Department of Mechanical Engineering
King's College London

MSc Project Report (7CEMM721)

Mine sniffer robot

by

Boubacar BARRY

Supervised by **Chris Lorenz**

9 September 2008

DECLARATION

This report is submitted to the Department of Mechanical Engineering (King's College London) as part of course 7CEMM721 and is solely the original work of its author, except where clearly specified otherwise. It has not been previously submitted to this or another university.

Student: Boubacar BARRY Signature: Date: 9/9/2008

Supervisor: Dr Chris Lorenz

2nd Marker: Dr Kaspar Althoefer

Abstract

Landmines cause every year numerous deaths and injuries. The scent of their explosives is the most obvious clue about their presence.

The odour leaked from the source is carried by the wind flow, providing only intermittent contact with the chemical plume. The mine sniffer robot has be adapted to main purpose of the scent following task also be able to deal with obstacles without compromising on the primary task.

Most plume tracing strategies are based on insects, or micro-organisms trail following behaviour. KIKS (Robot simulation software) was used to assess the performance of the E.Coli-Bug1 chemotaxis based algorithm and the Moth-Tangent anemotaxis based algorithm.

The E.Coli algorithm confronted to a time averaged plume produces slow and unsatisfying outcome to the simulation, even though obstacles are handled very well and favoured the search : the source can only be estimated with visual interpretation.

The much more satisfying Moth algorithm produces realistic results, when submitted to wind simulation. If weather conditions are stable, a good mine location estimation is produced. However, when the wind is set to turbulent, returned landmine position is often wide off the mark.

However, many important parameters, such as the landmine actual chemicals release rate, and the sensor characteristics have been omitted during the different simulation runs, making the outcome doubtful. Should those issues been addressed, the simulator could prove to be useful in developing an efficient mine sniffer robot, or any other chemical trail follower, provided that a sensor exits for the substance of interest .

Acknowledgments

I would like to thank Dr Chris Lorenz and Dr Kaspar Altoefer for their positive criticism an guidance, which were invaluable for the purpose of the project

I also would like to express my gratitude to my colleagues Arkapravo Bhaumik, Vaseleios Lahanas and for their help and support during this long and difficult year.

Table of contents

1/ Introduction……1

2/ Chemical plume characteristics……....3
2.1/ Instantaneous plume…….….4
2.2/ Time averaged plume……4
2.2.1/ Gaussian Plume Model……...6
2.2.2/ Carrying fluid flow statistical characteristics……6

3/ Behaviour based path planning……7
3.1/ Robot motion planning ……7
3.2/ Behaviours definition ……....8

4/ Literature review……..….10
4.1/ Ideal path response ……..…10
4.1.1/ Ideal dour dispersion model …….……10
4.1.2/ Ideal path response to reach the goal……...11
4.2/ Biomimetic strategies ……...…15
4.2.1/ Flying insects inspired strategies ……...…….…15
4.2.2/ Lobster/Bacterium inspired strategies ……23

5/ Simulation procedure……...…….28
5.1/ Robot simulation: KIKS…….….28
5.2/ Robot environnent ……..….29
5.3/ Odour flow simulation……..…30

6/ Highest gradient following Algorithm…….….32

7/ Chemotaxis Algorithm……..….37
7.1/ Signal model (Time averaged Gaussian plume)……...37
7.2/ Obstacle handling……...…..37
7.3/ Algorithm……..…38
7.5/ Results……..….40
7.6/ Discussion……...…..44

8/ Anemotaxis Algorithm……..46
8.1/ Signal modelling (Statistical model)……..…..46
8.2/ Obstacle handling…….…48
8.3/ Algorithm…….…49
8.4/ Results……..……52
8.5/ Discussion ……...58

9/ Conclusion……..60
9.1/ Successes achieved during the project…….60
9.2/ Problems encountered……..60
9.3/ Future recommendations……..61
9.4/ Closing comments……62

10/ References……63
11/Appendix……63

1/ Introduction

A land mine is an explosive device designed to be placed underground or underwater to explode when triggered by the proximity of a vehicle or a person.

For obvious reasons, anti-personnel mines are designed to be difficult to find: they are usually of a small size, and are made almost entirely of non metallic material to avoid detection. They cause every year numerous deaths and injuries for people and deminers because in most cases, no record were kept of their location. Clearing minefields involves finding and disposal of the mines. Metal detectors are usually inefficient, as low quantities of metallic material are used in the device, and because metallic debris are quite common on a battlefield. However, all landmines leak with time small amounts of explosives into nearby soil or water that can be found on surrounding ground and plant life.

Most techniques of mine detection aim at localizing the source of those leaked chemicals. The current technique employs animals such as sniffer dogs, or trained mongooses, capable of detecting a land mine thanks to their strong sense of smell. They are by far the most effective techniques for clearing mine fields, and also the most expensive. Both living beings, they are limited to a couple of hours work a day, that can be further shortened by weather or health conditions. Training animals and peoples for this risky task is time consuming, and mistakes in the demining process cannot be avoided, leaving behind a couple of mines in the process. It costs only a few dollars to lay a landmine, it can cost many hundreds of dollars to find and remove it using the current method.

To lower the cost and speed up the mine clearing process, novels methods are being developed, all looking for the physical changes that could signal the presence of a mine.

Ground penetrating radars[1] detects all buried objects susceptible to be a mine.

Thermal imaging[2] focuses on the difference in heat dissipation between the land mine and the neighbouring soil.

Hybrid techniques involving the use of animals and robots are also being developed, such as the Mongoose-robot association.

However, the most promising method involves autonomous robots able to track the chemical signal up to its source, in short an automated version of the dog and deminer association . Many obstacles stand in the way of an autonomous vehicle fulfilling this purpose.

The chemicals leaked from the mine are carried by the fluid flow, in this case ambient air, and dispersed by random turbulences, creating an odour plume. Finding the source will be the prime function of the mobile robot. It involves means of sensing the vapours emitted by the mine, an ability to navigate towards it, and the capacity to handle obstacles and changes in the wind direction.

Mother nature is providing numerous living organisms examples able to locate the source of a scent, and the aim of the project will be eventually to devise an algorithm fulfilling a similar purpose.

The strategy employed to track the source of the chemical will depend greatly on how the signal is modelled. The signal tracked will first be analysed, and generic models of odour dispersion extracted. The basic functions the mobile robot must possess to complete the task will be clearly defined, then suitable strategies will be elaborated and simulated.

Finally, observation of the results with a critical analysis of the obtained algorithms will be provided.

The final outcome should provide guarantees of reliability, consistency , produce the shortest possible path to the goal if reachable, and return failure as quickly as possible if not.

The final algorithm has to be kept simple. Comparison has to be made with existing algorithms to determine its performance, and subsequent needed improvements.

2.1/ Instantaneous plume

Chemical plumes[3] can be defined as a region of space that contain all molecules released from a given source, and are entirely described by chemicals concentration and flow direction. Outdoors plumes are under the influence of undulation of the surrounding air, as well as angular shifts in wind direction.

Sources do not usually release odour in an uniform manner, but rather discharge chemicals in bulk flow. The odour is broken down into patches by the wind and gradually dissolves in the carrying medium.

Convection forces (wind) carry the chemicals in the direction of the air flow: Since the fluid flow twists and breaks the smell emitted from the mine, there is no smooth concentration gradient, but interleaved patches of odour and patches of ambient air.

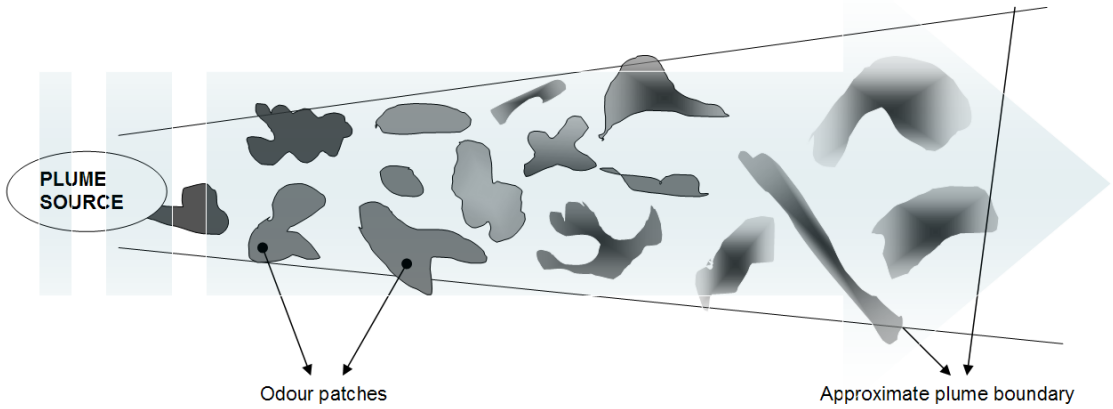


Fig 1: Real time plume

The plume concentration varies depending on time and space, with patches containing maximum concentration of chemical away from the source.

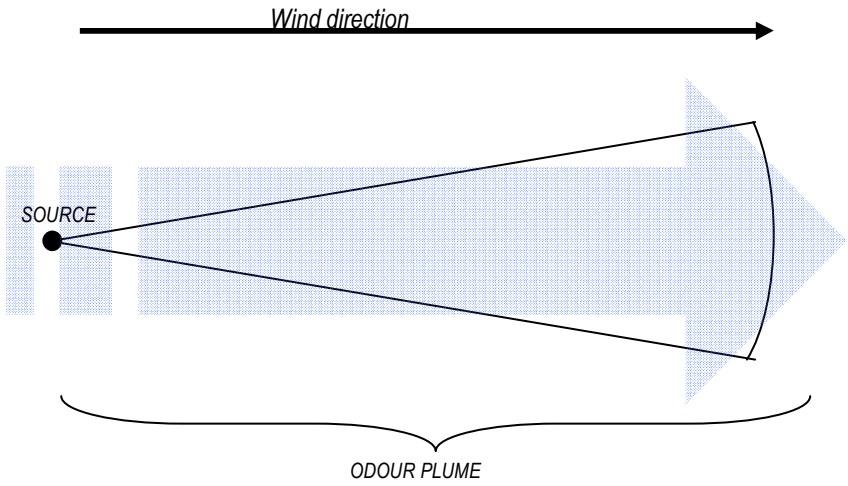
Odours can only be detected intermittently as patches are sweep by the wind. As the information received by the robot is intermittent, the gradient technique cannot be used to

determine the distance from the source. A mine sniffing algorithm must devise a strategy based on sporadic clues and partial information.

In theory, when the wind is blowing, no scent should be detected upwind of the mine. If the chemical emitted are detected, the source can only be searched in the upstream direction.

In the other hand, going downstream may increase the chances of detecting chemicals of interest if none are sensed.

The chemical trail will be schematised as a triangle having its most upwind point as the source of the chemicals, and their concentration gradient varying with distance.



2.2/ Time averaged plume

A real time (instantaneous) plume can be described as having a chaotic behaviour, but steady-state conditions exist when the concentrations values within the trail of chemicals are averaged, with respect to position. Continuous gradients from a chemical plume can be observed by averaging chemical levels over time within the plume, using a number of mathematical models, such as the Gaussian distribution or the Maxwell speed distribution model. The longer the

plume is time averaged on any axis orthogonal to the wind direction, the more accurate the concentrations "image" of the plume will be.

The Gaussian plume[4] is a simplified and idealised statistical model used to calculate air chemicals concentrations, and has been proven very effective to predict different concentrations levels under a variety of conditions.

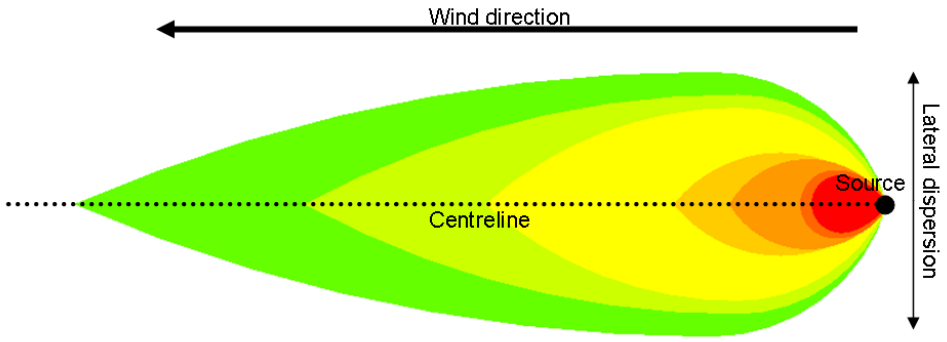


Fig 2 : Gaussian plume

The model assumes that leaked chemicals or materials are carried in a straight line downwind from the source. The distance and spread along the centreline is dependant on prerequisites such as the wind speed, wind direction and chemicals release rate . Additional parameters such as general meteorological parameters and geographical characteristics (Terrain, obstacles) can also be considered in order to refine the chemicals concentrations estimation.

2.2.1/ Gaussian Plume Model

The origin of the coordinate system (x=y=z=0) is located at the emission source. It is assumed that the wind goes in the direction the x-axis, and propagates laterally along the y-axis.

$$C(x,y,z)=\frac{Q}{2*\pi*u*\sigma y*\sigma z}*\exp\frac{-y^2}{2*\sigma y^2})*(\exp(\frac{-(z-h)^2}{2*\sigma z^2})+\exp(\frac{-(z+h)^2}{2*\sigma z^2}))$$

where:

1. C is the concentration of the emission (in micrograms per cubic meter) at any coordinates point x meters downwind of the source, y meters laterally from the centreline of the plume, and z meters above ground level.
2. Q is the quantity or mass of the emission (in grams per seconds)
3. u is the wind speed (in meters per second)
4. H is the height of the source above ground level (in meters)
5. σy and σz are the standard deviations of a statistically normal plume in the lateral and vertical dimensions, respectively .

2.2.2/ Carrying fluid flow statistical characteristics

The Gaussian plume model represent the spreading of the particles varying conditions, in our case atmospheric parameters. σy and σz are respectively the lateral and vertical dispersion[4] of the chemical trail, and quantify the spreading range of a set of chemical levels. If the standard deviation is small, the concentrations set is close to the mean value. If the standard deviation is large, the concentrations set is far from the mean value. Those statistical values describe the turbulent flow and are dependant on external conditions (Solar radiations, wind force), that need to be addressed for accurate concentration estimation. During the simulation, different settings, defining stable or unstable conditions can be selected.

3.1/ Robot motion planning

A robot planner[5] has to find a collision free path from a starting position to a goal. It implies some means to sense the goal, move towards it and deal with encountered obstacles. The whole motion planning process is based on local navigation, as the goal position and the obstacles laying ahead are unknown. The path of the robot should be gradually constructed while the algorithm is being executed.

To get the robot to a goal, in this case a land mine, two issues have to be resolved: localisation relative to the goal and navigation towards the given goal. Some mapping and memory elements, based on sensing, can also be used and will allow the robot to construct a representation of the environment. The algorithm should also include procedures to deal with encountered obstacles. The requirements for the ideal path planner are as follows:

Global Goal [5] **(**m line tracking / following)

Whether the robot knows the precise location of the goal or not, it needs a general direction and a straight line distance in order to go forward.

Artificial intelligence (Situation evaluation / Decision / Memory)

- Decision making[6] (Handle unexpected events).

  An unexpected event could be considered as anything that stops the robot progression towards the goal, basically obstacles. They have to be dealt with so the goal will eventually be reached. **Obstacles handling** is the main concern of the bug algorithms.

- Learning[7] (Improve performance in known similar conditions).

The algorithm should construct the path gradually and keep in memory all the necessary parameters supporting decision making.

3.2/ Behaviours definition

A behaviour-based planning strategy[3] is an efficient means to navigate an autonomous system in an uncertain environment. A behaviour represents the specific reaction of a robot depending on a specific event. A set of behaviours supported by some decision logic tree can be used to achieve a task if a if a suitable coordination is achieved.

Prior to designing an scent following algorithm, necessary features, based on biological models, must be determined. The overall odour tracing behaviour of insects can be broken down into smaller functions each having a specific purpose.

From the beginning, the position of the landmine is unknown, so is the chemical trail of chemicals leaking from it. Prior of tracing the plume up to the source, the first step will be to find it. In hardware terms, it means that the sensor must be detecting the desired chemicals. Plume finding[3] involves trying to make first contact with the chemical trail.

Many methods are available to fulfil this sometimes time consuming task, the most common being the random walk, which will be discussed later. Ideally, this task will be executed only once, this functionality being resource intensive.

Plume maintaining[3] is the logical follow up to plume finding. The chemical plume provide the robot with intermittent information because of the unpredictable air flow characteristics. The purpose of this task is to follow the odour up to the source, in an upflow direction (The robot itself being down the wind flow). This process is critical for mine finding, as the closer from the source, the more likely the released chemicals will be sensed.

Track-in techniques make the robot move upflow directly while track-out techniques cuts the wind, crossing the plume, covering a larger area.

Once the vehicle reaches the source of the plume, it should be capable of positively identify it. Although listed among behaviours, the source declaration[3] cannot be considered as such. It is rather a full stop, a "call off the search" procedure, as the source of the chemicals has been found. Each time the plume maintaining behaviour ends, the position of the robot is stored, depending on the distance on the direction of the flow. At the end of the search process, the most up flow point recorded is declared as the source.

As discussed before, the plume is made of intermittent patches and depending on the direction taken, the robot can easily find itself out of the plume. At this moment in time, a procedure must be in place so contact with the odour trail can be re-established. However, if the plume reacquiring[3] behaviour fails to get the robot back into the plume, the algorithm has to revert back to the time and resources consuming "plume finding" behaviour.

For the entire length of this report, all algorithms will be defined by the information they need for operation, a listing and description of their different behaviour, and a flowchart describing the relationship between them. In some cases, the algorithm will be too simple and pseudo code will be used instead of a flowchart.

4.1/ Ideal path response

4.1.1/ Ideal dour dispersion model

In the absence of turbulences, the strength of the scent particles emitted from the mine is submitted to the diffusion principle only. Diffusion[8] is an irreversible process by which scent particles move from one place to another. The process is unbounded in a sense that it never stops evolving. No strict rules apply concerning propagation of odour at macroscopic scale, but it can be assimilated (and approximated) as a signal decreasing in intensity over distance from source, and ideally quantified in intensity as an exponential decay.

Under those conditions, the distance from the source and its direction can easily be extracted from the information sensed. If the resolution of the sensors is infinite, the value of the signal will be proportional to the distance from the source, and a mine sniffing robot will be able to know the exact direction (corresponding to the highest gradient) and the distance separating itself with the mine.

