Theoretical Analysis of Three Bio-Inspired Plume Tracking Algorithms

Thomas Lochmatter and Alcherio Martinoli

Abstract— **We derive the theoretical performance of three bio-inspired odor source localization algorithms (casting, surge-spiral and surge-cast) in laminar wind flow. Based on the geometry of the trajectories and the wind direction sensor error, we calculate the distribution of the distance overhead and the mean success rate using Bayes inference. Our approach is related to particle filtering and produces smooth output distributions. The results are compared to existing real-robot and simulation results, and a good match is observed.**

I. Introduction

With the advances in robotics and chemicals sensor research in the last decade, odor sniffing robots have become an active research area. Notably the localization of odor sources would allow for very interesting robotic applications, such as search and rescue operations, safety and control operations on airports or industrial plants, and humanitarian demining [1] [2] [3] [4]. Many of these applications are time-critical, i. e. odor sources should be found as fast as possible. But as the structure of plumes in the air is intermittent in both time and space [5] [6], tracking plumes is a challenging problem.

In recent work, we compared the *surge-spiral* algorithm [7] [8] [9] [10], the *surge-cast* algorithm and pure *casting* [11] [12] [13] [14] [15] [16] [17] in both simulation [20] and with real robots [21] [22] in laminar wind flow. The results of both studies revealed that upwind surge algorithms are more efficient than pure casting. When combined with incremental plume reacquisition strategies such as spiraling, these algorithm are also more robust with respect to environmental conditions and the choice of algorithm parameters. Simulation experiments furthermore revealed that the plume lost distance (the distance between seeing the last odor patch and declaring that the plume was lost) does not have a major influence on the overall performance of the algorithm, while the accuracy of the wind direction sensor does.

In this paper, we are studying the same three algorithms from a theoretical perspective. Within a simple framework, we derive two equations for each algorithm: an analytical expression for the performance under ideal conditions with a perfect wind sensor, and a probabilistic model taking into account the error of the wind direction sensor. The latter allows us to numerically calculate the performance distribution, which we compare to the results obtained in simulation and with the real robots.

All authors are with the Distributed Intelligent Systems and Algorithms Laboratory, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland. (thomas.lochmatter@epfl.ch, alcherio.martinoli@epfl.ch)

This work was supported by the National Competence Center in Research on Mobile Information and Communication Systems NCCR-MICS, a center supported by the Swiss NSF under grant number 5005-67322.

As in our previous work, we are mainly interested in the success rate and distance overhead of plume tracking (i. e. following the plume towards the source). Plume finding and source declaration are not within the scope of our work.

The remainder of this paper is structured as follows. We first present the three algorithms (Section II), the model (Section III) and the metrics (Section IV). In sections V, VI and VII, we then derive the theoretical distribution of all three algorithms and compare them to the real-robot and the simulation results. Finally, we conclude in Section VIII.

II. Algorithms

All three algorithms used in this paper are bio-inspired and a combination of upwind surge, casting, and spiraling [23]. The algorithms use only binary odor information, that is, they either *perceive the odor* or *do not perceive any odor*, but ignore different concentrations levels. Commonly, the measured concentration is thresholded to obtain this binary value, but more elaborate processing could be used as well.

Finally, all three algorithms need a wind sensor to measure the wind direction. As molecules are mainly transported by advection, this piece of information is very valuable. The wind speed is ignored.

Since we are only interested in the plume tracking behavior, the robot starts in the plume, and declares failure if it gets too far away from it. This allows us to rule out arena geometry effects, which could greatly influence the results (e. g., high variance introduced by randomized search techniques).

Similarly, source declaration is done by a supervisor (ideal source declaration) and therefore does not affect the results. Experiments are considered successful if the robot has come in physical vicinity of the source.

