make estimations, with the 10 most recent estimations being considered to check the termination condition.

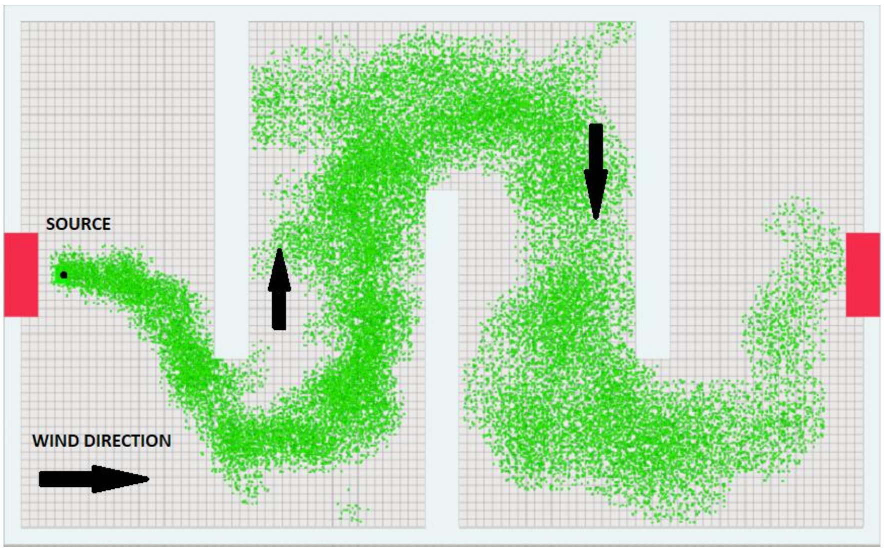


Figure 24. Environment chosen for Experiment 3. The presence of obstacles in the environment disrupts the formation of a plume, creating a more complex dispersion pattern, as well as posing a difficulty for robot navigation.

5.4.2 Results

The experiment has been run five times for each of the algorithms, and the results are shown in Table 4. Figure 25 shows an example of the followed path for each of the algorithms.

As can be observed, all the algorithms show a very small variance in travelled distance. This can be explained by the shape of the environment, which forms a narrow path that does not allow the robot to move outside of the plume.

Despite this, the variance in the time taken to traverse said distance is much larger. This is due to navigation issues, since the robot tries to move according to the predefined patterns and often ends up moving too close to walls, forcing the navigation tools (*move_base*, see Section 2.1) to perform recovery behaviours in order to find a path to the next goal that does not risk a collision. These behaviours are slow, and therefore have a significant impact on the final results.

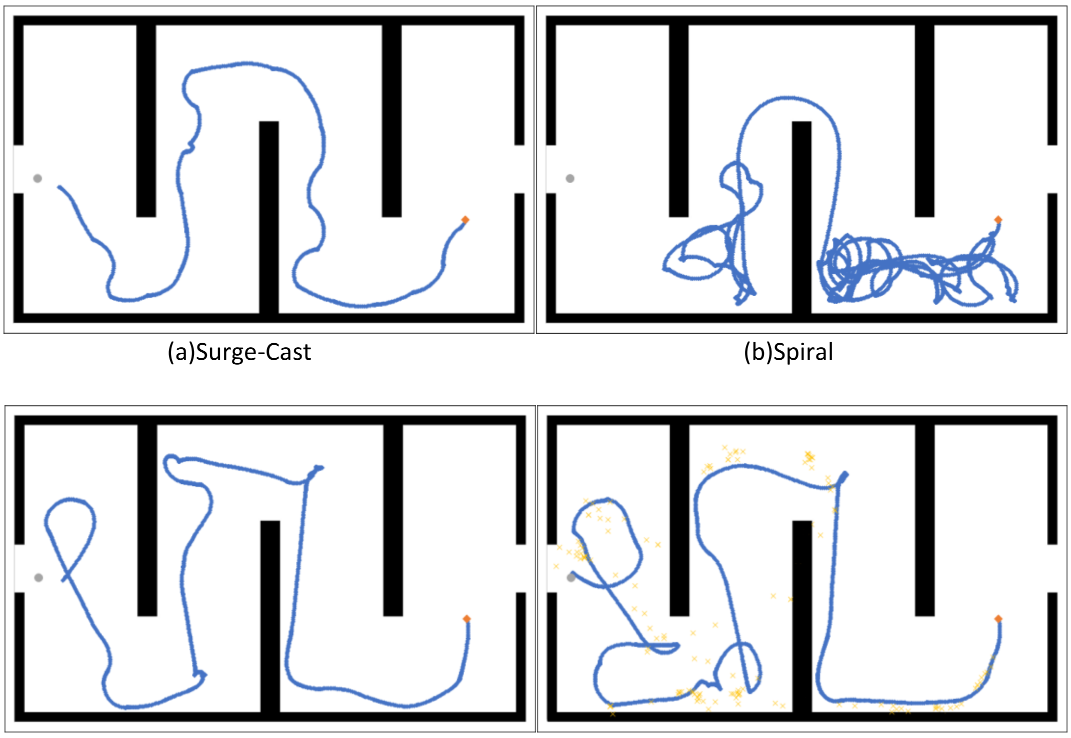
Analysing the results of each algorithm we can observe that Surge-Cast and Surge-Spiral have comparable performances, with Surge-Cast having a better average time but higher variance.