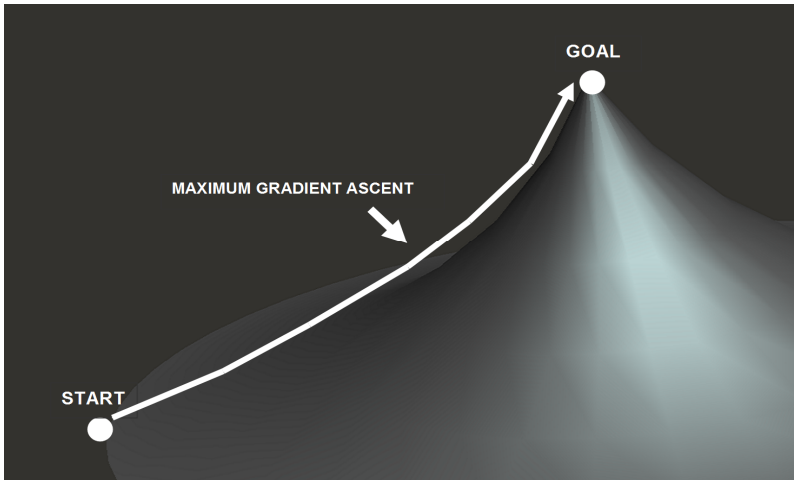


Fig 3 : “Smooth” exponential decay model

However, the sensors carried by mobile robots usually need the odour level to be above a certain threshold prior to detection. On top of that, they are also subject to some level of granularity: the sensor's resolution limits reading to minimum increments steps, and small changes in the medium cannot be monitored.

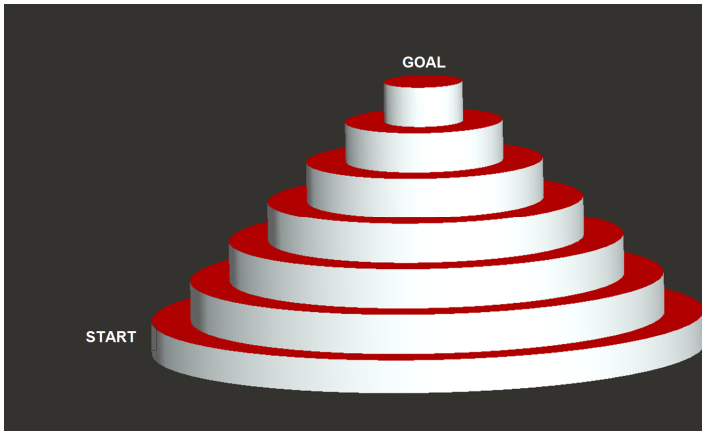


Fig 4 : "Staircase" exponential decay model

4.1.2/ Ideal path response to reach the goal

Bug algorithms generally exhibit two different behaviours: advance towards the goal by following the highest gradient, or boundary following. In order to work, the "bug" has to know the direction towards the goal, the exact distance from the goal, and the proximity of nearby obstacles at all times. Because of their simplistic nature, bug algorithms will fail in a number of situation, or provide an overcomplicated path.

a. Artificial potential field methods

There are many existing potential fields algorithms[5], most of them based on the same basic principle. The robot is treated as a particle attracted to the goal and repelled by obstacles. It provides the smoothest , but not often the shortest possible path towards the goal.

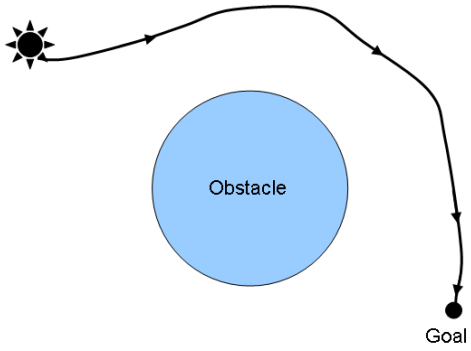
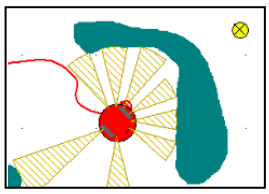


Fig 5 : Artificial potential field

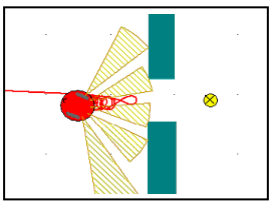
The APF can be qualified “unidirectional”: It is going towards the goal only and does not provide the ability to backtrack.

This algorithm is vey sensitive to the local minimum problem and the doorway problem.

- Local minima problem: The forces attracting the robot are equal to the ones repelling it. The robot cannot proceed.



- Doorway problem: The repelling forces of two close by obstacles are stopping the robot from going forward even thought there is enough room for it to go through.



b. Bug1

Bug1[5] exhibits two behaviours: Motion to goal, and boundary following.

The robot goes straight towards the goal until it comes across an obstacle. The entry point is kept in memory, and the robot goes round the obstacle until it returns to the hit point . The robot determines the closest point to the goal on the perimeter of the obstacle, goes there, and resumes its route. It is an exhaustive search algorithm. The main advantage besides its simplicity is that if there is a solution, the algorithm will find it. It also has a very predictable result, and is reliable.

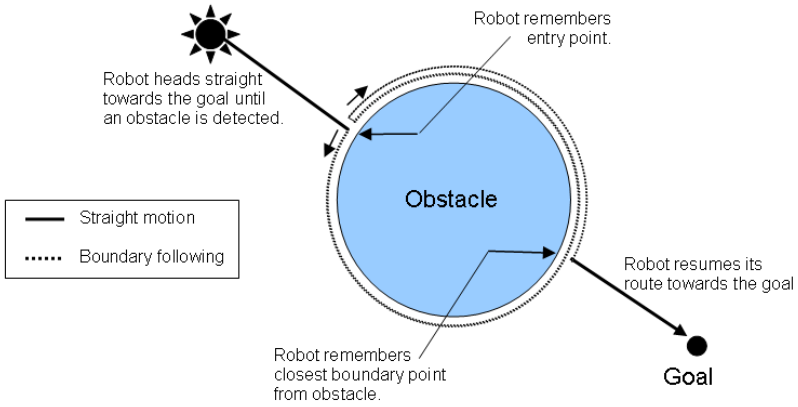


Fig 6 : Bug 1 algorithm

c. Bug2

Bug2[5] seems like an improved version of bug 1, and exhibits the same motion to goal and boundary following behaviours .

In this algorithm, the straight line between start and goal remain fixed. The boundary follower is activated when an obstacle is encountered. However, the robot does not have to go round the whole obstacle, making for a shorter path overall. It will resume its path to the goal as soon as the m line is encountered. It is an opportunistic search algorithm. Bug2 is more efficient than bug1 for simple obstacles. However, its comparative performance degrades when the complexity of the obstacle increases.

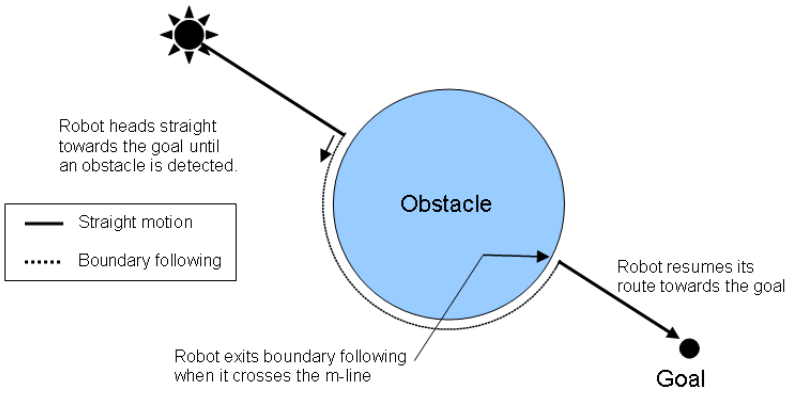


Fig 7 : Bug 2 algorithm

d. Tangent bug algorithm

The tangent bug algorithm[5] is an improvement over Bug2 in a sense that it will leave periphery of the obstacle as soon as the goal is in sight. It could be described as an ever changing m line in a bug2 algorithm. Furthermore, when hitting an obstacle, the robot will chose the boundary following direction that minimises the most its distance from the goal.

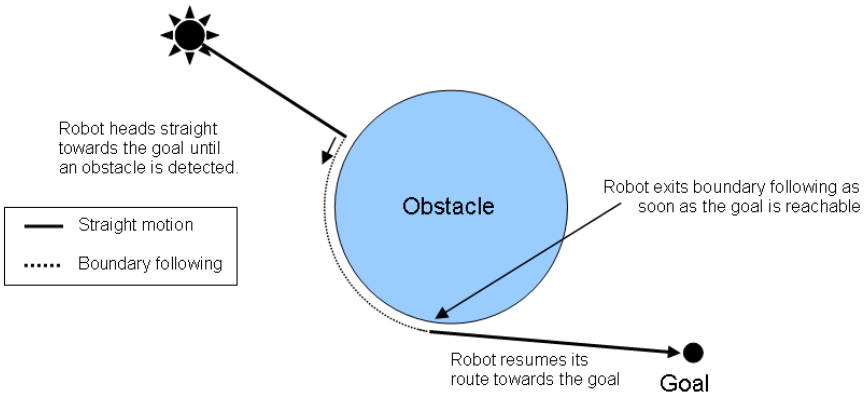


Fig 8 : Tangent bug algorithm

4.2/ Biomimetic strategies

Mother nature is offering scientists many examples of animals able to follow an air or water chemical plume up to their sources. Not surprisingly, they have been the main inspiration behind all the chemical plume tracing techniques. The numerous studies made on simple organisms such as insects or crustaceans, allowed the emergence of two main biomimetic strategies[3], theoretically in every way as effective as their living counterpart.

4.2.1/ Flying insects inspired strategies

Most insects are driven by food and reproduction. Both tasks involve following a chemical trail of hormones, or food odour, up to the source. The challenge for those organisms is to have an effective strategy[3] that will enable them to determine the location of the chemical source, even though the plume concentration is intermittent, the distance of the detected odour is unknown, and the flow of the carrier fluid varies with time and position.

The strength of the wind also has an incidence since it narrows and breaks down the plume[9], making gradient ascent technique not applicable: the dispersion of chemicals by turbulent fluid flow has complex structures and often characterised by many local maxima and minima[3]. The searcher detects odour in a sporadic sequence depending on its encounters with patches of fluid (or air) where turbulent mixing has failed to dissipate the odour.

The all flying insects sense the chemicals present in the environment with a number of chemo-receptors, which, although sensitive to concentration levels, work rather like binary detectors: the chemicals are either sensed or not. They also have the ability to measure the direction of the airflow so that they can orient themselves, depending on atmospheric conditions, while searching for the source of the chemicals.

a. Anemotaxis principles

In a minefield, the chemicals leaked by the mine are subject to the turbulences caused by the wind. A strategy based on following the highest gradient has little chance of being successful. The basic principles of anemotaxis[10] are continuously referred to when it comes to plume tracing algorithms.

Anemotaxis has been directly inspired by flying insects behaviour. This source searching strategy is using both the direction of mean flow of the carrier and chemical detection for guidance. Those two parameters are extremely important for successful plume tracking because of the random nature of a turbulent plume.

The linear search[11] behaviour is common to most living organisms employing anemotactic search, and is used to either make first contact with the plume or reacquire it when all scent information has been lost . It involves travelling across the wind on a straight line, therefore covering a greater upwind area.

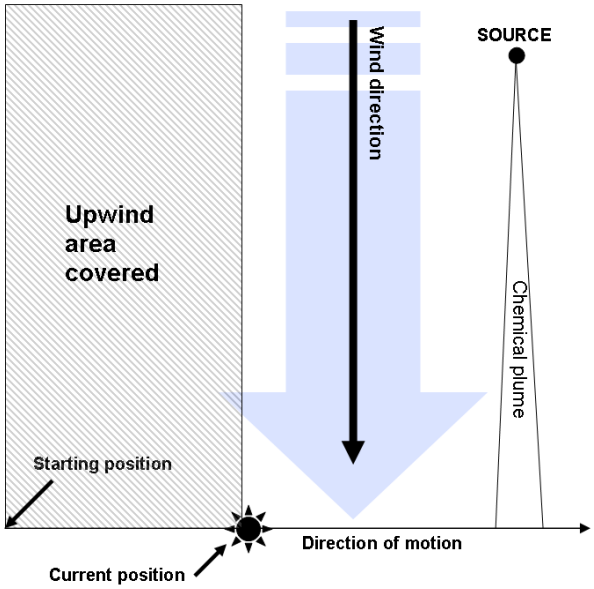


Fig 9 : Linear search

Detection of chemicals means that the source is located upwind.

The “Cast”[11-12] behaviour of insects is a procedure aiming at making first contact with the plume and is based on the linear search principle. If the insect determines that it is outside the plume, it starts moving back and forth directly across the wind, covering an increasing distance with each turn. Upon odour detection, the insect gets back into the chemical trail.

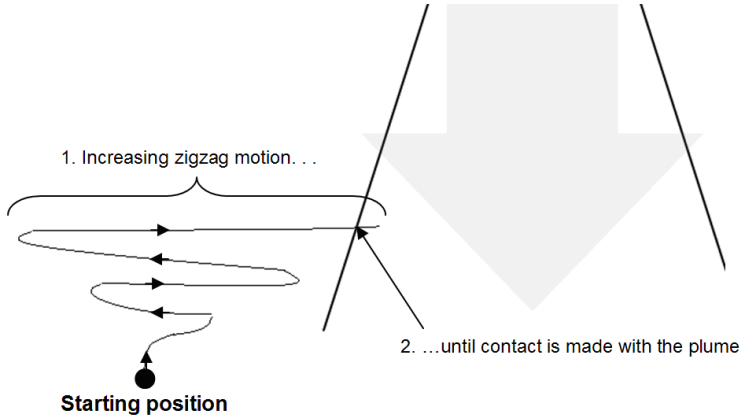


Fig 10 : Linear search against the wind (Casting)

Once the chemical has been detected, however, most insects evaluate if they are within the plume, and moves forward against the wind, or “Surge”, following a pattern specific to the specie, for as long as the odour can de detected.

The Simple Surge Anemotaxis[12] will correct its forward motion (Surge) to move exactly against the wind at all times, and move in the direction of the highest gradient. If the value of the chemicals stagnates or drops, the algorithm goes back to casting.

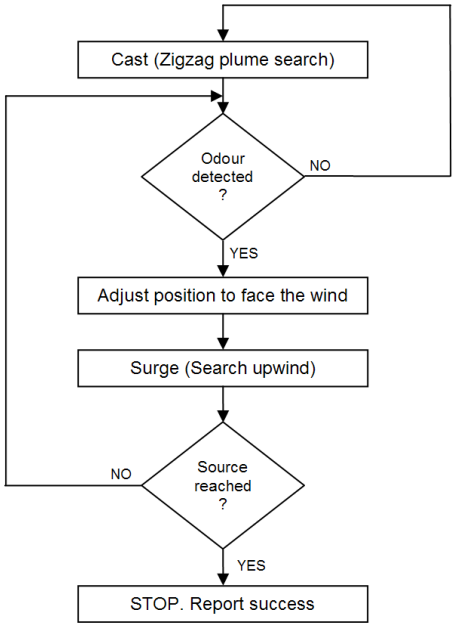


Fig 11 : Anemotaxis algorithm

By nature, the wind often changes direction. Observations on insects have demonstrated that the simplest way to re-enter the plume after a wind shift is to move perpendicular to the wind towards the centreline of the of the new position of the plume. To achieve this, changes in the wind direction are constantly monitored. As a result, small wind changes do not affect the Surge anemotaxis with wind-shift adjustment[12] and allows insects to keep track of the plume without reverting to the time and resource hungry casting.

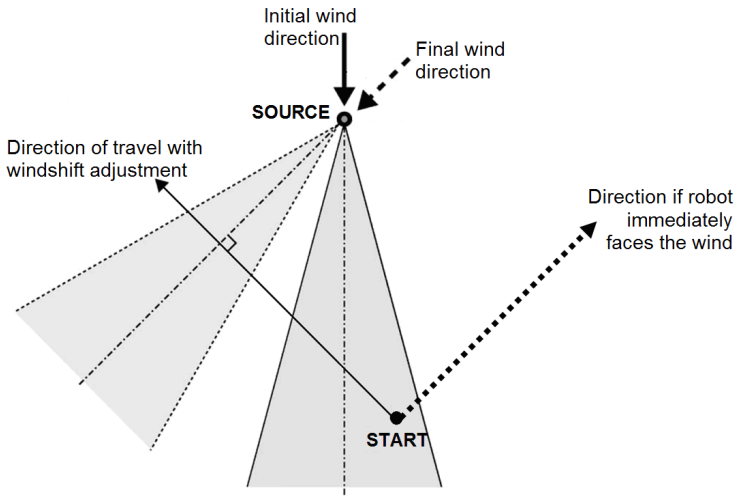


Fig 12 : Anemotaxis reaction to wind-shift

b. The dung beetle

- **The dung beetle strategy**

The dung beetle[11] has the ability to measure airflow direction , and possess only a single sensing organ. In order to find or get back into the plume, the beetle will perform a zigzag motion with increasing amplitude against the wind flow until contact is made with the plume. The plume maintaining behaviour is also made of the same casting motion, the insect will move across the full width of the plume crossing the wind in the opposite direction every time the plume is lost. End of the plume usually signals that the dung beetle is just above the source.

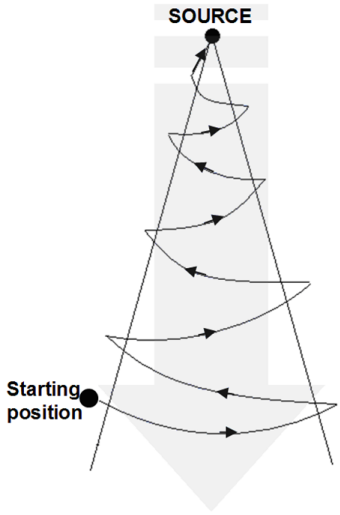


Fig 13 : Dung beetle surge behaviour

- **Past work**

The dung beetle algorithm[11] has been implemented on a mobile robot by Ishida et al.

The robot[13], equipped with a unique semiconductor oxide gas sensor and an anemometer, could track an ethanol vapour plume. The robot's dung beetle algorithm behaviours were

similar to its biological counterpart concerning plume maintaining. However, no precise details were provided concerning the plume search procedure and obstacles handling.