A. The Casting Algorithm

The *casting* algorithm is very similar to the one described by Li et al. [11]. As shown in Figure 1, a robot in the plume

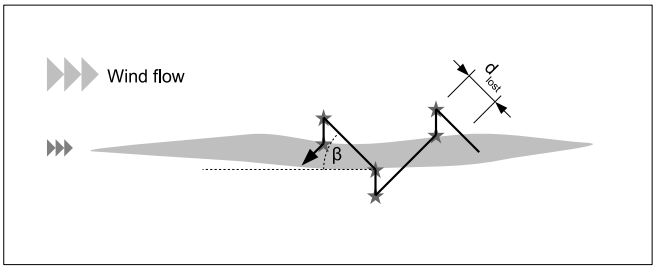


Fig. 1. Sketch of the *casting* algorithm. The stars indicate where the wind direction is measured.

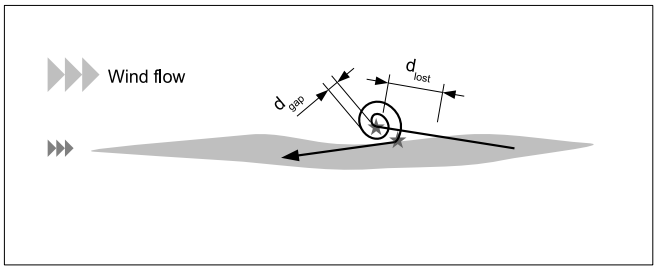


Fig. 2. Sketch of the *surge-spiral* algorithm. The star indicates where the wind direction is measured.

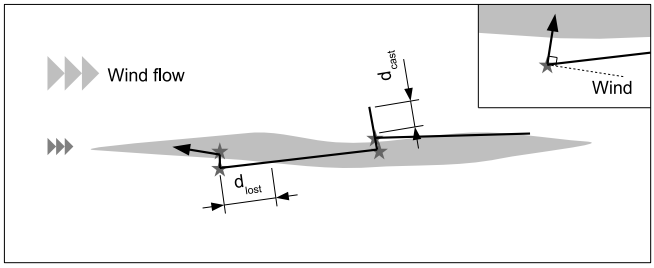


Fig. 3. Sketch of the *surge-cast* algorithm. The stars indicate where the wind direction is measured.

moves upwind with an angle β (relatively to the currently measured wind direction) until it is out of the plume for a certain distance, denoted d_{lost}. Once the plume is lost, the robot turns and moves cross-wind until it hits an odor packet, and then moves upwind with angle β again.

The wind direction is measured each time the robot switches to plume reacquisition, and when it encounters the plume again.

B. The Surge-Spiral Algorithm

The *surge-spiral* algorithm is similar to Hayes' algorithm presented in [7], except that we focus exclusively on its use for plume tracking here. Hence, we have a single spiral gap parameter.

A robot in the plume moves straight upwind until it loses the plume for a distance d_{lost}. It then tries to reacquire the plume by moving along an Archimedes spiral with gap size d_{gap}. Unlike [7], we start our spiral in upwind direction, as drawn in Figure 2.

The wind direction is measured when the robot switches from upwind surge to spiraling, and when it switches back to upwind surge.

C. The Surge-Cast Algorithm

The *surge-cast* algorithm [22] is a combination of upwind surge and cross-wind casting. It is similar to the surge-spiral algorithm, with the spiral being replaced by cross-wind movement.

A robot in the plume moves straight upwind until it loses the plume for a distance d_{lost}. It then tries to reacquire the plume by moving cross-wind for a set distance (d_{cast}), first on one side and then on the other. To maximize the chances of hitting the plume in the first cross-wind movement, the robot measures the wind direction to estimate from which side it left the plume.

If the robot did not reacquire the plume by casting, the run is considered unsuccessful. In a real application, the robot would probably switch back to plume finding behavior, or try to reacquire the plume with a larger cast distance or with spiraling.

The wind direction is measured when the robot switches from upwind surge to casting and when it switches back to upwind surge, as indicated in Figure 3.