The particle filter algorithm has a significant time overhead compared to Surge-Spiral for similar reasons to the last experiment, that is, the estimations made while traversing the environment are very imprecise due to the assumptions about the airflow not being applicable to this environment, and therefore the algorithms is only able to produce precise estimations of the location of the source once the robot has reached its vicinity.

Spiral is unable to find the gas source in this environment, due to the difficulty to navigate the narrow path with spiralling movements and the unreliability of the Proximity Index in such an advection-dominated environment.

Table 4. Results of the four algorithms in a maze-like environment with a steady airflow.

Algorithm	Success rate	Average time ± stdev (success)	Average distance travelled ± stdev (success)
Surge-Cast	5/5	182.42 ± 55.64 s	21.45 ± 1.74 m
Spiral	0/5	-	-
Surge-Spiral	5/5	194.98 ± 23.52 s	26.59 ± 3.52 m
Particle Filter	5/5	229.30 ± 27.65 s	28.55 ± 2.39 m



(a)Surge-Cast (b)Spiral

(c)Surge-Spiral (d)Particle Filter

Figure 25. In blue, the path followed by the robot from the starting point (orange) to the gas source (grey). The crosses in (d) mark the estimations made in each iteration.

6

Conclusions and Future Work

This work has presented a comparative experimental study of some of the most commonly used Gas Source Localization methods. Using the gas dispersion simulator GADEN, and the navigation and sensing tools offered by ROS, four chosen algorithms (Surge-Cast [38], Spiral [34], Surge-Spiral [40] and the Particle Filter based GSL presented by Li *et al.* [32]) have been implemented and then tested in different environmental conditions, ranging from open spaces with homogenous airflows to more complex environments featuring obstacles and turbulent wind. From the results of these simulated experiments, it can be concluded that:

- **Surge-Cast Plume Tracking** is a versatile algorithm that shows good results in all the experiments that have been carried out. This algorithm works best when a clear, stable plume exists, but has shown to be able to handle non-ideal situations as well. Results show that in the absence of an ideal plume it is particularly relevant that the parameters of the algorithm (most notably, the duration of the measurement phase) are adjusted for the specific conditions of the environment.

  It should be noted that all the chosen environments, even the ones that feature turbulences, allow some semblance of a plume to be formed. Being a plume-tracking algorithm, it is still to be tested what the performance of Surge-Cast would be in case of a more turbulent airflow that does not allow for the formation of any plume.

- **The Spiral algorithm** is the only one of the implemented algorithms that can be used in environments without a measurable airflow, since it does not require any wind information. The algorithm has been shown to be able to reliably find the gas source in diffusion-dominated environments, and to less reliably find the source in open environments with strong changing airflows.