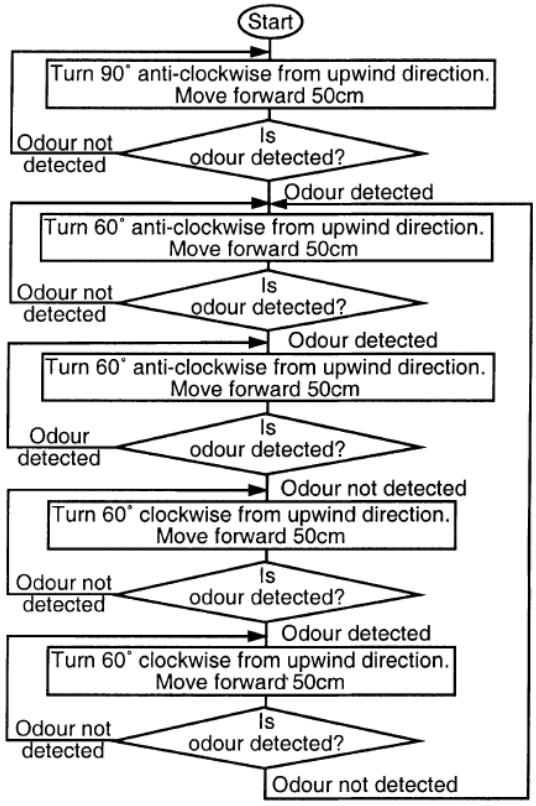


Fig 14 : Dung beetle algorithm [11]

Upon sensing ethanol, the robot was successful in finding the source of the trail, and highlighted the importance of the source height and the estimation of the gas release rate for an efficient scent following scent following mobile robot.

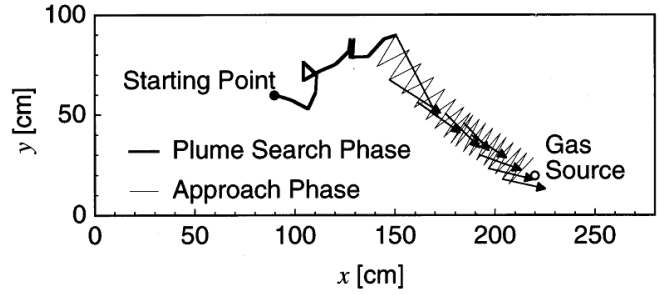


Fig 15 : Result of on-line identification of gas source location [13]

c. The Male moth

- **Male moth strategy**

Moths[14] are flying insects closely related to butterflies, and come in all shapes and sizes. They possess two odour sensing antennules on each side of the head. The insect's plume finding and plume reacquisition method is a casting behaviour similar to the dung beetle. Upon detection, the male moth will "surge", search upwind, until the source is reached or the plume is lost again.

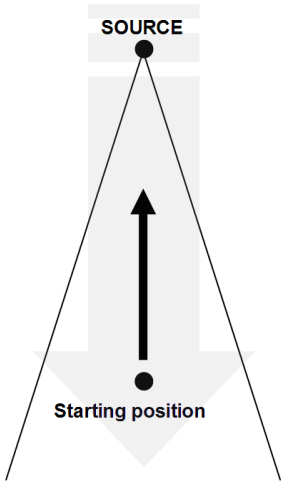


Fig 16 : Moth surge behaviour [11]

- **Past work**

Harvey et al designed and constructed a mobile robot[12] equipped with an ion sensor, and an anemometer able to monitor wind speed and direction. The robot had the unique purpose to assess the "surge anemotaxis" strategy and improve on the algorithm. Trials were conducted in a tunnel which fan structure is able to generate wind-shifts. The source was considered reached should the mobile robot get within 35 centimetres of the source.

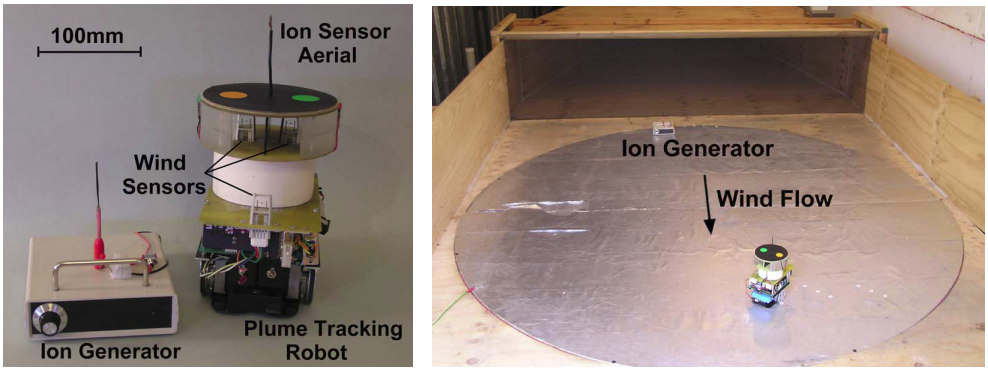


Fig 17 : Apparatus used [11]

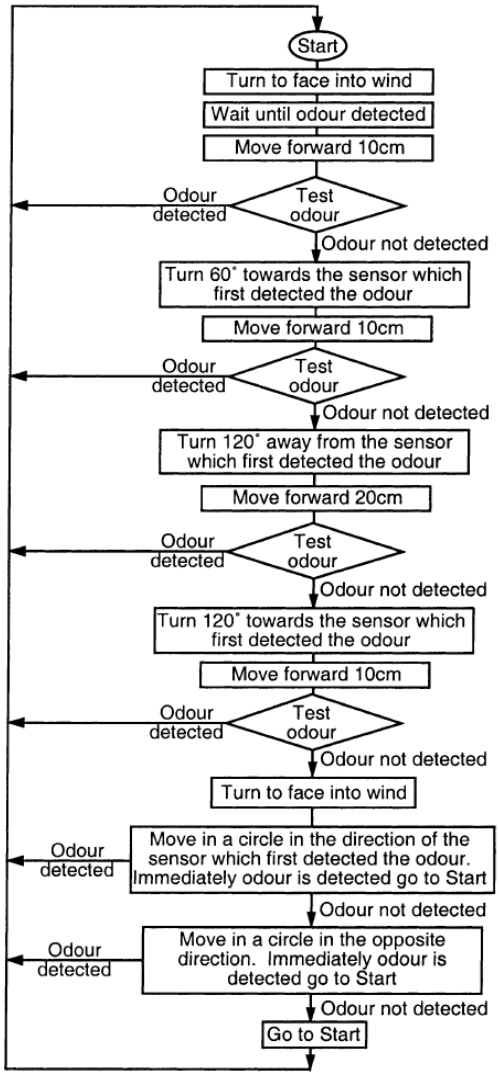


Fig 18 : Moth algorithm [11]

Experiments and trials showed the less wind-shifts, the shortest the time the robot will take to get to the ion source. They also noticed that waiting for the signal to be above a certain threshold level before starting the search improved greatly the performance.

Some failures occurred because of the limited size of the test area that, in some cases, doesn't have the room to allow wind-shifts adjustment to be made: obstacles were not accounted in any way.

4.2.2/ Lobster/Bacterium inspired strategies

a. Chemotaxis

Chemotaxis[15] is a mechanism by which simple living organisms direct their motion depending on the chemicals present in their surrounding. For example, in order to survive, a micro-organism might want to find nutriments (chemoattractants), and avoid hazardous substances (chemorepellants). Chemotaxis is described as positive when attracted by a substance, or negative when the direction taken is opposite the source location.

This strategy need lengthy time averaging[9] and can be used only when there is little to no flow: high convection in the medium would break the gradients structure of the plume.

b. The lobster

- **The lobster strategy**

Lobsters[16] are invertebrates marine crustaceans. They have the ability of tracing in water chemical plumes up to the source of the released chemicals. They possess on their heads a pair of antennules, acting as sensors. The concentration information collected is used in a very simple fashion to steer the lobster towards the source: it will orient its forward motion in the direction of the antenna sensing the higher concentration.

- **Past work**

Naeem et al implemented the simple lobster chemotaxis behaviour in a mobile robot. Trials and experiments were conducted underwater, where "Robolobster"[3] tracked a salt plume

with two onboard conductivity sensors as antennules. Two simple rules were followed. The robot would turn towards the sensor reading the highest concentration, and proceed forward when both report approximately the same value. The plume reacquiring is done by going backwards, allowing the robot to re-enter the plume. No details are given in the research paper concerning the plume finding method.

Information needed:

Value sensed by left antenna (S1)

Value sensed by right antenna (S2)

Plume search behaviours: (When no chemicals are detected) :

Turn left

Turn right

Go Backwards

Plume maintaining behaviour (When chemicals are detected): Go forward

Pseudo code:

```
While S1>0 or S2>0 (Upon detection of the plume)
        If S1 = S2, Go Forward
        If S1 > S2, turn left
        If S1 <S2, Turn right
        If S1 =S2=0, Go Backwards
```

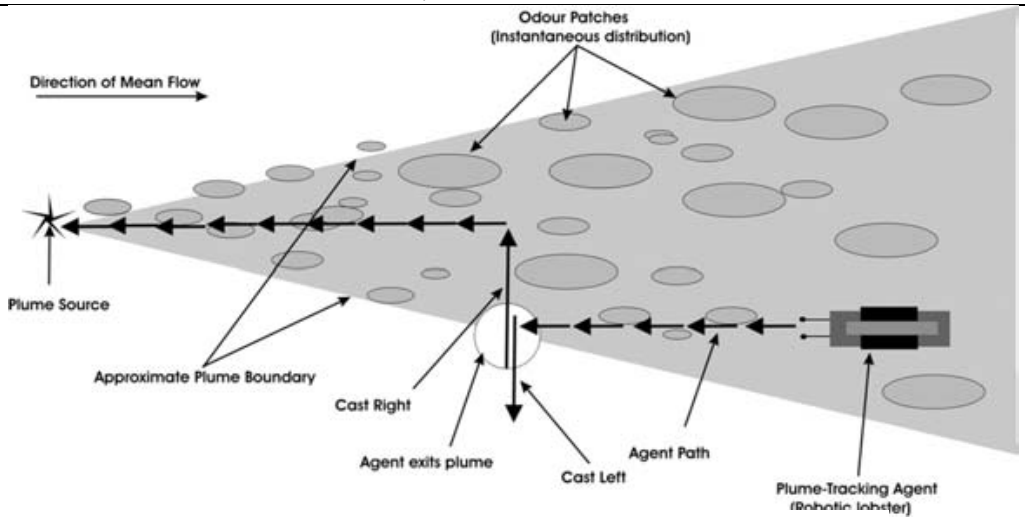


Fig 19 : Robot performing lobster strategy [3]

c. The Bacterium

- **The Bacterium strategy**

At microscopic scale, the odour dispersion is very similar to the ideal exponential decay model: the closer one gets to the source, the more concentration of chemicals.

It allows bacterium such as the Escherichia coli (E.Coli) to find its way to food, using gradients of released chemicals[17].

Bacteria are sensitive to the chemical levels contained in its environment. They are able to measure the medium concentration of the substance of interest at different intervals. The bacterium's flagellum allows two different locomotion behaviours.

The "Run" behaviour allows a straight motion in a particular direction, and the "tumble" behaviour allows the bacterium to choose a random direction and reorient itself for another run. In the absence of gradients, the bacterium executes a random walk, and as the concentration level increases, the tumbling frequency is decreased until the source is reached. General motion towards the goal is ensured by reducing the tumbling frequency. The E.Coli bacterium works almost exactly like the bug algorithms, although no sufficient information is given concerning obstacles handling and source declaration.

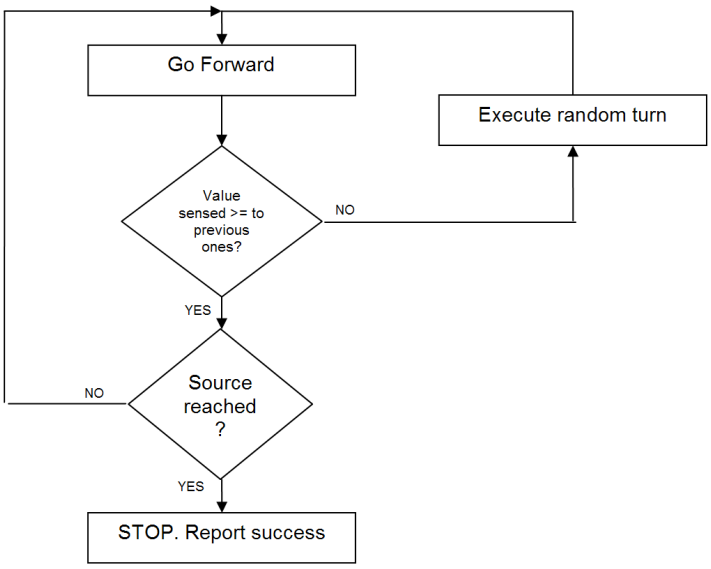


Fig 20 : E.Coli algorithm

When no gradients can be sensed, the bacterium walks around randomly. As soon as a chemical is sensed, it tries to proceed towards the source following the highest gradient, changing direction when necessary.

Although very efficient in static fluid flow conditions, the bacterium will fail when conditions are unstable, or when sources are small, weak or mobile.

- **Past work**

In "Bacterium-inspired robots for environmental monitoring", Dhariwal et al came up with a novel technique using a random walk to simulate the bacterial chemotaxis. The algorithm was implemented in a light source tracking ROBOMOTE[17] mobile platform equipped with a light sensor.

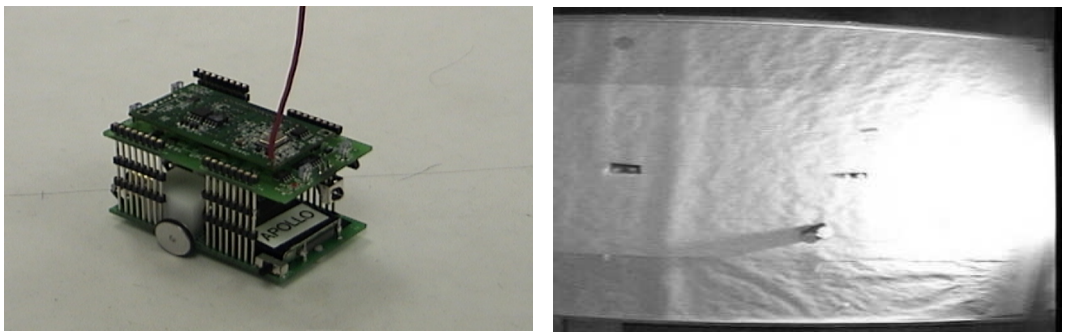


Fig 21 : "ROBOMOTE" and its environment [17]

The behaviour of the robot is based on bacterial motion, that employs two behaviours: "Run" and "Tumble". The robot has a tendency in going straight when it is in a gradient ascent situation, and in changing direction of motion otherwise. This algorithm requires very little memory, as only the last concentration reading needs to be stored, and subsequently compared to the next reading.

The light diffusion model[17] for the set of experiments has been approximated to an inverse square law distribution:

$$Intensity(x,y) = \frac{1}{K}\sum_{i=0}^{m}\frac{q_i}{r_i^2}$$

Where:

- q_i is the intensity of the source S_i
- K is a constant of proportionality
- r_i is the distance between the grid point (x; y) and the centre of source S_i

Another set of experiments have been conducted under source intensity variation over time[17]:

$$q_i(t) = (q_i(0)e^{-k_1 t}) - k_2\sum_{j=0}^{t} N_{ij}$$

Where

- $q_i(0)$ is the initial intensity of source S_i,
- k_1 and k_2 are constants whose values are set based on the type of source we are trying to model
- N_{ij} is the number of robots at source S_i depleting its energy at time instant j.

Results showed that the randomness of algorithm made prevents the robot finding itself trapped in a local maximum, demonstrated the importance of the "bias" (Change in tumble frequency) in the success of the search. The strategy was also proven successful in finding small or intensity varying sources .

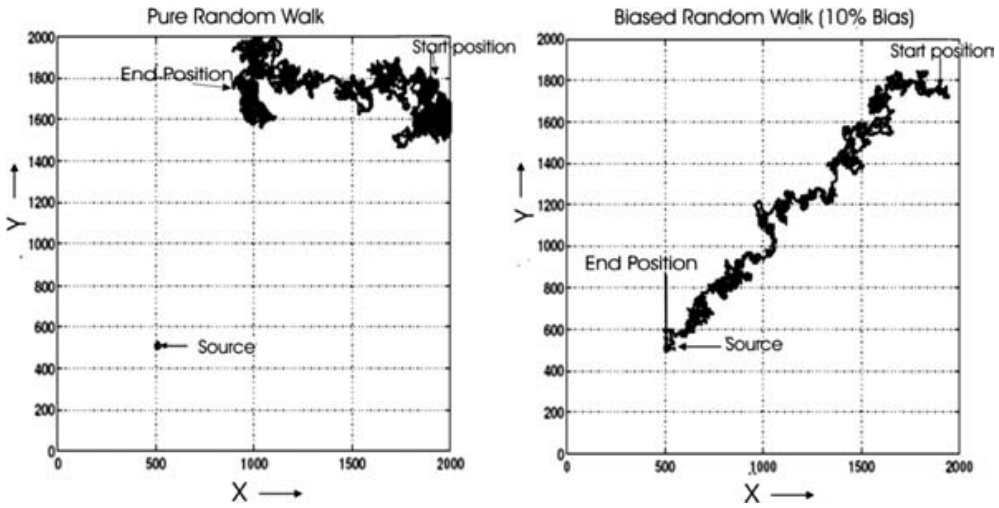


Fig 22 : "ROBOMOTE" results[17]

Living beings, especially insects, show a variety of behaviours optimized for their respective environment and purposes. They will be used as an inspiration. Computer simulation will be useful to evaluate the applicability of each algorithm in various simulated situations. The limit of the tested algorithm will be clarified, and comparison between different strategies will be possible.

Simulation is a useful way to accelerate the development of plume-tracking systems. It takes a long time to develop a new system by trial and error, let alone engineer specific, short lived hardware and software for the purpose of an unique algorithm.

There are several ways the simulation aims of this project could be achieved. The most precise way would be to simulate all the dynamics of fluid (Odour dispersion) and the kinetics of sensors and robots (Robot programming) in a computer. However, this method is unlikely to be achieved, simulation of turbulent flow with fine resolution requiring time and resources not available for this project. It is however possible to produce a simulation based on "visualized" plume data . This level of simulation is effective in testing algorithms and determining behaviours requirements for the successful deployment of a mine sniffer. The simulation techniques are expected to facilitate this progress by providing a way to compare algorithms performance in a variety of situations.