III. Theoretical Model

The model used in this paper is closely related to the experimental setup used in our previous work with the real robots [21] [22] and in simulation [20]. However, plume and sensors are abstracted to simple mathematical objects that allow for a mathematical analysis. A comparison of the three models is given in Table III.

A. Wind and Plume Model

We consider a 2D space with a perfectly laminar wind flow and a single odor source emitting a chemical substance at constant rate. This substance is only transported by (large-scale) advection. Small-scale advection (responsible for the intermittent structure of the plume) and diffusion (an effect a few orders of magnitude smaller) are not modeled. For simplicity and without loss of generality, the wind is blowing in positive x direction at a speed of 1 m/s, i. e.

$$a(u) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ [m/s]} \tag{1}$$

at any position u.

Since the algorithms only take a binary input from the odor sensor, we model our plume as a straight line of constant width w starting at the odor source and extending to infinity in the direction of the wind. The robot — modeled as a point in 2D space — is considered *in the plume* if it lies on this line, and *out of the plume* otherwise.

While this model is far from physical reality, the behavior of all three algorithms in such a simplified model is approximately the same as in the real plume. As the algorithms pass the (binary) odor sensor input through a filter to smooth out all "gaps" shorter than the distance d_{lost} (see Figure 4), there

TABLE I

Comparison of the real-robot experiments, the simulation experiments and the theoretical model.

	Real exp.	**Simulation**	**Theory**
Environment	wind tunnel	Webots [24]	MATLAB
Wind	$\approx$ laminar	laminar	laminar
Plume	real, ethanol	filaments [25]	straight line
Plume width (w)	≈ 35 cm	35.4 cm	35.4 cm
d_{lost}	≈ 60 cm	61.4 cm	61.4 cm
Robot	Khepera III	Khepera III	point
Locomotion	diff.-drive	diff.-drive	holonomic
Odometry	good	perfect	perfect
Wind sensor error	non-gaussian	$N(0, (5.7^{\text{o}})^2)$	$N(0, (5.7^{\text{o}})^2)$
Odor sensor error	negligible	Gaussian, small	0
Odor sensor delay	$t_{90} \approx 0.1$ s	none	none

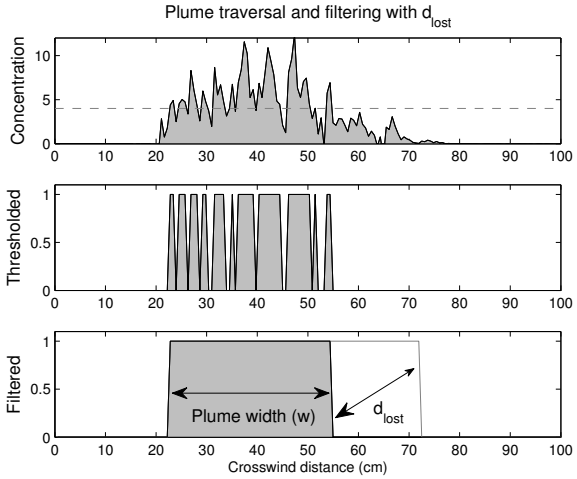


Fig. 4. Preprocessing of the odor concentration signal by the robots. In the real-robot and simulation experiments, the concentration threshold was tuned manually. Note that the plume lost distance, d_{lost}, is measured in along the trajectory of the robot, while the plume width, w, is measured in cross-wind direction.

is no need to model these gaps. This is actually the purpose of that filter, which has been shown to work well in the experiments with the real robot [22].

B. Wind Direction Sensor Model

The output of the wind direction sensor at position u, $\boldsymbol{a}_s(u)$, is modeled as an unbiased sensor with added Gaussian noise. That is,

$$\boldsymbol{a}_s(u) = a(u) + \boldsymbol{v}_a \tag{2}$$

where

$$\boldsymbol{v}_a \sim \begin{pmatrix} N(0,\sigma_a^2) \\ N(0,\sigma_a^2) \end{pmatrix} \text{ [m/s]} \tag{3}$$

This is the same model that we used in simulation [20]. Even though the noise is added to the X and Y components of the wind vector, the distribution of the angular noise is approximately Gaussian as well for sufficiently small σ_a. Each wind direction measurement is therefore susceptible to an angular error modeled by the random variable

$$\boldsymbol{\alpha} \sim N(0,\sigma_a^2) \qquad \text{with } \sigma_a = 0.1\,[\text{rad}] \tag{4}$$

The angular noise of the real sensor has a more complicated distribution, mostly because of quantization (10° intervals) and the occasional large errors.