  It must also be concluded from the results that Spiral is not the appropriate choice for more complex environments with obstacles, and that, in the presence of a strong airflow, it is less efficient than anemotactic strategies.

- **Surge-Spiral Plume Tracking** is similar to Surge-Cast, and has managed to reliably find the source in every environment. Because of its less strict policy with respect to staying inside of the gas plume, it is able to out-perform Surge-Cast in environments where the plume is broken into patches or displaced by a strong wind. On the other hand, it is on average slower in environments with a stable plume because leaving the plume more easily forces it to spend more time trying to regain it.

- **The Particle Filter Algorithm** shows good results when the specific conditions for which it was designed (that is, a near homogeneous wind) are met, but does not have a good performance in more complex environments, only being able to give a correct estimation of the source position after the robot physically reaches it through the Surge-Spiral movement strategy. It must be concluded, then, that it is only appropriate for use in outdoors environments where the assumption of a homogeneous wind is admissible.

  It should be pointed out that he particle filter algorithm is the only of these four that offers a way to perform source declaration, and it remains to be tested whether that specific application might perform well in complex environments.

From the previous conclusions and the ideas discussed in this work, some proposals for future investigation can be drawn, including:

- Performing real-world experiments with the implemented algorithms to verify the simulated results with actual robotic hardware and real gas dispersion data.

- Using the ROS and GADEN tools to implement a larger number of GSL algorithms, creating an open-source repository that would allow for more extensive testing, as well as easy deployment on real robots.

- Extending the scope of the experiments to consider the plume-finding and source declaration subproblems. In particular, testing the applicability of the particle filter algorithm as a source declaration method even in complex environments.

- Developing new GSL methods that use the information about the shape of the environment to make predictions and plan the navigation, either with prior knowledge of the map or combining the GSL methods with online mapping techniques.

References

[1] G. Kowadlo and R. A. Russell, “Robot odor localization: A taxonomy and survey,” *Int. J. Rob. Res.*, vol. 27, no. 8, Aug. 2008.

[2] D. Martínez *et al.*, “A mobile robot agent for gas leak source detection,” in *Advances in Intelligent Systems and Computing*, vol. 293, Springer, Cham, 2014.

[3] J. Burgués and S. M. Colás, “Signal Processing and Machine Learning for Gas Sensors: Gas Source Localization with a Nano-Drone,” Universitat de Barcelona, 2019.

[4] J. P. Trevelyan, S.-C. Kang, and W. R. Hamel, “Robotics in Hazardous Applications,” in *Springer Handbook of Robotics*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[5] M. Reggente *et al.*, “The DustBot System: Using Mobile Robots to Monitor Pollution in Pedestrian Area,” *Chem. Eng. Trans.*, vol. 23, 2010.

[6] P. Schneider, N. Castell, M. Vogt, F. R. Dauge, W. A. Lahoz, and A. Bartonova, “Mapping urban air quality in near real-time using observations from low-cost sensors and model information,” *Environ. Int.*, vol. 106, no. May, 2017.

[7] W. Tsujita, A. Yoshino, H. Ishida, and T. Moriizumi, “Gas sensor network for air-pollution monitoring,” *Sensors Actuators, B Chem.*, vol. 110, no. 2, 2005.

[8] D. M. Wilson, S. Hoyt, J. Janata, K. Booksh, and L. Obando, “Chemical sensors for portable, handheld field instruments,” *IEEE Sens. J.*, vol. 1, no. 4, 2001.

[9] C. W. Gardner, R. Wentworth, P. J. Treado, P. Batavia, and G. Gilbert, “Remote chemical biological and explosive agent detection using a robot-based Raman detector,” in *Unmanned Systems Technology X*, 2008, vol. 6962.

[10] C. Humphrey and J. A. ADAMS, “Robotic tasks for CBRNE incident response,” *Adv. Robot.*, 2009.

[11] J. Monroy, J. R. Ruiz-Sarmiento, F.-A. Moreno, C. Galindo, and J. Gonzalez-Jimenez, “Towards a Semantic Gas Source Localization under Uncertainty,” in *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Applications*, Springer, Cham, 2018.