5.1/ Robot simulation: KIKS

Kiks Is a Khepera Simulator, will be used for this project. KIKS is a robot simulator that works as an extension of Matlab. It is going to be used purely as a visualisation tool for the produced algorithms. This application has been chosen for it relative simplicity over more advanced robot simulation software and because its open source nature allows for free modifications via Matlab code or C.

However, being based on a real robot it suffers from some limitations, mostly linked to the reduced capabilities of the Khepera itself. The distance sensors working only at very close range , the robot can only detect nearby obstacles. It can be assimilated to a blind person able to detect obstacles only via physical contact. There are also some relative restrictions on the obstacles that can be drawn in the simulator, they can only be square shaped.

Finally, the main drawback is inherent to Matlab itself, as it does not support anything like concurrency, ruling out a parallel real time simulation of both the robot and the odour dispersion mechanism. Three algorithms will be tested.

- The tangent bug algorithm uses distance and exact direction towards the goal for guidance, and represents the shortest way to get from the starting position to the source, and will be the benchmark the others algorithms will be compared to.
- The E.Coli algorithm based on chemotaxis search will be using the concentration level at its current location alone to find its way towards the source.
- The male moth algorithm based on anemotaxis search that need two elements of information: the wind direction and the current odour status level.

5.2/ Robot's environment

The mine field being searched is usually strictly delimitated: it is unrealistic to send a mobile robot to an unbounded area. The current demining practice first define clear paths within the minefield that can be walked safely, then subdivides the remainder of the area into roughly 30 x 30 meters squares that are then searched the one after the others.

The robot will have at least to deal with the limits of the minefield, that will be treated as obstacles.

A mine field will also contain natural obstacles such as rocks, trees, and in general all areas human beings cannot set a foot on. They will be dealt with on contact, as it cannot be excluded that a mine could sit very close or under their periphery.

The robot's environment, that all algorithms will face, will be modelled in such a way that it mimics its actual minefield counterpart. The simulator window will be made of a 3000 x 3000 pixels matrix, and the odour dispersion parameters chosen to represent an area of 30 x 30 meters.

Obstacles representative of the one found in rough terrain will also be included, allowing for a simulations conditions as close as possible to the reality.

The time taken to identify the location of the source is critical in assessing the performance of an algorithm. In order to maintain inter comparison consistency, the robot speeds of boundary following , rotation and "Go Forward" should be common to all algorithms. The distance from goal at all times, and distance covered, will complete the list of parameters recorded.

5.3/ Odour flow simulation (Wind simulation / Odour values look up table)

The odour sniffing behaviour of the robot will be largely inspired by those of insects. In order to make the simulation realistic, the robot should be given only similar information concerning the environment: an odour level from an onboard sensor, and obstacles proximity.

The detection of the obstacles can be done with the help of the infrared sensors, integrated in the design of the robot.

KIKS will not allow for an additional odour propagation simulation. Each pixel of the "Arena" (onscreen representation of the robot's environment), can be assimilated as an element of a matrix.

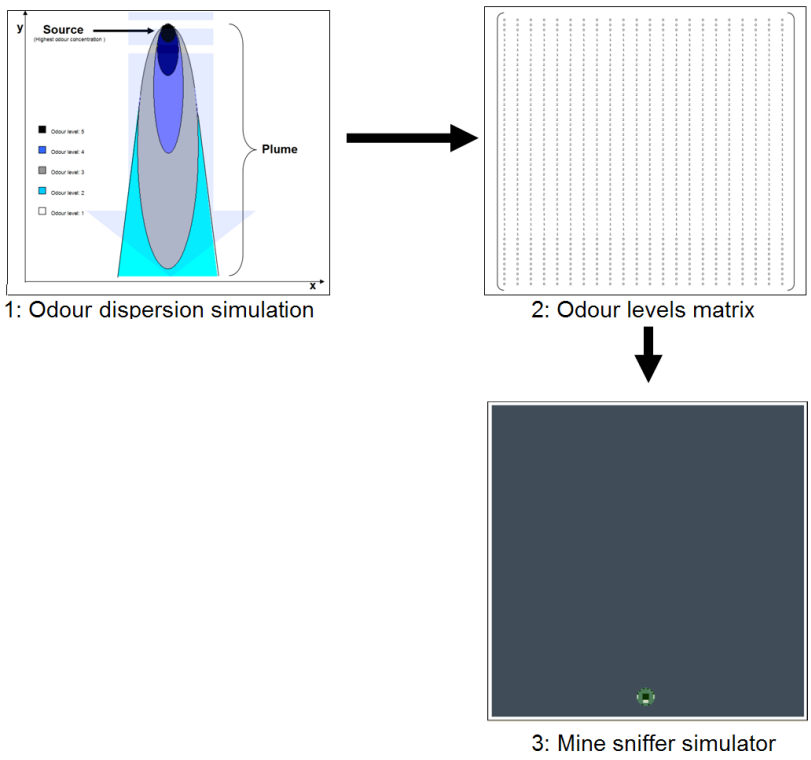


Fig 23 : Odour simulation procedure

As illustrated, the chemical dispersion gradients will be simulated offline(1), turned into a matrix of exactly the same size of the arena (2), that will be provided to the robot as realistic sensorial information (3). The whole matrix will be available at all times, but the robot will have access only to the value corresponding to its coordinates (Location in the environment).

The procedure previously described can be used as such if the flow of chemicals from the source is considered static, and not disturbed by wind turbulences. A three-dimensional matrix will be extracted from the chemical dispersion simulation, representing the changes of odour levels over time. The simulated robot will only have access to the odour value corresponding to the current coordinates and simulation time.

Those two concepts will be further expanded in their respective algorithms sections.

Information regarding past position and chemical concentration sensed are stored by the robot. The plume tracking algorithms will be compared in terms of the resources used, such as time to reach the source and the accuracy / precision of the returned result.

Dynamic goal tangent bug algorithm

The tangent bug algorithm represents in most case the shortest path from one point to another, obstacles being detected on contact. It will be used as the benchmark for mine sniffing algorithms, as it supposed to require minimal time and resources to run to completion.

Basic principles

1. The base of the algorithm is the basic tangent bug algorithm

Information needed: Distance from goal, direction of goal.
Behaviours: advance towards goal, follow boundary of obstacle

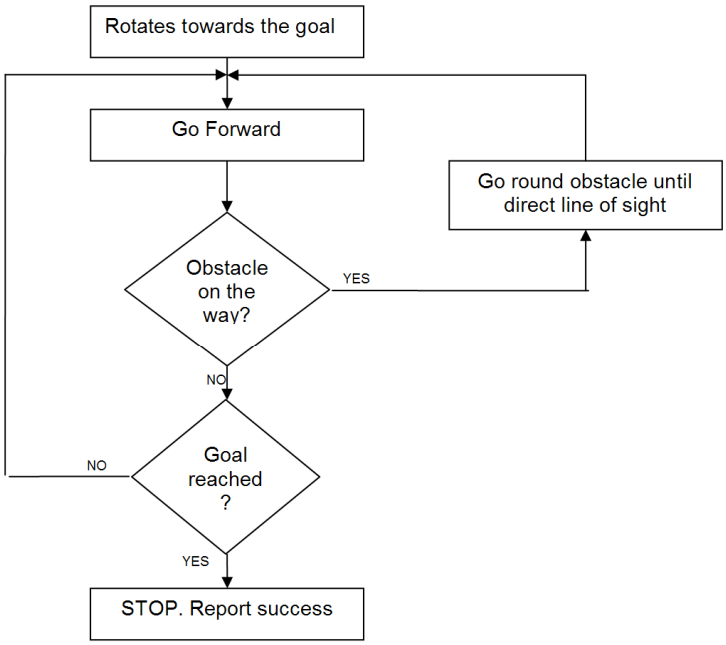


Fig 24 : Basic tangent bug algorithm

2. Every time the robot hits an obstacle, a decision will be made about which direction the boundary should be followed. The robot will always take the direction that

maximises its progression towards the goal, by following the highest gradient (Standard decision).

The location coordinates where every obstacle has been encountered will be stored (Hit point) and numbered in order of occurrence. If any progress cannot be made any further towards the goal, using the standard decision, the robot will go back to the precedent hit point and follow the boundary of the obstacle in the other direction.

3. The notion of "Tolerance" is introduced. The robot needs information about when it has to return to the precedent hit point. The Tolerance at a hit point is defined as the distance between itself and the precedent hit point: It is assumed that past this distance, any position can be achieved from the preceding hit point.

   When the robot is following an obstacle, it is not allowed to go further away than:

   Closest ever distance from goal +Tolerance

4. Finally, in the hypothetic case the robot comes back to the starting point, and has tried all the combinations without success, the algorithm reports that the goal is unreachable, but will provide the location of the closest distance from goal coordinates it has been to.

Notes on the "dynamic goal tangent bug algorithm":

- With an infinite tolerance, the algorithm reverts to a classic "Tangent bug algorithm".
- The tangent bug algorithm is merely an locomotion means that supports a kind of exhaustive search logic, starting as close from the goal as possible, going back when a solution is not found.

- Some success has been achieved coding the algorithm using KIKS, but the imperfections of the simulator itself makes the outcome rather erratic (No absolute consistency in the results).

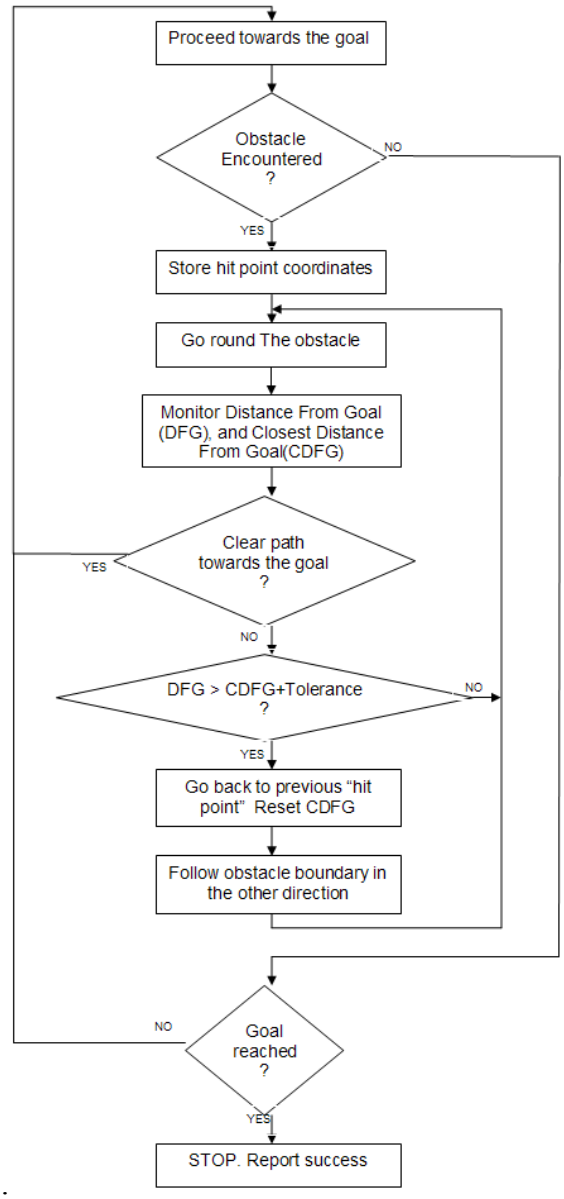


Fig 25 : "Dynamic goal" tangent algorithm

Results

The algorithm is given very precise information concerning the distance from the source an the direction to be taken to reach it at all times. It represent the shortest possible path towards the source.

The “Dynamic goal tangent Algorithm” works perfectly as a “simple” tangent bug. However, the search logic involving “tolerance” and “Hit points” fails in most cases to perform as expected. The algorithm could have been further improved, but given time limitations, further work has been judged outside the scope of the project: minefields are usually simple environment.

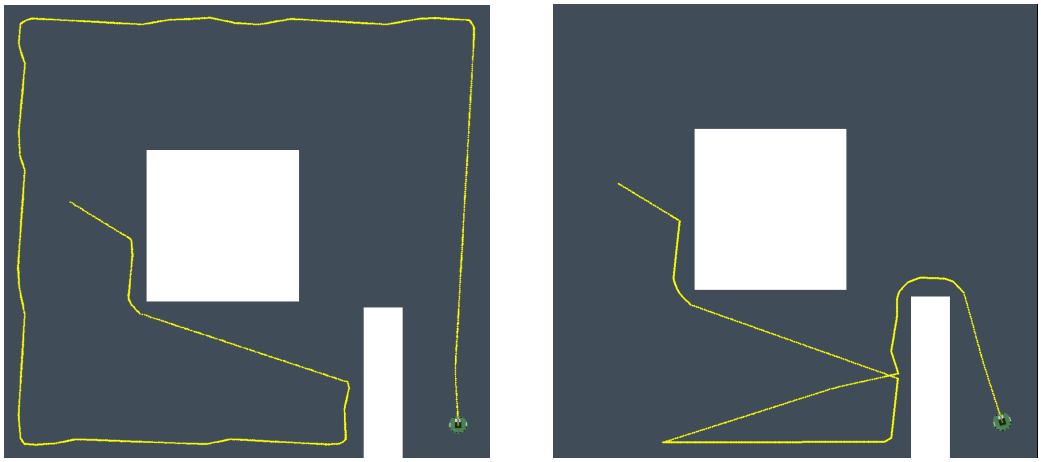


Fig 26 : “Basic”(Left) and “Dynamic goal” tangent algorithm responses(Right)

7.1/ Signal model (Time averaged Gaussian plume)

The chemotaxis strategy requires odour levels in order to make decisions concerning the robot's motion. The strategy is efficient over short distances from the source or when the fluid flow is low or inexistent. A Gaussian plume will be used, and will provide the algorithm with already time averaged values. The source coordinates are located a X=Y=0, and the averaged odour level at the robot's plane position (X:Y) will be calculated according to the following formula:

$$C(x,y,z)=\frac{Q}{2*\pi*u*\sigma y*\sigma z}*\exp\frac{-y^2}{2*\sigma y^2})*(\exp(\frac{-(z-h)^2}{2*\sigma z^2})+\exp(\frac{-(z+h)^2}{2*\sigma z^2}))$$

The Z value corresponds to the distance of the sensor from the ground, fixed at 0.1m (10 centimetres) from ground. Chemicals released by the mine are not supposed to be detected upwind of the source. The value of all those (X,Y) coordinates will be assigned a value of 0, representing a total absence of odour.

Usually, robots have to travel very slowly or stay stationary for the while necessary to obtain accurate averaged concentration values. This waiting process is not going to be necessary, allowing for a much faster algorithm simulation.

7.2/ Obstacle handling

The obstacle handling function of the E.Coli algorithm will be inspired by the Bug 1 algorithm, and will be used only when the robot is in plume maintaining mode.

While going round the whole periphery of the obstacle, the coordinates of the highest scent level will be memorised: the E.Coli algorithm will search for the highest chemical concentration, just as the Bug1 algorithm is looking for the obstacle boundary point closest to the goal.

While the robot is in plume search mode, detection of obstacles will cause the robot to stop and continue the random search in another direction.

7.3/ Algorithm

Information needed: chemical concentration

Plume search behaviour: (When no chemicals are detected) :

- Tumble (Robot chooses a random direction)
- Go forward (Travels on straight line for a random distance)

Plume maintaining behaviour: (When chemicals are detected)

- Tumble (Robot chooses a random direction)
- Go forward: Robot progresses towards the chemical source

Source declaration: (Estimation of the source location)

Real bacterium certainly use physical contact to identify the source, usually food. None of this is available in the case of mine detection. In the case of chemotaxis search, the highest odour value detected by the robot will be considered as the source, and its coordinates stored until an highest value is encountered.

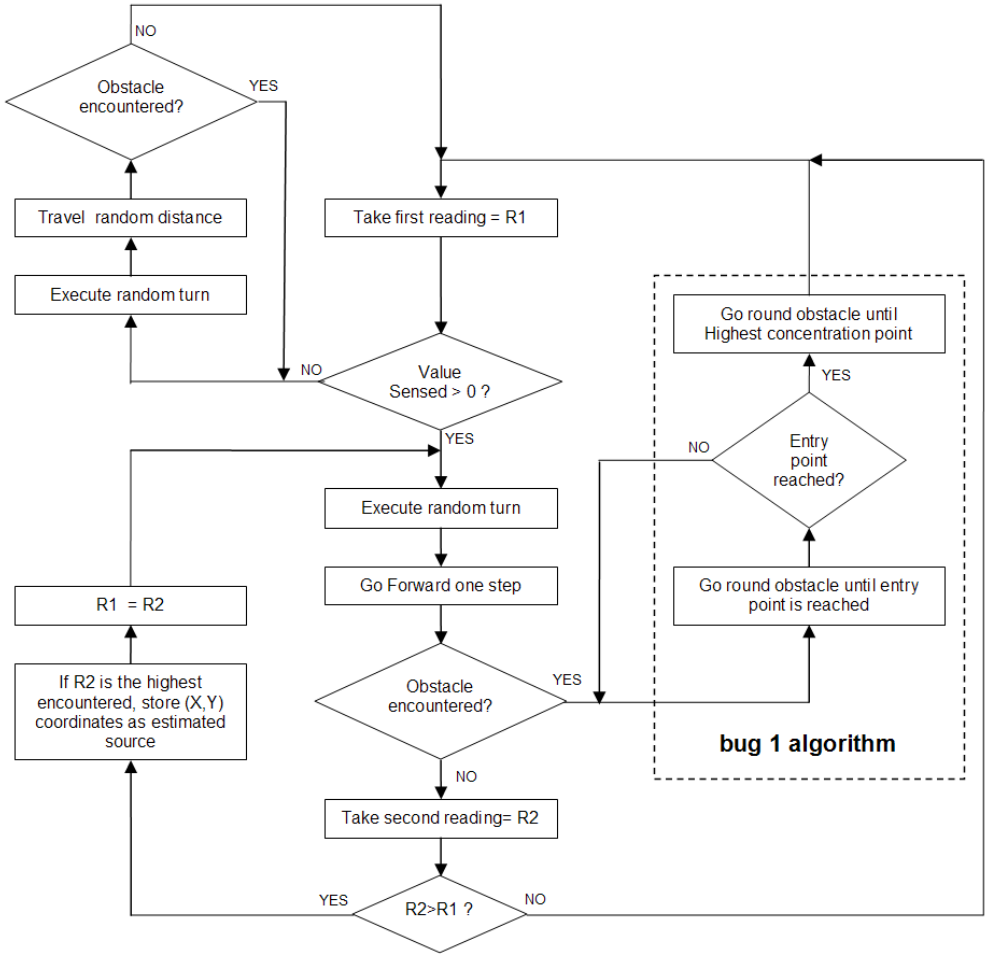


Fig 27 : E.Coli-Bug1 Algorithm

General principles of the algorithm:

In absence of detected chemicals, the robot walks around randomly: It chooses a random direction, and travels for a random distance. If an obstacle is encountered during the random search, the robot stops, chooses another direction, and resumes its random walk.