IV. Metrics

In all our experiments, we compared the distance overhead, d_o, and the success rate, s_r, of the algorithms. The former is calculated as

$$d_o = \frac{d_t}{d_u} \tag{5}$$

where d_t denotes the effectively traveled distance by the robot and d_u the upwind distance (i. e. the distance by which the robot came closer to the source). The advantage of this metric is two-fold:

- ▷ d_o is distance independent as long as the plume structure (width, intermittency, concentrations) remains the same over the whole length. Hence, the results of different starting positions can be compared.
- ▷ d_o is independent of the kinematic constraints of our differential-drive robot.

The success rate, s_r, is the fraction of runs in which the robot successfully found the source. Note that this is a distance dependent value and stands with the success probability per upwind distance, s_p, in the following relation:

$$s_r = (s_p)^{d_u} \tag{6}$$

In the experiments presented in this paper, $d_u \approx 14\,\text{m}$.

V. Performance of the Casting Algorithm

We now calculate the distance overhead for all three algorithms with and without taking into account the wind direction sensor noise. For the case without noise, we derive an expression for the mean distance overhead, d_o, while for the case with the wind direction sensor noise, we numerically calculate the distribution of distance overhead and the mean of the success rate, s_r.

The procedure is the same with all three algorithms, as they all proceed by repeating a basic pattern until the source is found. These basic patterns are depicted in Figure 5. Each repetition of this pattern (called *iteration* in the remainder of this paper) brings the robot closer to the source, but also entails a certain probability to lose the plume completely. In this section, we present our approach in details for the *casting* algorithm, while the following sections only provide the equations for the other two algorithms.

A. Ideal Wind Direction Sensor

With an ideal wind sensor ($\boldsymbol{\alpha_1} = 0$, $\boldsymbol{\alpha_2} = 0$), the trajectory produced by the *casting* algorithm in our theoretical model is deterministic. Its distance overhead can be written as

$$d_o(\beta) = \frac{\frac{1}{\sin\beta} + f\,(1+\sin\beta)}{\left(\frac{1}{\sin\beta} + f\right)\cos\beta} \geq \frac{1}{\cos\beta} \tag{7}$$

with

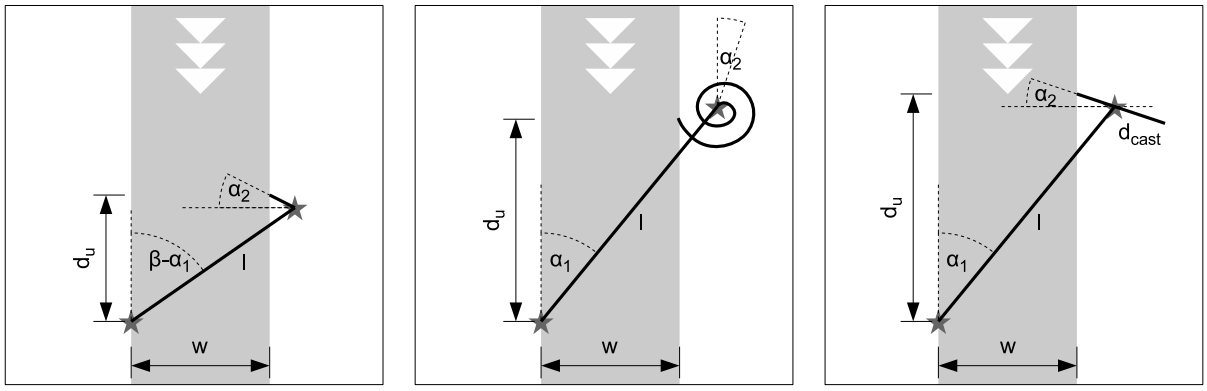
$$f = \frac{d_{\text{lost}}}{w} \tag{8}$$

Even though this expression may look complicated, it can easily be derived by looking at the geometry of the trajectory.