[12] J. W. G. and P. N. Bartlett, “Electronic Noses. Principles and Applications,” *Meas. Sci. Technol.*, vol. 11, no. 7, Jul. 2000.

[13] J. Gonzalez-Jimenez, J. G. Monroy, and J. L. Blanco, “The multi-chamber electronic nose-an improved olfaction sensor for mobile robotics,” *Sensors*, vol. 11, no. 6, Jun. 2011.

[14] I. Moreno, R. Caballero, R. Galan, F. Matia, and A. Jimenez, “Electronic nose: State of art,” *RIAI - Rev. Iberoam. Autom. e Inform. Ind.*, vol. 6, 2009.

[15] A. Gongora, J. Monroy, and J. Gonzalez-Jimenez, “An Electronic Architecture for Multipurpose Artificial Noses,” *J. Sensors*, vol. 2018, Feb. 2018.

[16] J. Monroy and J. Gonzalez-Jimenez, “Odor Classification in Motion: How Fast Can the E-nose Go?,” *16th Int. Symp. Olfaction Electron. Nose*, 2015.

[17] J. G. Monroy, E. J. Palomo, E. López-Rubio, and J. Gonzalez-Jimenez, “Continuous chemical classification in uncontrolled environments with sliding windows,” *Chemom. Intell. Lab. Syst.*, vol. 158, no. 158, Nov. 2016.

[18] J. G. Monroy, J. L. Blanco, and J. Gonzalez-Jimenez, “Time-variant gas distribution mapping with obstacle information,” *Auton. Robots*, vol. 40, no. 1, Jan. 2016.

[19] M. Reggente and A. J. Lilienthal, “Using local wind information for gas distribution mapping in outdoor environments with a mobile robot,” in *Proceedings of IEEE Sensors*, 2009.

[20] J. Monroy, J. R. Ruiz-Sarmiento, F. A. Moreno, F. Melendez-Fernandez, C. Galindo, and J. Gonzalez-Jimenez, “A semantic-based gas source localization with a mobile robot combining vision and chemical sensing,” *Sensors (Switzerland)*, vol. 18, no. 12, 2018.

[21] A. Liberzon, K. Harrington, N. Daniel, R. Gurka, A. Harari, and G. Zilman, “Moth-inspired navigation algorithm in a turbulent odor plume from a pulsating source,” *PLoS One*, vol. 13, no. 6, Jun. 2018.

[22] M. Hutchinson, C. Liu, and W. H. Chen, “Information-Based Search for an Atmospheric Release Using a Mobile Robot: Algorithm and Experiments,” *IEEE Trans. Control Syst. Technol.*, 2018.

[23] J. Monroy and J. Gonzalez-Jimenez, “Towards odor-sensitive mobile robots,” in *Electronic Nose Technologies and Advances in Machine Olfaction*, 2018.

[24] S. Sklavounos and F. Rigas, “Validation of turbulence models in heavy gas dispersion over obstacles,” *J. Hazard. Mater.*, vol. 108, no. 1–2, Apr. 2004.

[25] C. W. Miller and L. M. Hively, “A review of validation studies for the Gaussian plume atmospheric dispersion model,” *Nucl. Saf.; (United States)*, 1987.

[26] J. Monroy, V. Hernandez-Bennetts, H. Fan, A. Lilienthal, and J. Gonzalez-Jimenez, “GADEN: A 3D gas dispersion simulator for mobile robot olfaction in realistic environments,” *Sensors (Switzerland)*, vol. 17, no. 7, 2017.

[27] M. Quigley *et al.*, “ROS: an open-source Robot Operating System,” *ICRA Work. open source Softw.*, no. 3.2, 2009.

[28] “Fundamentals of ROS - ROS Robotics Projects.” [Online]. Available: https://subscription.packtpub.com/book/hardware_and_creative/9781783554713/1/ch01lvl1sec8/fundamentals-of-ros. [Accessed: 07-Sep-2019].