From the moment the sensor detects an odour level above the threshold, the robot switches to plume maintaining mode. It will make a reading, move forward for a given distance, and make a reading again. Those readings are compared, and if the level of the latest reading is superior, the robot proceeds forward for a defined step. If the value read gets worse, the robot pick another random direction and tries again its luck.

7.5/ Results

7.5/ Results

The performance of the E.Coli-Bug1 cannot be easily assessed, given that it does not return a result. The results obtained through simulation of the algorithm will be mainly descriptive.

1. The random walk behaviour is used to make contact with the plume, but the robot often stays confined on the same area of the minefield. Once the odour plume has been found, the algorithm produces very slow progress towards the goal, and proves to be time and resources intensive at all times. Fig 28 highlights that the plume search and maintaining can be very time consuming at times.

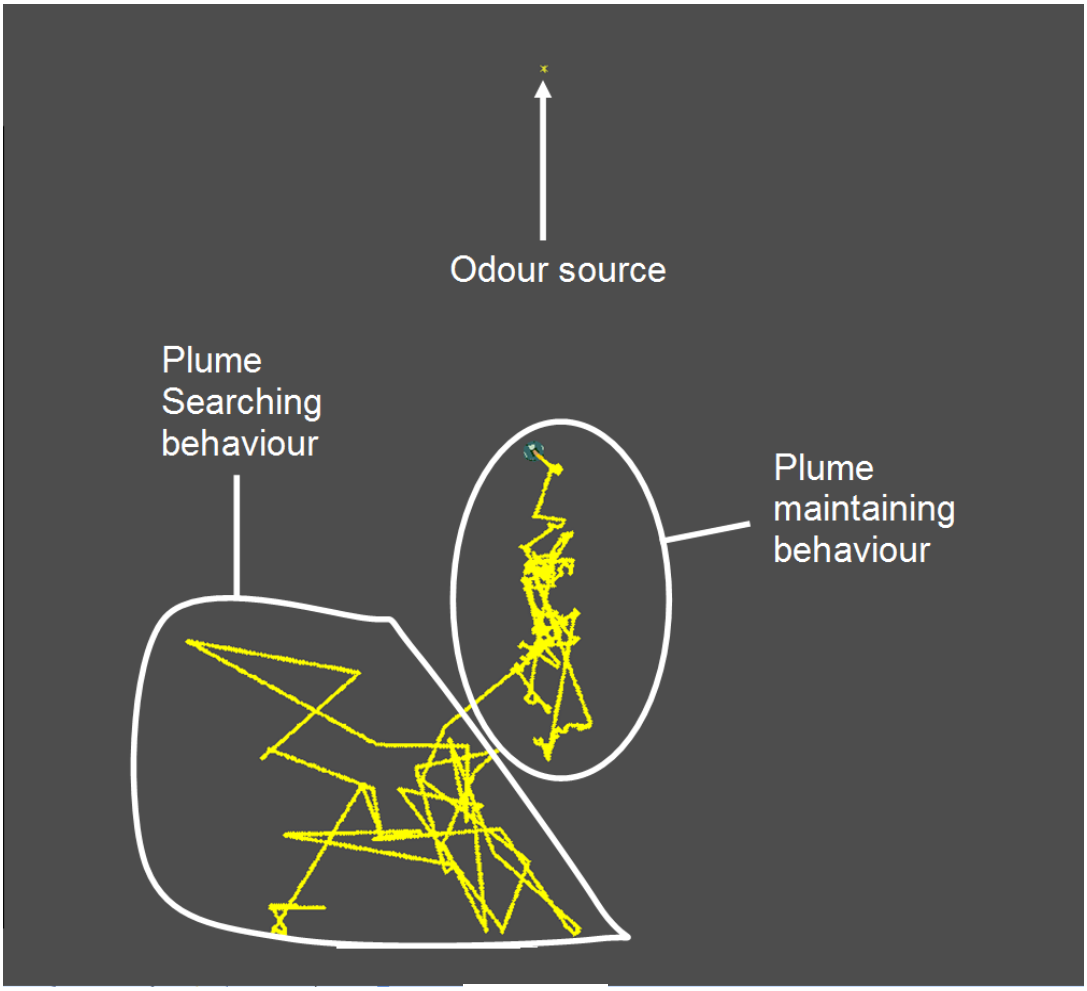


Fig 28

2. The robot will often travel back and forth, following a straight path, within the time averaged plume, without reaching the source location.

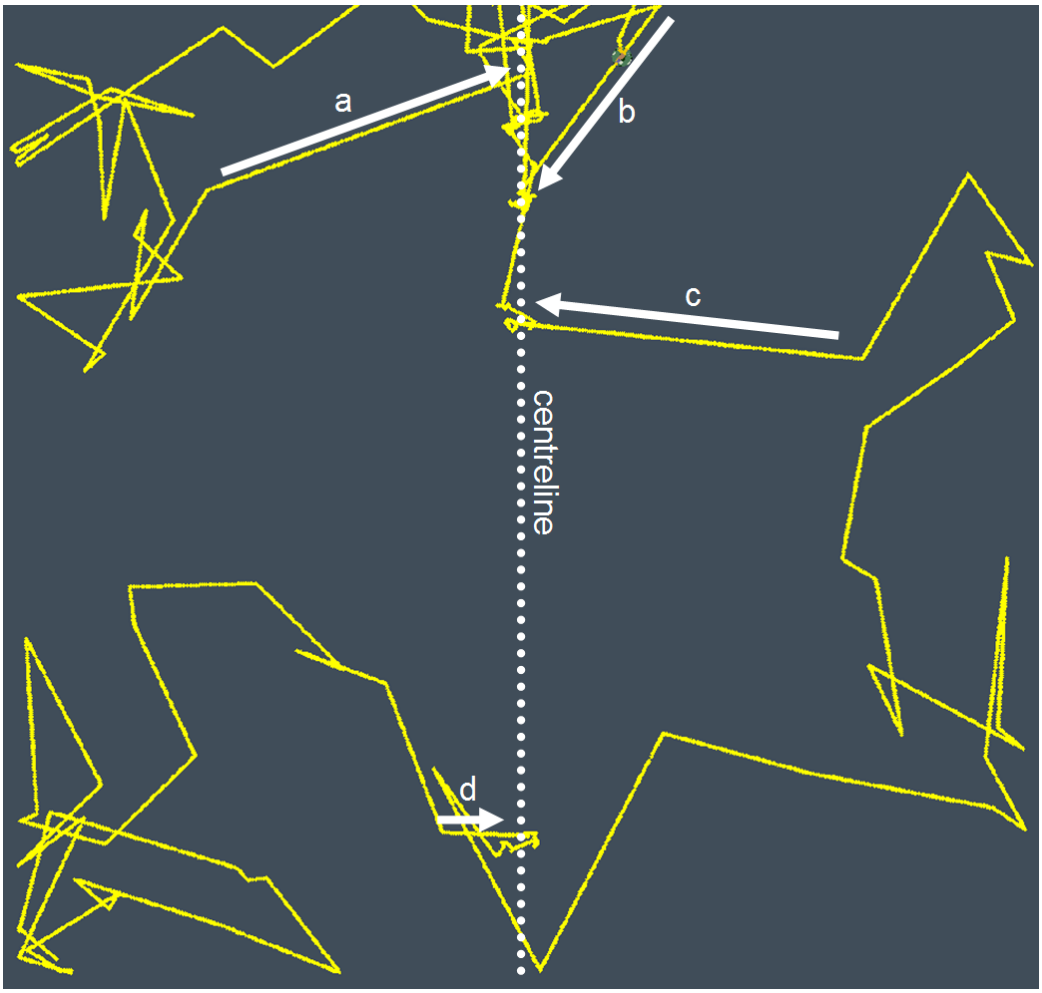


Fig 29

It also seems to be attracted to the centreline of the plume, as arrows a, b, c, and d (Fig 29) demonstrate.

3. Obstacles are well handled, in a Bug1 fashion, and actually help the robot progressing towards the goal. The robot will go round obstacles (Fig 30) only when in plume maintaining behaviour; during the random walk, they are only avoided. It will often happened that the same obstacle is encountered, and dealt with many times.

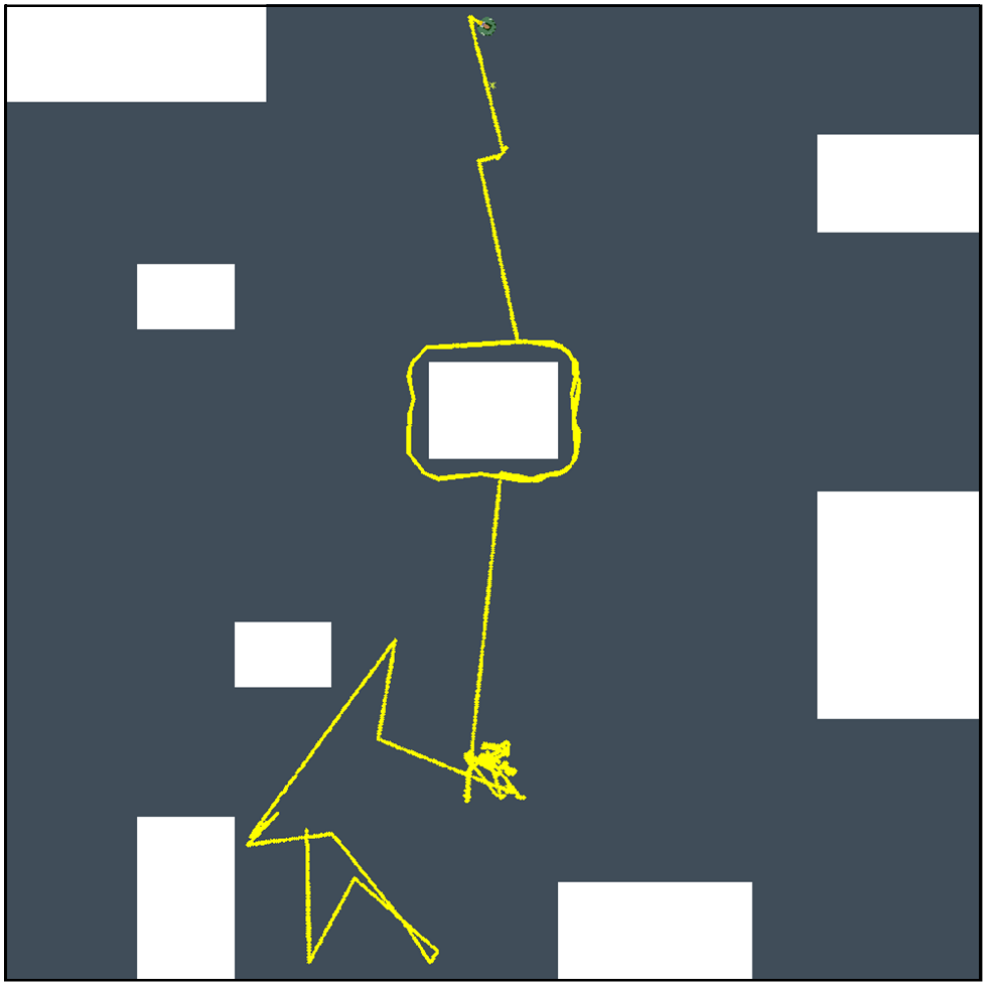


Fig 30

4. Whether stable or unstable conditions are selected doesn't seem to affect the outcome of the simulation. However, as the plume is spread over a greater area in unstable condition settings, the task of finding it using random walk is a less time consuming task. The algorithm does not comes up with a final landmine location. However, after a varying time, it can be observed that the robot settles around the highest concentration location of the plume.

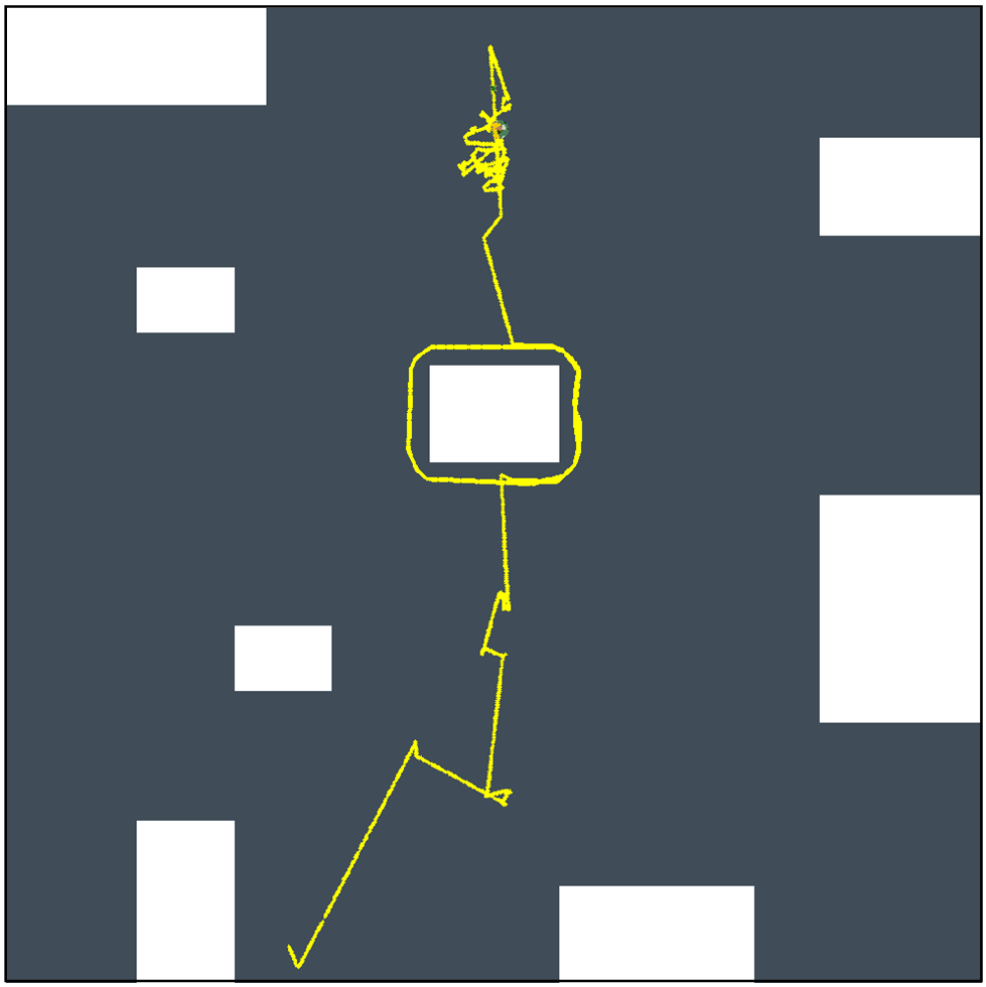


Fig 31

The performance of the E.Coli-Bug1 cannot be easily assessed and quantified, given that no results concerning the location of the source is returned. Its performance would be further improved if a means of stopping the search and estimate the landmine location was added, and if records of the minefield areas and obstacles already covered were kept.

7.6/ Discussion

Bacteria find the source of the chemicals by consistently moving towards the highest concentration. The chemotaxis strategy implemented in the robot, is eventually achieving to guide it towards the goal. Most of the results obtained during the numerous simulations find their explanation in the particular structure of the Gaussian plume.

It happens that the robot proceeds smoothly and in a straight line towards the goal if located precisely on the centreline and heading 90 degrees towards the source. This behaviour is caused by the structure of the signal (Fig 31) provided to the robot as sensorial information. The centreline of the Gaussian plume offers a seamless path to the source as each new value read will be higher than the previous one.

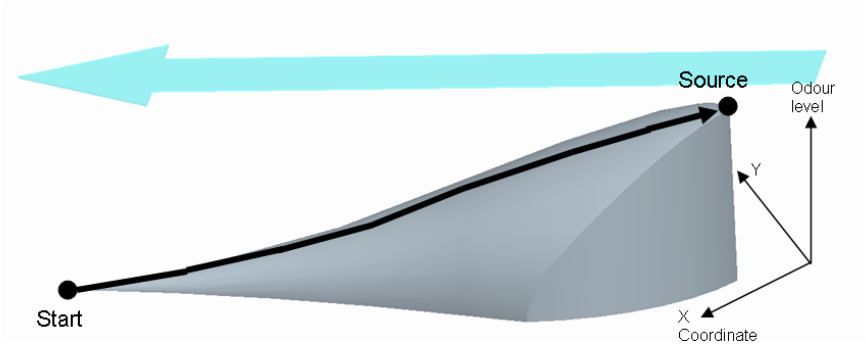


Fig 31

Another particularity of the Gaussian plume is that following the highest gradient doesn't necessarily lead to the goal, but rather to the centreline (Fig 32), the source being the most upwind part of it.

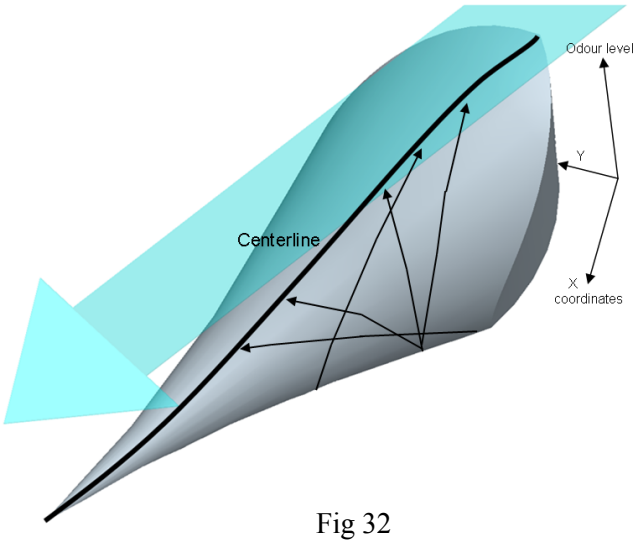


Fig 32

This is probably the main reason behind the erratic behaviour of the robot, and the explanation of why progresses towards the goal are so slow once the centreline of the plume has been reached: only one direction (90 degrees) will provide an higher value.

If the E.Coli algorithm behaviour is implemented in a real robot, it will take even longer than the simulation results suggest: the Gaussian plume model provides already time averaged chemical levels, and obtaining them in actual minefield conditions will require the robot to stay stationary at the same spot for a non negligible duration of time whenever a reading will be taken.