B. Noisy Wind Direction Sensor

a) Distribution of one iteration: With the *casting* algorithm, the basic pattern is produced by the following two steps:

1) Move upwind with an upwind angle β until the plume is lost for a distance greater than d_{lost}.
2) Move cross-wind until the plume is found again. Whether to turn left or right for this cross-wind motion is decided using the wind direction.



(a) Casting *(b) Surge-Spiral* *(c) Surge-Cast*

Fig. 5. Basic patterns (building blocks) of the three algorithms. In one iteration, the robot moves along the thick line.

Both steps consist of measuring the wind direction, turning towards the respective angle and moving forward while continuously sampling the odor concentration. While sampling speeds and accelerations are large enough to be ignored, the wind direction measurement introduces a non-negligible error. Each of the two readings is susceptible to noise modeled by the random variables $\boldsymbol{\alpha_1}$ resp. $\boldsymbol{\alpha_2}$. Note that as each reading is assumed to be independent, $\boldsymbol{\alpha_1}$ and $\boldsymbol{\alpha_2}$ are independent as well. Hence, the robot actually goes upwind with an angle $\beta - \boldsymbol{\alpha_1}$ and cross-wind with $\frac{\pi}{2} + \boldsymbol{\alpha_2}$.

Under these assumptions, we can — for a single iteration — calculate the distribution of the distance that the robot covers ($\boldsymbol{d_t}$), the distribution of the distance by which it approaches the source ($\boldsymbol{d_u}$), as well as the probability that the robot loses the plume completely and fails the run ($1-\boldsymbol{s}$). Using trigonometry, the following equations[1] are obtained:

$$\boldsymbol{d_t} = \boldsymbol{l} + d_{\text{lost}} \frac{\sin|\beta - \boldsymbol{\alpha_1}|}{\cos \boldsymbol{\alpha_2}} \tag{9}$$

$$\boldsymbol{d_u} = \boldsymbol{l}\cos(\beta - \boldsymbol{\alpha_1}) + d_{\text{lost}} \sin|\beta - \boldsymbol{\alpha_1}| \tan \boldsymbol{\alpha_2} \tag{10}$$

$$\boldsymbol{s} = \begin{cases} 1 & \text{if } \boldsymbol{\alpha_2} < \beta - \boldsymbol{\alpha_1} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where

$$\boldsymbol{l} = \frac{\boldsymbol{w'}}{\sin|\beta - \boldsymbol{\alpha_1}|} + d_{\text{lost}} \tag{12}$$

$$\boldsymbol{w'} = \begin{cases} w & \text{if } \boldsymbol{\alpha_1} < \beta \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Note that $\boldsymbol{d_t}$, $\boldsymbol{d_u}$ and $\boldsymbol{s}$ are dependent random variables, as they are generated using the same samples of $\boldsymbol{\alpha_1}$ and $\boldsymbol{\alpha_2}$.

The mean success probability of a single iteration can be calculated by marginalizing over $\boldsymbol{\alpha_1}$ and $\boldsymbol{\alpha_2}$:

$$\mathrm{E}(\boldsymbol{s}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{P}(\boldsymbol{s}|\boldsymbol{\alpha_1}, \boldsymbol{\alpha_2})\mathrm{P}(\boldsymbol{\alpha_1})\mathrm{P}(\boldsymbol{\alpha_2})\, \mathrm{d}\boldsymbol{\alpha_1}\mathrm{d}\boldsymbol{\alpha_2} \tag{14}$$

The mean distance overhead, $\mathrm{E}(\frac{\boldsymbol{d_t}}{\boldsymbol{d_u}})$, of one iteration could be calculated in a similar fashion, but is not of particular interest.