[29] B. P. Gerkey, R. Vaughan, and A. Howard, “The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems,” *Proc. Int. Conf. Adv. Robot.*, 2003.

[30] J. A. Farrell, J. Murlis, X. Long, W. Li, and R. T. Cardé, “Filament-based atmospheric dispersion model to achieve short time-scale structure of odor plumes,” in *Environmental Fluid Mechanics*, vol. 2, no. 1–2, 2002.

[31] B. Zhao, C. Yang, X. Yang, and S. Liu, “Particle dispersion and deposition in ventilated rooms: Testing and evaluation of different Eulerian and Lagrangian models,” *Build. Environ.*, vol. 43, no. 4, Apr. 2008.

[32] J. G. Li, Q. H. Meng, Y. Wang, and M. Zeng, “Odor source localization using a mobile robot in outdoor airflow environments with a particle filter algorithm,” *Auton. Robots*, vol. 30, no. 3, 2011.

[33] C. Sanchez-Garrido, J. Monroy, and J. Gonzalez-Jimenez, “Probabilistic estimation of the gas source location in indoor environments by combining gas and wind observations,” *Front. Artif. Intell. Appl.*, vol. 310, 2018.

[34] G. Ferri, E. Caselli, V. Mattoli, A. Mondini, B. Mazzolai, and P. Dario, “SPIRAL: A novel biologically-inspired algorithm for gas/odor source localization in an indoor environment with no strong airflow,” *Rob. Auton. Syst.*, vol. 57, no. 4, Apr. 2009.

[35] J. Macedo, L. Marques, and E. Costa, “A Comparative Study of Bio-Inspired Odour Source Localisation Strategies from the State-Action Perspective,” *Sensors*, vol. 19, no. 10, 2019.
[36] T. Lochmatter and A. Martinoli, “Theoretical analysis of three bio-inspired plume tracking algorithms,” *Proc. - IEEE Int. Conf. Robot. Autom.*, 2009.
[37] R. A. Russell, A. Bab-Hadiashar, R. L. Shepherd, and G. G. Wallace, “A comparison of reactive robot chemotaxis algorithms,” *Rob. Auton. Syst.*, vol. 45, no. 2, Nov. 2003.
[38] H. Ishida, T. Nakamoto, T. Moriizumi, T. Kikas, and J. Janata, “Plume-Tracking Robots: A New Application of Chemical Sensors,” *Biol. Bull.*, vol. 200, no. 2, 2001.
[39] H. Ishida, K. Suetsugu, T. Nakamoto, and T. Moriizumi, “Study of autonomous mobile sensing system for localization of odor source using gas sensors and anemometric sensors,” *Sensors Actuators A. Phys.*, vol. 45, no. 2, Nov. 1994.
[40] A. T. Hayes, A. Martinoli, and R. M. Goodman, “Distributed odor source localization,” *IEEE Sens. J.*, vol. 2, no. 3, 2002.
[41] V. H. Bennetts, A. J. Lilienthal, P. P. Neumann, and M. Trincavelli, “Mobile robots for localizing gas emission sources on landfill sites: Is bio-inspiration the way to go?,” *Front. Neuroeng.*, vol. 4, no. JANUARY, 2012.
[42] Y. Wada, M. Trincavelli, Y. Fukazawa, and H. 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*, vol. 2010.5, no. 0, 2010.
[43] G. Cabrita, P. Sousa, and L. Marques, “Player/Stage simulation of olfactory experiments,” in *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, 2010.
[44] J. Monroy, J.-L. Blanco, and J. Gonzalez-Jimenez, “An Open Source Framework for Simulating Mobile Robotics Olfaction,” in *15th International Symposium on Olfaction and Electronic Nose (ISOEN)*, 2013.
[45] J. Monroy, J. Gonzalez-Jimenez, J.-R. Ruiz-Sarmiento, and J. Gonzalez-Jimenez, “An Evaluation of Plume Tracking as a Strategy for Gas Source Localization in Turbulent Wind Flows,” *2019 IEEE Int. Symp. Olfaction Electron. Nose*, May 2019.