This algorithm suffers from another serious drawback. It is totally unable to declare the source of the chemical leak. The characteristics of the source are not known, nor is the highest possible concentration that could be sensed: the robot is forever looking for the source because unlike real bacterium in search for food, there is no physical goal to signal the robot that the search should be called off. Being hidden, the mine will never provide such an information, and the E.Coli-Bug1 algorithm guiding a mine sniffer would probably need the supervision and judgment of qualified personnel to estimate the landmines location.

8.1/ Signal modelling (Statistical model)

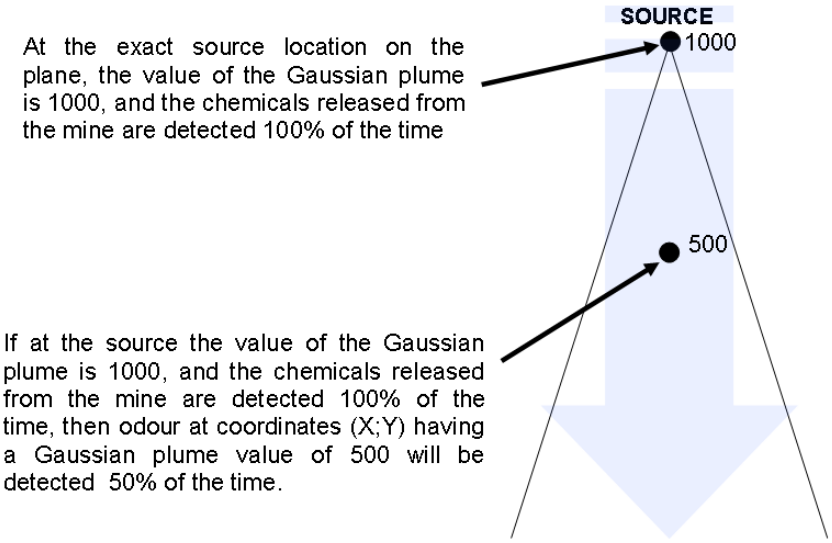
The process of odour dispersion in a fluid flow is by nature a dynamic process. A signal structure similar to the one used previously for the chemotaxis strategy will be generated using probability theory. The Gaussian plume will be used to evaluate the quantitative values of the chemicals leaked from the mine. The purpose of the following subroutine will be to add a time dimension to the odour dispersion model, and make it closer to the real conditions faced on the minefield.

In a Gaussian plume, every position on the plane representing the robot's environment is given an average value of chemicals over time. In order to make a realistic simulation, this value must be varying with time, and amount eventually to the very same Gaussian average value .

In an actual minefield, the source's geographical location is the place where the highest concentration levels of chemicals are likely to be sensed, and also where the chemicals are detected the most often. The anemotaxis strategy does not require concentration of the chemical of interest as the information received by the robot is treated as binary: the odour is either sensed or not.

Let's note that chemicals being released in bulk flow, the scent from the mine cannot be detected 100% of the time. Depending of the characteristics of the source, the chemicals discharge rate - percentage of time the chemicals can be sensed - will be determined. Then, a statistical comparison will be made between every other plane position (X;Y) within the Gaussian plume, the goal average odour value, and its detection percentage. Those parameters will be used as inputs to determine the probability that the odour plume can de sensed at that (X;Y) position:

$$\text{Odour detection rate at coordinates} = \text{Odour detection rate at source} \; X \; \frac{\text{Gaussian plume value at coordinates}}{\text{Gaussian plume value at source}} \; X \; 100$$



A random variable value between 0 and 100 will be generated using the MATLAB "rand" function to which the detection percentage at the position will be compared. If the value generated is lower than the percentage calculated, the function returns "0" to the robot, signalling that no chemicals are detected at that moment in time. However, if the value generated is greater or equal than the percentage calculated, the function returns "1" to the robot, signalling that chemicals reading above the sensor threshold are being detected.

As a result, the robot will be provided at all times with an odour level refreshed every second comprised between the maximum possible concentration, and a 0 value corresponding to all readings below the threshold value of the sensor.

When the robot stays stationary at the same position, the average value sensed is conform to the Gaussian plume output at the corresponding (X;Y) position. Furthermore, the closer the robot will get to the source, the higher the frequency of the "1" will be. The maximum odour detection frequency can only be obtained at the geographical location of the source, and the further away from the source, the scarcer the "1" becomes.

Being based on the Gaussian odour plume formula, the chemical dispersion simulation can be adapted to many meteorological conditions, with realistic effect: with the most unstable settings, the Gaussian plume values are in general lower than with stable settings, and the frequency of "1", the frequency at which chemicals are detected, much lower.

The resulting odour dispersion simulation can be assimilated to a Poisson distribution[18]:

$$f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Where:

- e is the base of the natural logarithm ($e = 2.71828...$)
- k is the number of occurrences of an event - the probability of which is given by the function
- $k!$ is the factorial of k
- λ is a positive real number, equal to the expected number of occurrences that occur during the given interval.

In statistics theory, the Poisson distribution expresses the probability of a number of events occurring in a well defined period of time if their average rate of occurrence is known and independent of the time since the last event.

8.2/ Obstacle handling

The obstacle handling during the plume maintaining procedure will be inspired by the Tangent bug algorithm. When an obstacle is encountered, its boundary is followed until the robot faces the wind again.

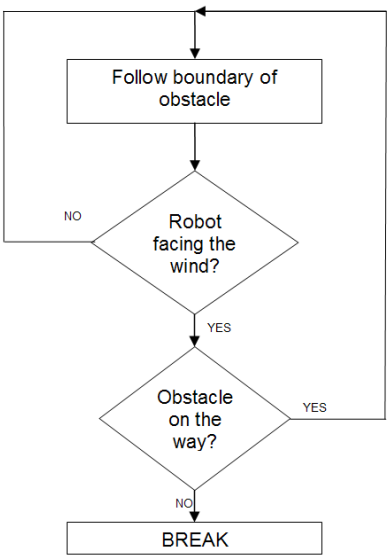


Fig 33: Moth-Tangent

8.3/ Algorithm

Information needed:

- Chemical concentration
- Wind direction

Plume search / maintaining behaviours:

The plume search and plume maintaining behaviours will be undertaken by the same onboard "Cast" function, as no chemicals leaked from the mine are sensed in both cases.

The casting behaviour of the robot will be based on the linear search principle familiar with insects. When no chemicals are detected and no major wind shift has occurred, the robot will move crosswind with increasing amplitude (Cast) until contact is made with the chemical trail. Upon odour detection, the algorithm will exit the casting process and "Surge": the robot will face the wind and proceed forward for a certain distance (Step).

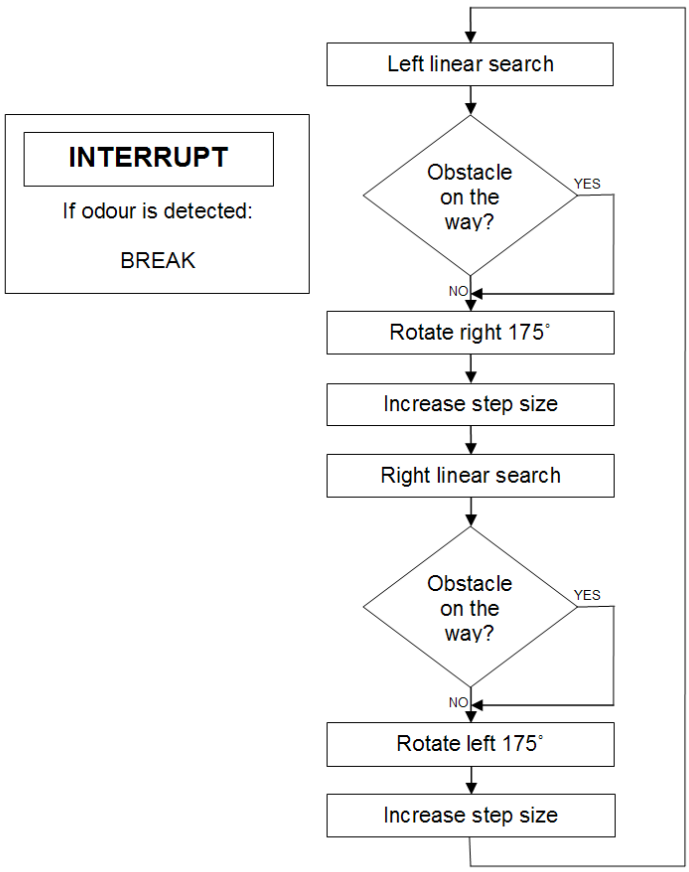


Fig 34: Moth-Cast

Source declaration (Estimation of source location.)

Male moths use their sense of smell a chemotactic search strategy to get close to its source. Other sensorial elements such as sight or touch are used to finalise the approach and confirm that the source of the odour has been reached. Such information will not be available, so the robot will use multiple passes to have a better estimation of the landmine position. Every time the upwind limit of the delimited area has been reached, the robot will travel downwind, and restart the search, remembering the most upwind position the odour has been detected. For more precise mine location estimation, this process could be repeated a number of times. After the end of the procedure, the most upwind position the odour has ever been detected will be declared as the source, and the distance from the actual source location will be returned,

indicating the precision of the result. For our trials however, only one pass will be sufficient to assess the algorithm efficiency.

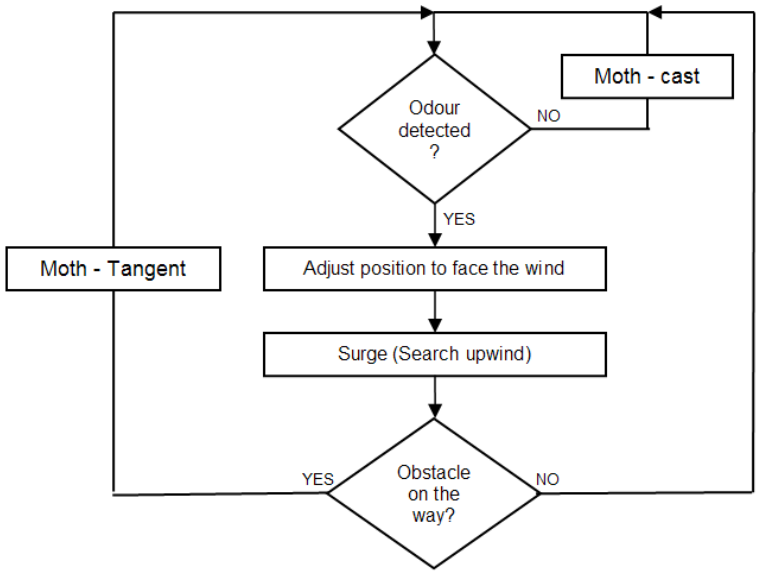


Fig 35: Moth-Tangent algorithm

General principles of the algorithm:

It is assumed that the robot will receive sensorial information about the odour level at all times. Upon start-up, the robot will search upwind for the source if the sensed chemical levels are nonzero, or immediately switch to casting behaviour otherwise. When the upwind minefield limit is reached, the robot will travel downwind and repeat the search pattern. The most upwind position the odour has ever been detected will be considered as the estimated source.

8.4/ Results

The anemotaxis search simulation produces satisfying quantifiable results. The robot will get to the approximate location of the landmine and return an estimated source location in a much shorter time than the E.Coli-Bug1 association (Even though numerous passes are necessary for a precise result).

However the precision and repeatability are greatly affected by the atmospheric conditions settings. The repeatability of the results produced by the Moth-tangent algorithm will be much higher under stable conditions than with unstable condition setups.

Analysis procedure

Accuracy and repeatability of the experimental results have to be analysed.

The returned estimated position of the source of the odour during the different algorithm runs represent the "real thing" that will be used to assess the accuracy, precision, and in general the efficiency of the "Moth-tangent Algorithm" in fulfilling its task.

Accuracy for a single measurement is defined as the difference between a measured value and the desired value (Δf). In our case, Δf will represent the distance between the source estimation position and the real landmine coordinates. In order to quantify it, the root mean square of the different Δf of an experiment will be computed, giving a numerical value in centimetres, that could be turned into a percentage if necessary.

$$\Delta f = \sqrt{(\text{Estimated X} - \text{Landmine X})^2 + (\text{Estimated Y} - \text{Landmine Y})^2}$$

Precision, also called repeatability, is a measure of consistency in the ability to reproduce a reading. If an experiment is repeated a number of times, there are variations in the readings, and the less variations, the more precise the robot is in locating the mine location. The standard deviation σ will be calculated, and will provide an useful value regarding precision depending on preset conditions .

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

where σ = measure of dispersion of the data about mean
N = number of attempts
x_i = each distance difference
$\bar{x}$ = average distance difference

- **Stable conditions:**

Settings: 1 Gram / sec release rate, stable conditions, no obstacles

Under stable conditions, a source with an high release rate of chemicals should be easily detected and. tracked by the robot.

Five simulation runs are executed using the stable settings, which emulates a narrow plume.

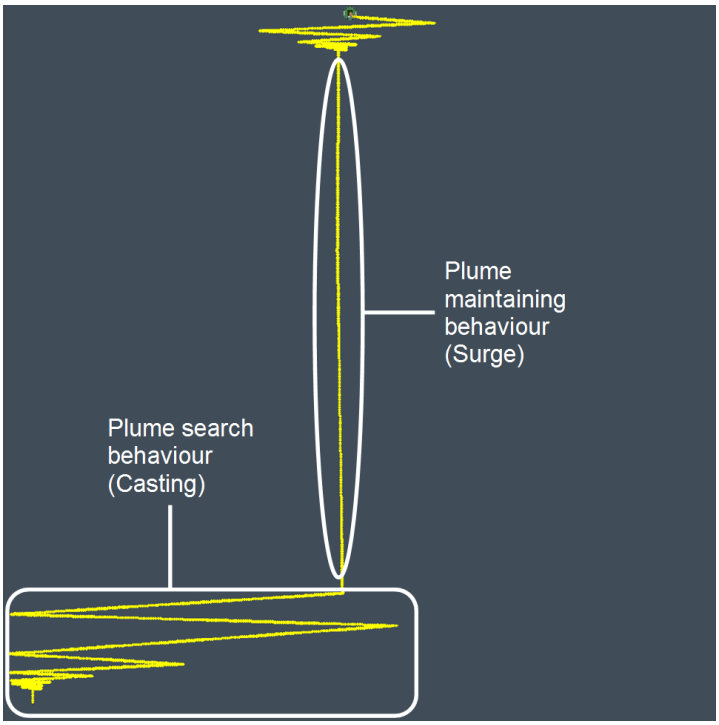


Fig 36: Typical algorithm response (Stable)

	Time taken	Estimated X coordinates	Estimated Y coordinates	Distance difference(Δf)	Variation from mean
Run 1	8:37	250	1465	35 cm	-5.4
Run 2	10:00	251	1524	24 cm	-16.4
Run 3	14:43	264	1438	64 cm	23.6
Run 4	7:42	250	1479	21 cm	-19.4
Run 5	10:09	250	1558	58 cm	17.6

The algorithm will take an average of 10 minutes and 14 seconds to return a result.

Over five runs, the location of the chemical source will be estimated at an average, of 40.4 cm (Accuracy) away from the actual landmine location, with a standard deviation *σ of 17.557 cm.*

The plume centreline and its immediate surroundings represents in general the areas where the odour has the most chances of being detected. The robot will begin the "Casting" plume search behaviour, and, upon detection under stable settings, the robot will travel upwind and return a very accurate estimated mine location. Good repeatability, or precision is observed over multiple runs.

Settings: 1 Gram / sec release rate, stable conditions, obstacles

As expected, the reaction to the obstacles is different depending on the behaviour the robot is in. When the odour is being tracked, a "Tangent bug" is executed , but when the robot is searching o tries to reacquire the plume, it just casts in the other direction.

When obstacles are introduced, the motion of the robot is restricted when it enters the plume search behaviour. The robot is also more likely to be interrupted in its "surge" behaviour, and is forced to revert to the time and resources consuming "casting".

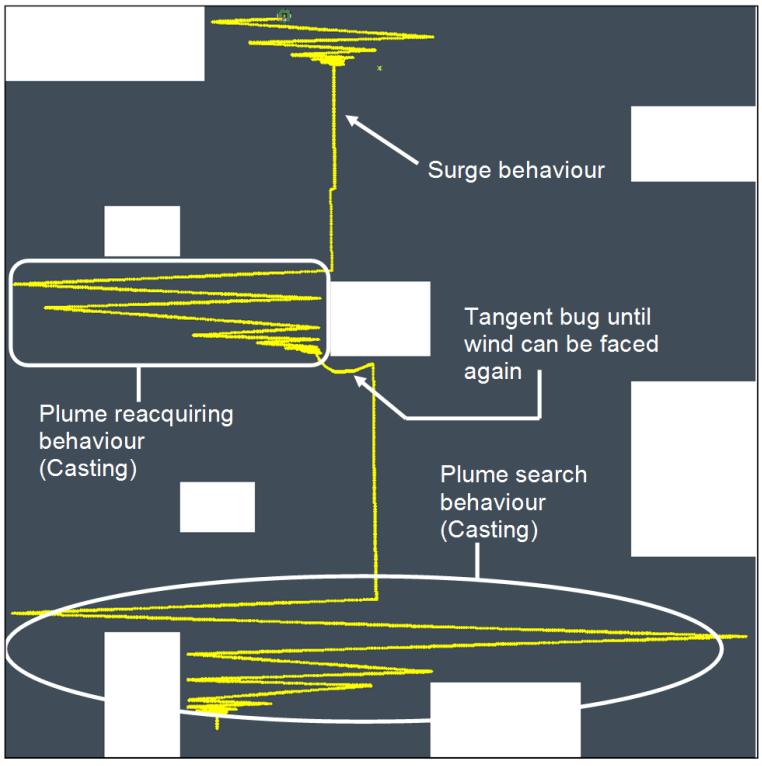


Fig 37: Typical algorithm response (Stable)

	Time taken	Estimated X coordinates	Estimated Y coordinates	Distance difference(Δf)	Variation from mean
Run 1	13:50	298	2336	171 cm	35.8
Run 2	16:49	257	1635	135 cm	-0.2
Run 3	14:43	264	1438	64 cm	-71.8
Run 4	14:39	294	1317	188 cm	52.8
Run 5	13:30	265	1617	118cm	-17.2

The algorithm will take an average of 14 minutes and 42 seconds to return a result. This is significantly longer than the average time recorded without obstacles in this specific environment.

Over five runs, the location of the chemical source will be estimated at an average, or accuracy, of 132.2 cm away from the actual landmine location, with a standard deviation σ of 43.636 cm.