[1]To simplify the notation, we use random variables (i. e., distributions) as if they were normal variables (e. g. a sample of that distribution), but write them in bold font.

b) Distribution of a whole run: What we would like to calculate instead is the distance overhead of a complete run. To do that, we need to combine the iteration distributions until the upwind distance exceeds 14 m. Let us define

$$S^{(1)}(t,u) = \mathrm{P}(\boldsymbol{d_t} = t, \boldsymbol{d_u} = u, \boldsymbol{s} = 1) \tag{15}$$

as the distribution of the successful runs after one iteration in the space spanned by d_t and d_u. Note that $S(t,u)$ is not a probability density function in the strict sense, because

$$\int_{-\infty}^{\infty} S^{(1)}(t,u)\, \mathrm{d}(t,u) = \mathrm{E}(\boldsymbol{s}) \leq 1 \tag{16}$$

While the distribution of the successful runs after two iterations, $S^{(2)}(t,u)$ is simply the convolution of $S^{(1)}(t,u)$ with itself, the distribution after three iterations, $S^{(3)}(t,u)$, is the convolution of $S^{(2)}(t,u)$ with $S^{(1)}(t,u)$, and so on. Hence, we can combine any number of iterations by applying the convolution equation,

$$S^{(i)}(t,u) = \int_{-\infty}^{\infty} S^{(i-1)}(t-t_1, u-u_1) S^{(1)}(t_1,u_1)\, \mathrm{d}(t_1,u_1) \tag{17}$$

once for each added iteration. Note that this is only valid because iterations are mutually independent, i. e. $\alpha_1^{(j)}$ is indep. of $\alpha_1^{(k)}$, and $\alpha_2^{(j)}$ is indep. of $\alpha_2^{(k)}$, $\forall j \neq k$. As an example, the first four iterations for $\beta = 30^{\text{o}}$ are depicted in Figure 6.

Since $S^{(1)}(t,u)$ does not include the failing runs after one iteration, $S^{(i)}(t,u)$ do not include them neither. Hence, the fraction of successful runs after i iterations is simply

$$\mathrm{E}(\boldsymbol{s}^{(i)}) = \int_{-\infty}^{\infty} S^{(i)}(t,u)\, \mathrm{d}(t,u) = (\mathrm{E}(\boldsymbol{s}))^i \leq 1 \tag{18}$$

To calculate the distribution of complete runs, it suffices to combine iterations until

$$S^{(i)}(t,u) = 0 \qquad \forall u < 14\,\mathrm{m} \tag{19}$$

and to collect statistics.

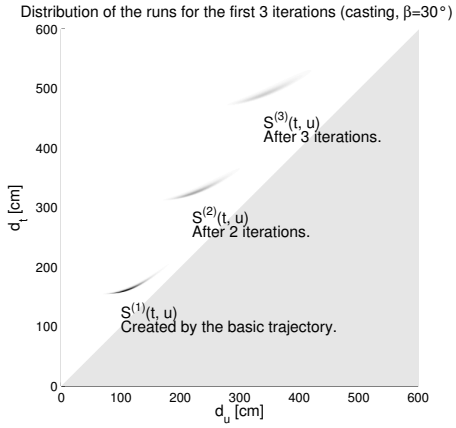


Fig. 6. $(\boldsymbol{d_t}, \boldsymbol{d_u})$ space with the first 3 iterations (overlaid) of the *casting* algorithm with $\beta = 30^{\circ}$. $S^{(1)}(t, u)$ is the distribution for a single iteration and all other distributions are convolutions of it. The gray shaded lower right triangle is impossible to reach because $d_t > d_u$ (by construction).

c) Implementation Issues: Carrying out the above procedure analytically is clearly not viable. However, the equations can easily be solved numerically, either with a Monte-Carlo approach (randomly selecting input samples and calculating a histogram of the output samples) or by discretizing the distributions and calculating the joint distributions numerically. We have chosen the second approach, as it provides smoother output distributions. This corresponds to particle filtering with fixed particles placed on a regular grid.

The implementation is fairly straightforward. We have chosen a resolution of 0.02° for α_1 and α_2 and cut them off at $\pm 25^{\circ}$. The distribution $S^{(i)}(t, u)$ was approximated with a 30 m by 30 m square lattice with a resolution of 1 cm in each direction. Care must be taken with big values for d_t and d_u that do not fit within this lattice, as one would lose probability mass when ignoring them. We simply added these values to the cell (30 m, 30 m) which approximates their $\frac{d_t}{d_u}$ ratio with 1.

In addition, equation (17) is $\mathrm{O}(n^2)$ where n denotes the number of cells of $S^{(i)}(t, u)$, which can be large. Since most values in this matrix are zero or very small, however, the algorithm can be boost by using sparse matrices. After each iteration, we furthermore removed all values with $d_u > 14$ m after having calculated the necessary statistics.

C. Results

Figure 7 shows the theoretically calculated as well as the experimentally measured distance overhead of the *casting* algorithm. Figure 8 shows the corresponding success rates.

For $\beta > 15^{\circ}$, the theoretically derived distance overhead distribution is almost normal and in accordance with our previous findings [26].

The simulation results (boxes) match very well with the theoretical distribution. Small differences with larger upwind angles could be due to the placement of the odor sensor. In the theoretical model, this sensor was assumed to be centered on the robot, while the real sensor was put in front of the robot, at about 7 cm from its kinematic center.

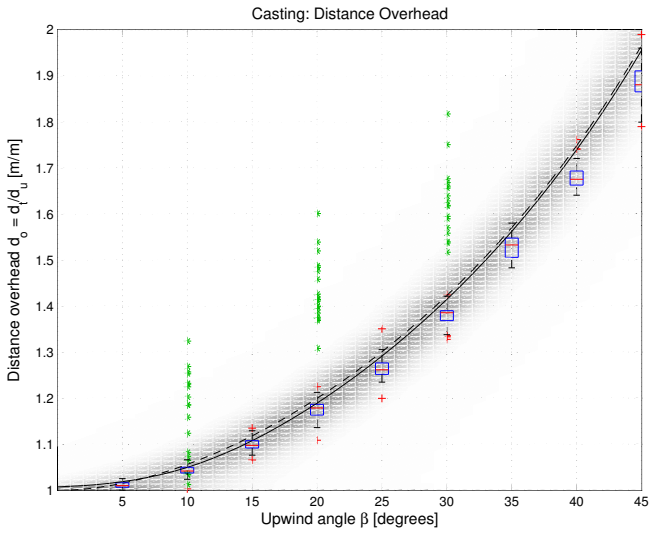


Fig. 7. Theoretical and experimental distance overhead of the casting algorithm. *Green stars:* Real-robot experiments (20 runs each). *Blue boxes and red crosses:* Box-plot of the simulation results (50 runs each). The box shows the lower/upper quartile and the red line denotes the median. Red crosses stand for outliers. *Gray shading:* Theoretically derived distribution of the upwind overhead. *Black lines:* Expected mean upwind distance, calculated from the distribution (solid line) and with equation (7) (dashed line).