- **Instable weather:**

1 Milligram / sec release rate, unstable conditions, no obstacles

Instable conditions are characterised by changing wind directions (wind-shifts), creating a turbulent fluid flow that spreads . In theory, the odour particles emitted from the landmine should spread over a greater area over time, making them more difficult to detect.

Many simulations runs show that the robot stays in casting mode for longer, and the first contact with the plume is made more upwind than in stable conditions. Furthermore, once acquired, contact with the plume, should not be taken for granted: the odour should be highly intermittent, forcing many "plume reacquiring" procedure.

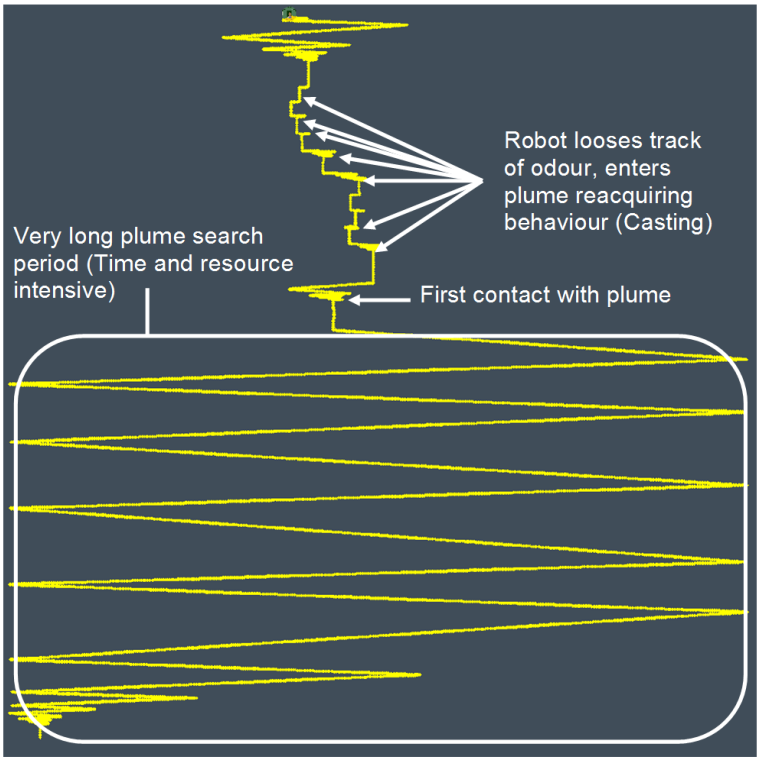


Fig 38: Typical algorithm response (Unstable)

	Time taken	Estimated X coordinates	Estimated Y coordinates	Distance difference(Δf)	Variation from mean
Run 1	24:46	265	1517	23 cm	-69
Run 2	23:13	265	1501	15cm	-77
Run 3	23:54	274	1217	284 cm	192
Run 4	20.57	294	1511	45 cm	-47
Run 5	21:20	309	1428	93 cm	1

Average time taken: 22:50 min ; Mean = 92 cm ; σ = 99.763 cm

The algorithm will take an average of 22 minutes and 50 seconds to return a result. As expected, the completion times of the simulations are higher because the robot has to cast longer to establish contact with the plume, and often reverts to casting again for plume reacquiring reasons. Over five runs, the location of the chemical source will be estimated at an average, or accuracy, of 92 cm away from the actual landmine location, with a standard deviation σ of 99.763 cm.

Because of the odour tracking difficulties, the estimated landmine position often ends up being far away from the actual explosive device.

1 Milligram / sec release rate, unstable conditions, obstacles

Adding obstacles to instable conditions do not change much to the outcome of the different simulations completion time.

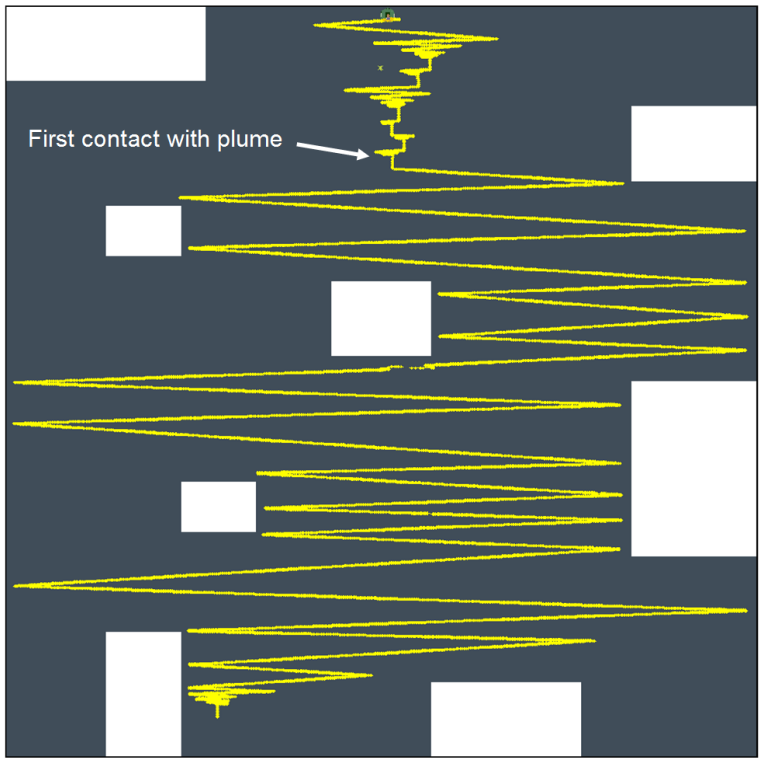


Fig 39: Typical algorithm response (Unstable)

Run 1	Time taken	Estimated X coordinates	Estimated Y coordinates	Distance difference(Δf)	Variation from mean
Run 2	24:08	809	1542	561 cm	353
Run 3	23:38	280	1446	62cm	-146
Run 4	19:55	292	2473.	50 cm	-158
Run 5	25:49	255	1697	197cm	-11
Run 1	19:41	306	1340	170cm	-38

Average time taken: 22:38; Mean = 208 cm; σ = 185.717 cm

The algorithm will take an average of 22 minutes and 38 seconds to return a result. But as the robot is prevented by obstacles to cast in the area covered by the odour, first contact with the plume occurs later.

However, the accuracy and precision of the results, already bad compared to stable weather conditions, get even worse. Over five runs, the location of the chemical source will be estimated at an average, or accuracy, of 208 cm away from the actual landmine location, with a standard deviation σ *of* 185.717 cm.

8.5/ Discussion

The "stable" Gaussian plume settings allows the simulation of a chemical plume very close in structure (narrow) to an instantaneous real time plume, in which the frequency of odour occurrence increases as the robot gets closer to the source. The Male moth algorithm is successful in declaring a source location very close to the actual position of the landmine.

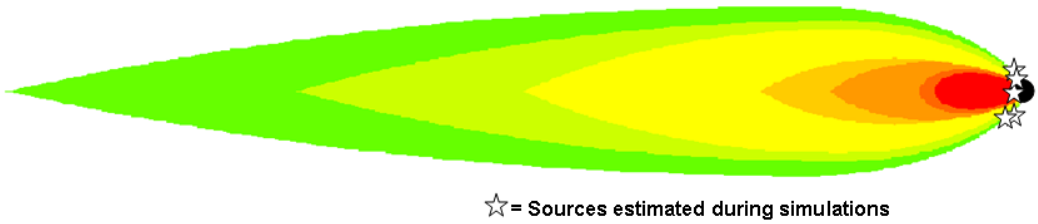


Fig 41: Gaussian plume, stable settings

"Unstable" conditions settings imply quite important wind shifts (Changes in wind direction). When used in the MATLAB simulation, those extreme conditions will cause a wider plume, due to the lateral spread of the chemical particles.

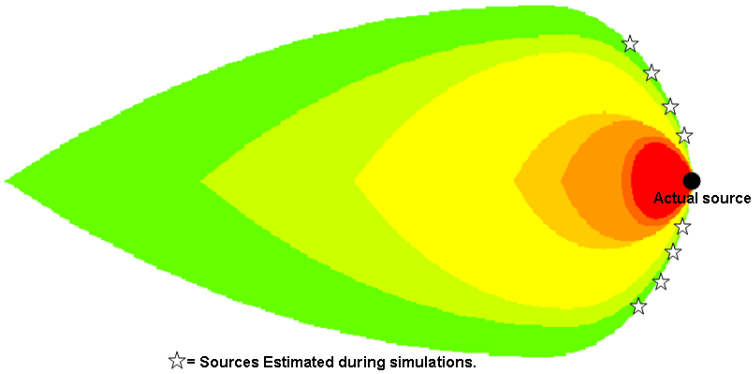


Fig 42: Gaussian plume, unstable settings

The shape of the plume has an incidence over the KIKS simulation. The plume being too wide, and the wind shifts ignored, the algorithm often ends up declaring a source location far off the actual position of the landmine. At the end of a simulation, the declared source can be anywhere along the upwind boundary of the chemical plume.

This latest observation highlights the limit of the odour model. Being based on a Gaussian plume, the concentration levels turned into odour detection rate provide only half the information needed by the anemotaxis algorithm. The changes in the wind direction, crucial for a realistic simulation, are not included, and influence the result. It can also be said that the wind simulation is based on a subjective interpretation of the odour dispersion not relying in any proven realistic quantitative values. Although giving a very good idea of the mechanisms behind the anemotaxis search, success of the algorithm in the KIKS simulator cannot guaranty success in real minefield situations.

9.1/ Successes achieved during the project

The two only ways of finding the mine are based on the two biomimetic methods that are chemotaxis and anemotaxis. Both strategies algorithm have been successfully programmed and tested onto the KIKS robot simulation software. The male moth algorithm based on the anemotaxis approach, having by far the best performance, appears to be most adapted to the mine sniffing problem.

The use of the principles behind the bugs have been successfully used and helped the algorithms handle the different simulated obstacles with satisfying results.

During the project, different odour dispersion models, with increasing realism have been coded, appropriate solutions found, and their understanding has greatly contributed in improving the overall efficiency of the mine sniffer robot.

Chemical leaks from the mines, allied with adequate and precise enough sensors to detect them, are the necessary prerequisites for any scent following algorithm to work. The robot will be totally inefficient if the mines have been freshly planted or designed/modified to avoid leaks. In the simulator, realistic parameters such as the sensor's sensitivity, resolution or threshold, as well as the source's chemicals release rate can be handled, helping produce algorithms potentially successful in an actual minefield.

9.2/ Problems encountered

The Gaussian plume represents a good image of the average concentration of odour over a given area. However, its structure is quite different (too wide) than an instantaneous plume, and information concerning the real time change in wind directions cannot be provided to the robot, as they are being lost in the averaging.

Logically, in a minefield, more than a single mine is likely to be found. It opens the possibility of different mines scents interfering with each other. If the anemotaxis strategy alone is used, only the most upwind exploding device will be found, the others will be missed, or ignored: the robot will not be able to look for another as long as the first one, and all the chemical contaminated soil has not been disposed of.

During the whole duration of the report, the characteristics of the odour sensor as well as those of the landmines have been completely overlooked. No such information is widely available and as a result, some doubts might arise regarding the computer simulated results.

9.3/ Future recommendations

Odour dispersion simulation

Information given to the robot is crucial, landmines differring considerably depending on the type and quantity of explosives used and the nature of the soil they are buried in. When the characteristics of the emission source are unknown, most of the parameters in the Gaussian equation are undetermined. For a good and accurate simulation, the exact nature, quantity and discharge rate of the chemicals leaked by the mines must be known and accounted for.

The E.Coli algorithm is slow, and returns no precise result. On the contrary, the theoretical bug algorithms are very successful in reaching the goal because they are given very accurate information regarding the direction and travel distance. The more information types are given to the robot, the more efficient it will be in finding the source. Mines must be further studied, and new clues about their locations devised so the mine detection process can be facilitated.

Odour sensor parameters

Sensors parameters determine whether the scent from the mine can be detected and are critical to simulate any prospective mine sniffing algorithm. Sensors in general can be entirely described by the smallest value they can detect (Threshold), the smallest change in input they can measure (Resolution), and finally its sensitivity to monitored quantity (Response time).

The means to control threshold, sensitivity an response time have successfully been coded into KIKS, but never used to evaluate the algorithms. One of the recommendations for future work would be to study further the characteristics and nature of the chemicals emitted by the mines and the sensor ability to detect them so the realism of the mine sniffing simulator can be improved.

9.4/ Closing comments

The title of this project is deceiving. Coming up with a scent following algorithm is merely the '*tip of the iceberg'* given that its efficiency and performance has to be proven. As a result, most time and energy for the duration of this project have been spread over peripheral matters such as the environment (The signal model and the odour sensor) and the robot simulator (Wheel encoders, different proximity and position sensors) .

For future work in this domain, I would advise making a special purpose computer simulator able to support concurrently both the robot behaviour and a odour dispersion simulation (Preferably based on fluid dynamics) BEFORE starting work on developing a mine sniffing robot's algorithm. It will allow for a realistic development environment free of real life robot constraints (E.g.: sensors, wheel slip), and ease the algorithm's development process.

10/ References

[1] Chignell. R, (7-9 Oct 1996) **, The Detection of Abandoned Land Mines: A Humanitarian Imperative Seeking a Technical Solution, EUREL International Conference No. 431,** (103-108)

[2] Carter. L, (2001) , **Landmine detection using stimulated infrared imaging,** Fifth International Symposium on Technology and Mine Problem, IGARSS '01. IEEE 2001 International volume 3, (1110,1112)

[3] Naeem. W et al, (MAY 2007)**, Chemical Plume Tracing and Odour Source Localisation by Autonomous Vehicles,** THE JOURNAL OF NAVIGATION VOL. 60, NO. 2 , (173–190)

[4] **Gaussian plume model**: http://san.hufs.ac.kr/~gwlee/session9/gaussian.html

[5] Althoefer. K ,(January 2006), **Lecture Notes Part B1**, Robotics/Robotics Systems, (2-15)

[6] Swere. E et al, (2003), **Robot Navigation Using Decision Trees**, ELECTRONIC SYSTEMS AND CONTROL DIVISION RESEARCH, (15-17),

[7] Smart .W et al, (2002),**Effective Reinforcement Learning for Mobile Robots, Robotics and Automation,** ICRA '02. IEEE International conference Volume 4, (3404- 3410)

[8] **Wikipedia page on diffusion** :http://en.wikipedia.org/wiki/Diffusion

[9] Ishida .H et al, (April 2001), **Plume-Tracking Robots: A New Application of Chemical Sensors** Biological Bulletin 200**, (** 222–226)

[10] **Wikipedia page on Taxis**: *http://en.wikipedia.org/wiki/Taxis*

[11] Russell. A et al, (2003), **A comparison of reactive robot chemotaxis algorithms,** Robotics and Autonomous Systems 45 , (83–97)

[12] Harvey. D el al, (February 2008) **Effectiveness of Insect-Inspired Chemical Plume-Tracking Algorithms in a Shifting Wind Field,** IEEE TRANSACTIONS ON ROBOTICS NO. 1, Volume 24, (196, 201)

[13] Ishida. H et al, (1998), **Remote sensing of gas/odour source location and concentration distribution using mobile system,** Sensors and Actuators B 49 , (52–57)

[14] **Wikipedia page on** *Moths: http://en.wikipedia.org/wiki/Moth*

[15] **Wikipedia page on** *Chemotaxis: http://en.wikipedia.org/wiki/Chemotaxis*

[16] **Wikipedia page on** *Lobster: http://en.wikipedia.org/wiki/Lobster*

[17] Dhariwal. A et al, (New Orleans , April 2004), **Bacterium-inspired Robots for Environmental Monitoring,** International Conference on Robotics & Automation, (1436-1443)

[18] **Wikipedia page on Poisson distribution**:
http://en.wikipedia.org/wiki/Poisson_distribution

Other references:

[18] **Lateral and Vertical Dispersion Coefficients**:
http://san.hufs.ac.kr/~gwlee/session9/images/dispeqs.gif

[19] Islam. M, (5 January 1998), **Application of a Gaussian plume model to determine the location of an unknown emission source, "**Water, Air, and Soil Pollution" 112**, (**241-245)

[20] Farrell. J et al, (22-26 Sept. 2003), **Chemical Plume Tracing Experimental Results with a REMUS AUV,** OCEANS 2003 Proceedings, volume 2, (962 - 968)

1. Gaussian plume

Reference : http://san.hufs.ac.kr/~gwlee/session9/images/dispeqs.gif

1.1 Gaussian Plume Formula

$$C(x,y,z)=\frac{Q}{2*\pi*u*\sigma y*\sigma z}*\exp\frac{-y^2}{2*\sigma y^2})*(\exp(\frac{-(z-h)^2}{2*\sigma z^2})+\exp(\frac{-(z+h)^2}{2*\sigma z^2}))$$

where:

1. *C is the concentration of the emission (in micrograms per cubic meter) at any coordinates point x meters downwind of the source, y meters laterally from the centreline of the plume, and z meters above ground level.*
2. *Q is the quantity or mass of the emission (in grams per seconds)*
3. *u is the wind speed (in meters per second)*
4. *H is the height of the source above ground level (in meters)*
5. σy *and* σz *are the standard deviations of a statistically normal plume in the lateral and vertical dimensions, respectively .*

1.2 Lateral and Vertical Dispersion Coefficients:

- σy *lateral dispersion*

$$\sigma y = x*\alpha*\frac{1}{\sqrt{1+0.0001*x}}$$

For very unstable conditions: α=0.22

For moderately unstable conditions: α=0.16

For slightly unstable conditions: α=0.16

For neutral conditions: α=0.08

For somewhat stable conditions: α=0.06

For stable conditions: α=0.04

- σz *vertical dispersion*

*For very unstable conditions: σz= x*0.20*

*For moderately unstable conditions: σz=x*0.12*

For slightly unstable conditions.

$$\sigma z = x * 0.08 * \frac{1}{\sqrt{1 + 0.0002 * x}}$$

For neutral conditions

$$\sigma z = x * 0.06 * \frac{1}{\sqrt{1 + 0.00015 * x}}$$

For somewhat stable conditions

$$\sigma z = x * 0.03 * \frac{1}{1 + 0.0003 * x}$$

For stable conditions

$$\sigma z = x * 0.016 * \frac{1}{1 + 0.0003 * x}$$

2. Gaussian plume setup code

```
function [ Concentration, Wind_direction ] = Plume_setup( x, y )
%PLUME_SETUP Summary of this function goes here
%   This function acts as a look up table of gradients for the robot.
%
%--------------------------------------------------------------------
% In order for the robot to read the right gradiens ,the values sent to
% the function have to be offset.
% The coordinates x and y are received from main  funtion in millimeters,
% so need to be converted in meters, hence the "/1000"
x = (3000-x)/10;
y = (y-1500)/10;
z = 0.1; % Sensor on robot located 10 cm above the ground
%--------------------------------------------------------------------

%--------------------------------------------------------------------
%%%%%%%%                Source parameters     %%%%%%%%%%%%%%%%%
%--------------------------------------------------------------------

%Quantity or mass of the emission (grams / seconds)
Q =1 ;
%Wind speed (meters / second)
U = 5;
%Wind direction ;
Wind_direction = 90 ;
%Height of the source above ground level ( meters )
h = 0;
%--------------------------------------------------------------------

%--------------------------------------------------------------------
%%%%%%%%                Wind settings          %%%%%%%%%%%%%%%%%
%        Select one block depending on chosen conditions       %
%--------------------------------------------------------------------

% %Unstable conditions:
% a=0.22;
% dev_z= x*0.20;

% %Moderately unstable conditions:
% a=0.16;
% dev_z=x*0.12;

% %Slightly unstable conditions:
% a=0.16;
% dev_z = x*0.08*(1/sqrt(1+0.0002*x));

% %Neutral conditions:
% a=0.08;
% dev_z = x*0.06*(1/sqrt(1+0.00015*x));

% %Somewhat stable conditions:
% a=0.06;
% dev_z = x*0.03*(1/sqrt(1+0.0003*x));

%Stable conditions:
a=0.04;
dev_z = x*0.016*(1/sqrt(1+0.0003*x));

%Lateral dispersion standard deviation calculations%
dev_y = x*a*(1/sqrt(1+0.0001*x));

%--------------------------------------------------------------------

%--------------------------------------------------------------------
%Concentration calculation formula (Gausian plume model formula)
Concentration = (Q/2*pi*U*dev_y*dev_z) * exp(-y^2/(2*dev_y^2)) * (exp(-(z-
h)^2/(2*dev_z^2))+exp(-(z+h)^2/(2*dev_z^2)));
%--------------------------------------------------------------------
```

3. Common subroutines

3.1 Right boundary follower code

```
function Right_boundary_follower( port,time,ref )

global  Wind_direction

 Get_reading(ref ) ;
while (kiks_ktime(port)<time)

reflex = kProximity(ref);

 pos =kBattle_sensors(ref);
    % Angle robot is facing.
     angle = pos(3);

    if   (angle > (Wind_direction-20))&&(angle <
(Wind_direction+20))&&((reflex(3) < 100) || (reflex(4) < 100))
        kSetSpeed(ref,0,0)% Stop
       Face_The_Wind(ref, port, time   )
        break
    elseif   (angle > ((Wind_direction+180)-20))&&(angle <
((Wind_direction+180)+20))&&((reflex(8) < 100) || (reflex(7) <
100))&&(reflex(1) > 100)
        kSetSpeed(ref,0,0)

        Face_The_Wind(ref, port, time  )
        break
    elseif ((reflex(3) < 500) && (((reflex(1)+reflex(2))/2) <
1000)&&(((reflex(1)+reflex(2))/2) > 300))
        kSetSpeed(ref,5,5)% Go straight

    elseif  ((reflex(2) > 900)&& (reflex(3) > 900) )
            while (reflex(3)>400)
                reflex = kProximity(ref);
                kSetSpeed(ref,5,-5)% Turn right
            end

     elseif  (reflex(3) > 400) || (reflex(2) > 400) || (reflex(4) > 400) ||
(reflex(5) > 400)
                  kSetSpeed(ref,5,-5)% Turn right


     elseif (((reflex(1)+reflex(2))/2) < 300)
        kSetSpeed(ref,-5,5)    % Turn left
    end;
      kiks_print(sprintf('Follows right the boundary of the obstacle'));

end;
end
```

3.2 *Left boundary follower code*

```
function Left_Boundary_follower( port,time,ref )

global  Wind_direction

Get_reading(ref ) ;
while (kiks_ktime(port)<time)

reflex = kProximity(ref);
 pos =kBattle_sensors(ref);
    % Angle robot is facing.
     angle = pos(3);

% To increase the chances of the robot heading for the highest gradient,
% better to stop within a range of angles (If angle of the robot is within
% a certain range)
     if   (angle > (Wind_direction-20))&&(angle <
(Wind_direction+20))&&((reflex(3) < 100) || (reflex(4) < 100))
        kSetSpeed(ref,0,0)

        Face_The_Wind(ref, port, time  );
        break

     elseif   (angle > ((Wind_direction+180)-20))&&(angle <
((Wind_direction+180)+20))&&((reflex(8) < 100) || (reflex(7) <
100))&&(reflex(6) > 100)
        kSetSpeed(ref,0,0)

        Face_The_Wind(ref, port, time  );
        break


      elseif ((reflex(3) < 500) && (((reflex(5)+reflex(6))/2) <
1000)&&(((reflex(5)+reflex(6))/2) > 300))
        kSetSpeed(ref,5,5)

      elseif  ((reflex(5) > 900)&& (reflex(4) > 900) )
            while (reflex(4)>400)
                reflex = kProximity(ref);
                kSetSpeed(ref,-5,5)
            end


       elseif  (reflex(4) > 400) || (reflex(5) > 400) || (reflex(3) > 400)
|| (reflex(2) > 400)
                  kSetSpeed(ref,-5,5)


       elseif (((reflex(5)+reflex(6))/2) < 300)
        kSetSpeed(ref,5,-5)
     end;
 kiks_print(sprintf('Follows left the boundary of the obstacle'));

end;
```

3.3 *Rotate robot to face a certain direction (Or goal)*

```
function[ ] = Head_towards_goal( ref,port,time )

global angle Heading_angle goal_x goal_y

   get_status(ref,goal_x,goal_y)

     while(kiks_ktime(port)<time)

    get_status(ref,goal_x,goal_y)

         if( angle ~= (Heading_angle))
            if (Heading_angle>angle)
                kSetSpeed(ref,-1,1)
            elseif(Heading_angle<angle)
                kSetSpeed(ref,1,-1)
            end
         elseif ( angle ~=
(Heading_angle))&&((reflex(3)>500)||(reflex(4)>500)||(reflex(2)>500)||(refle
x(5)>500))
             break
         else
                 kSetSpeed(ref,0,0)
         break
         end

end
```

4. E.Coli algorithm code

4.1 Main

```
function E_Coli_Algorithm(port,baud,time)

if nargin<3; time=inf; end;
if nargin<2; baud=9600; end;
if nargin<1; port=-1; end;

ref=kiks_kopen([port,baud,1]);
kFlush(ref)
kSetEncoders(ref,0,0);
[ Concentration ] = Get_reading(ref);


%Robot Reads odour value and prints it on screen
while(kiks_ktime(port)<time)
reflex = kProximity(ref);
   Old_value = Concentration ;
   [ Concentration ] = Get_reading(ref);

    if   (Concentration<0.0001)
        Random_Walk(port,ref,time)

    elseif
((reflex(3)>500)||(reflex(4)>500)||(reflex(2)>500)||(reflex(5)>500))

       Right_boun_follow(port,ref,time,Concentration)
   elseif (Concentration > Old_value)

   Go_Forward( port,time,ref );
   elseif   (Concentration <= Old_value)
        Chose_random_direction(ref, port, time)
        kSetSpeed(ref,10,10)


   end

end
kiks_kclose(ref);
end
```

4.2 "Random walk" behaviour code

```
function Random_Walk(port,ref,time)
%Walks the robot around, with random direction and turning angle, as long
%as the value sensed by the odour sensor is below a certain thresold.
%The maximum step size and the odour thresold can be adjusted.

kFlush(ref)
kSetEncoders(ref,0,0);
Max_Step_Size = 100000 ;%Shaft encoders pulses.


while(kiks_ktime(port)<time)

while(kiks_ktime(port)<time)

reflex = kProximity(ref);
[ Concentration ] = Get_reading(ref);

if (Concentration>10)
    break
end

Step_Size = Max_Step_Size*rand(1);
kSetSpeed(ref,10,10)
value = kGetEncoders(ref);
if ((value(1)>=Step_Size) ||
(value(2)>=Step_Size))||((reflex(3)>500)||(reflex(4)>500)||(reflex(2)>500)||
(reflex(5)>500))
kSetEncoders(ref,0,0);
    break

end

 kiks_print(sprintf('Concentration = 0, Random Walk'));


end

if (Concentration>0.0001)
    break

end

Chose_random_direction(ref, port, time)


end

end
```

4.3 "Choose Random direction" code

```
function   Chose_random_direction(ref, port, time)
% Chooses a random heading. It is simulating the "Tumble" property of the
% bacteriums.
%

%Heading angle calculations.
    Heading_angle = round(360*rand(1)) ;
    if (Heading_angle >= 359)
       Heading_angle = 0 ;
    end
while(kiks_ktime(port)<time)

  % Read the Battle Turret sensors
     pos =kBattle_sensors(ref);

    % Angle robot is facing.
     angle = pos(3);


if( angle ~= (Heading_angle))
            if (Heading_angle>angle)
                kSetSpeed(ref,-1,1)
            elseif(Heading_angle<angle)
                kSetSpeed(ref,1,-1)
            end
         elseif ( angle == (Heading_angle))
            kSetEncoders(ref,0,0);
             break
%          else
%                  kSetSpeed(ref,0,0)
%          break
end


end
```

4.4 “Go Forward”, “Surge” behaviour code

```
function [ ] = Go_FORWARD( ref,port,time  )
%Robot goes straight ahead until an event or an obstacle stops it
%

global  x_position y_position  reflex goal_x goal_y  minx maxx miny maxy

   get_status(ref,goal_x,goal_y)

while(kiks_ktime(port)<time)

 get_status(ref,goal_x,goal_y)

 kSetSpeed(ref,10,10)

        if
((reflex(3)>500)||(reflex(4)>500)||(reflex(2)>500)||(reflex(5)>500))
            break

%%%%%%%%%%If the goal is reached, stop.%%%%%%%%%%%%%%%%%%%
        elseif
((y_position>miny)&&(y_position<maxy))&&((x_position>minx)&&(x_position<maxx
))
                kSetSpeed(ref,0,0)
                kiks_print(sprintf('GOOOOOAL !!!!!!!!!!'))
                break
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        end
end
```

6. Gaussian plume instantaneous odour dispersion code (Wind simulation)

```
function Get_reading(ref)
% This function turns the odour level received from the plume setup nto a
binary signal, indicating to the robot that the signal is sensed or not. The
higher the odour level, the highest the the probability out of 100 the odour
is detected. The maximum odour level is located around the source area, and
has a probability of been sensed of 100 %. Detection rate will be calculated
from this base.

global ESTIMATED_SOURCE_X ESTIMATED_SOURCE_Y Odour_status Wind_direction x y

x = 250 ;
y = 1500 ;
D_R_Source = 1 ;
[ Concentration, Wind_direction ] = Plume_setup( x, y ) ;

Hundred_Percent_level = Concentration ;


     pos =kBattle_sensors(ref);
    % X position of robot (in mm, from left edge).
     x = pos(2);
    % Y position of robot (in mm, from top edge).
     y = pos(1);
    if  (x<250)
       Concentration = 0 ;
    else
       [ Concentration ] = Plume_setup( x, y ) ;
       Concentration = Concentration * D_R_Source ;
    end

if (rand(1 )< (Concentration / Hundred_Percent_level))
   Odour_status = 1 ;

if x < ESTIMATED_SOURCE_X ;
    ESTIMATED_SOURCE_X = x ;
    ESTIMATED_SOURCE_Y = y ;
end

else
   Odour_status = 0 ;
end


kiks_print(sprintf('Most upwind point odour has been detected: %d  %d
[%d]',ESTIMATED_SOURCE_X,   ESTIMATED_SOURCE_Y, Odour_status ));
```

6. Male moth-Tangent algorithm

6.1 Main

```
function Male_Moth_Algorithm(port,baud,time)
%Main Algorithm, modelled on the behaviour of the Male moth.
global ESTIMATED_SOURCE_X  ESTIMATED_SOURCE_Y Odour_status Passes
if nargin<3; time=inf; end;
if nargin<2; baud=9600; end;
if nargin<1; port=-1; end;

Passes = 1;
ref=kiks_kopen([port,baud,1]);
kFlush(ref)
kSetEncoders(ref,0,0);
Step_Size = 100 ;
pos =kBattle_sensors(ref);

    % X position of robot (in mm, from left edge).
    ESTIMATED_SOURCE_X = pos(2);
    % Y position of robot (in mm, from left edge).
     ESTIMATED_SOURCE_Y = pos(1);

%Robot goes forward every time odour is detected
while(kiks_ktime(port)<time)

Get_reading(ref);
reflex = kProximity(ref);
pos =kBattle_sensors(ref);
    % X position of robot (in mm, from left edge).
     x = pos(2);

if  (Odour_status == 1)
     Cast_Forward( ref, port, time, Step_Size) ;
elseif ((reflex(3)>500)||(reflex(4)>500)||(reflex(2)>500)||(reflex(5)>500))&& x > 55
kSetSpeed(ref,0,0)
Boundary_follower( port,time,ref );
elseif ((Odour_status == 0) && ((reflex(3)<500)||(reflex(4)<500)))

        CAST2( ref, port,  time) ;
        Face_The_Wind(ref, port, time  );
        Cast_Forward( ref, port, time, Step_Size )   ;
elseif   (((reflex(3)>500)||(reflex(4)>500)||(reflex(2)>500)||(reflex(5)>500))&& x < 55 )
kiks_print(sprintf('Upwind limit of the field reached, travelling downwind'));
Passes=Passes-1;
if Passes>0
    TANGENT( port,ref,time )
    %Travel_downwind( ref, port, time );
else

% Calculations and end of simulation procedures
kiks_kclose(ref);
end
else
    kSetSpeed(ref,0,0)
end
kiks_print(sprintf('Odour status = %d',Odour_status));
end

 Dist_from_Source = sqrt((ESTIMATED_SOURCE_X-250)^2 +  (ESTIMATED_SOURCE_Y - 1500)^2);

kiks_print(sprintf('Distance from actual source: %d', round(Dist_from_Source)));

kiks_kclose(ref);
end
```

6.2 Casting behaviour code

```
function CAST2( ref, port,  time )
%Function emulates the casting behaviour of the Male moth.
%Robot will move left and right with increasing amplitude
global  Odour_status Wind_direction

Get_reading(ref);
Step_Size = 100 ;
pos =kBattle_sensors(ref);
% Y position of robot (in mm, from top edge).
     y = pos(1); if (y <= 1500)
%_____________________________________________________________________
while(kiks_ktime(port)<time)
Get_reading(ref);
 reflex = kProximity(ref);
pos =kBattle_sensors(ref);
    % X position of robot (in mm, from left edge).
     x = pos(2);
if   (((reflex(3)>500)||(reflex(4)>500)||(reflex(2)>500)||(reflex(5)>500))&& x < 55 )
break
end
 Ten_degrees_Right( ref, port, time ) ; % turns right
if (Odour_status == 1)
     break
end
 Step_Size = Step_Size*(1.5) ; %Increases step size
Cast_Forward( ref, port, time, Step_Size ); % Goes forward
 if (Odour_status == 1)
     break
end
One_seventy_Deg_Left( ref, port, time ); % turns left
 if (Odour_status == 1)
     break
end
 Step_Size = Step_Size*(1.5); %Increases step size
Cast_Forward( ref, port, time, Step_Size )  ; % Goes forward
if (Odour_status == 1)
     break
end
end
%_____________________________________________________________________

elseif (y > 1500)
%_____________________________________________________________________
  while(kiks_ktime(port)<time)
 One_seventy_Deg_Left( ref, port, time );
  reflex = kProximity(ref);
pos =kBattle_sensors(ref);
    % X position of robot (in mm, from left edge).
     x = pos(2);
if   (((reflex(3)>500)||(reflex(4)>500)||(reflex(2)>500)||(reflex(5)>500))&& x < 55 )
 break
end
 if (Odour_status == 1)
     break
end
Step_Size = Step_Size*(1.5) ; %Increases step size
Cast_Forward( ref, port, time, Step_Size )   ; %Goes forward
 if (Odour_status == 1)
     break
end
Ten_degrees_Right( ref, port, time ) ; % turns right

 if (Odour_status == 1)
     break
end
 Step_Size = Step_Size*(1.5); %Increases step size
Cast_Forward( ref, port, time, Step_Size )   ; %Goes forward
if (Odour_status == 1)
     break
end
end
%_____________________________________________________________________
end
```

7. "Basic" tangent bug algorithm

```
function TANGENT( port,ref,time )
global x_position y_position  reflex goal_x goal_y minx maxx miny maxy
ESTIMATED_SOURCE_X  ESTIMATED_SOURCE_Y

kSetEncoders(ref,0,0);
goal_y = ESTIMATED_SOURCE_X + 1000;
goal_x = ESTIMATED_SOURCE_Y;

while (kiks_ktime(port)<time)

%%%%%%%%%%%%%%%%%%%Robot heads towards the goal.%%%%%%%%%%
Head_towards_goal( ref,port,time )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%Robot Advances towards the goal.%%%%%%
Go_FORWARD( ref,port,time  )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Robot selects slope of obstacle boundary closest to straight motion to
goal slope (Distance from goal minimisation). achieved by following obsacle
boundary farthest from robot front sensors (s3 and s4)
 if (reflex(4)<reflex(3))

 %%%%%%%%%%%%%%%%%RIGHT Boundary follower starts here.%%%%%
 Right_boundary_follower_TAN( port,time,ref)
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



 elseif (reflex(4)>reflex(3))
 %%%%%%%%%%%%%%LEFT Boundary follower starts here.%%%%%%%%%%%%
Left_Boundary_follower_TAN( port,time,ref)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 end

%%%%%%%%%%%%%%If the goal is reached, stop.%%%%%%%%%%%%%%%%%%
if
((y_position>miny)&&(y_position<maxy))&&((x_position>minx)&&(x_position<maxx
))
           Final_Approach(ref,port,time  )
           break
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 end

end
```

8. Aims and objectives

1. Program all the main algorithms(Bug1, 2, tangent). The performance of the created bug will have to be compared to existing bugs , which implies coding them into the simulator.

2. Design an algorithm based on the tangent bug algorithm, with added decision making. It will also include some element of exhaustive searching in case the shortest path fails, so all solution can be tried before concluding that the goal cannot be reached.

3. Program and include in the simulator a “Chemical plume” to help developing an algorithm more efficient in real conditions.