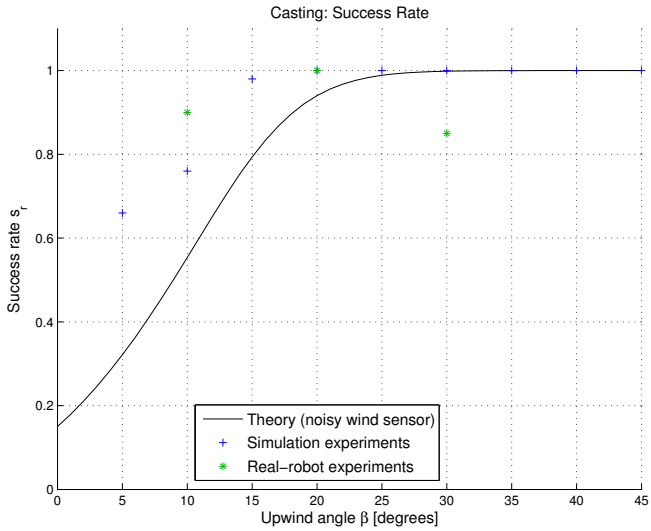


Fig. 8. Theoretical and experimental success rate of the *casting* algorithm.

In addition, equation (7) (dashed line) for an ideal wind sensor is a very good approximation of the mean obtained with our non-ideal wind sensor. While an ideal wind sensor would allow us to reach the optimal performance for very steep upwind angles, its performance is slightly worse for $\beta > 8^{\circ}$. Randomness can indeed boost the performance here, as the relationship between performance and effective upwind angle is not linear.

The real-robot results are significantly worse in terms of distance overhead, but better when comparing the success rate (except for $\beta = 30^{\circ}$). Reasons for this are believed to be two-fold. First, closer inspection of the real robot trajectories

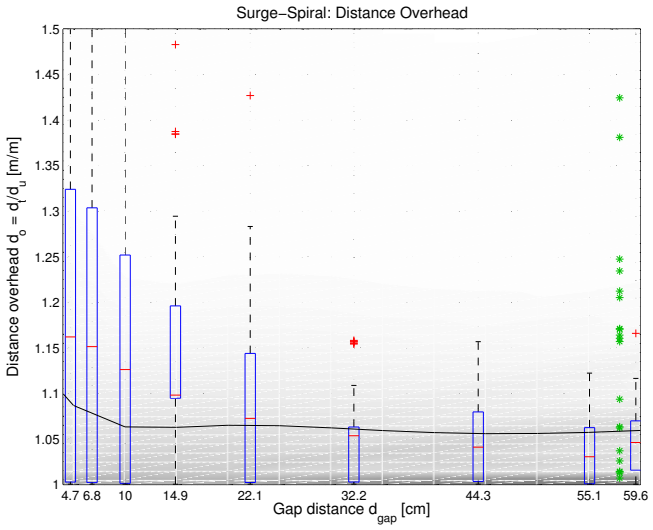


Fig. 9. Theoretical and experimental performance of the *surge-spiral* algorithm. *Green stars:* Real-robot experiments (20 runs each). *Blue boxes and red crosses:* Box-plot of the simulation results (50 runs each). The box shows the lower/upper quartile and the red line denotes the median. Red crosses stand for outliers. *Gray shading:* Theoretically derived distribution of the upwind overhead. *Black line:* Expected mean upwind distance, calculated from the distribution.

revealed that the cross-wind angle was almost systematically biased towards the downwind direction. Similarly, the actual upwind angle was 2° – 5° higher than what was configured. This is an artefact of the measurement resolution of the wind direction sensor, which was only 10° [22]. Second, the flow right in front of the odor source was slightly turbulent and sometimes caused additional errors in the wind direction measurement. Even though these were manually removed in the most detrimental cases, the trajectories close to the source are still slightly less ideal.

VI. Performance of the Surge-Spiral Algorithm

A. Ideal Wind Direction Sensor

With an ideal wind direction sensor, the upwind overhead of the *surge-spiral* is simply

$$d_o(d_{\text{gap}}) = 1 \tag{20}$$

Since the robot starts in the plume and moves straight upwind in this ideal plume, it never leaves it. Hence, in contrast to the *casting* algorithm, the *surge-spiral* algorithm achieves optimal performance under ideal conditions.

B. Noisy Wind Direction Sensor

With a noisy wind direction sensor, we again break the basic pattern of the the algorithm into two steps:

1) Move straight upwind until the plume is lost for a distance greater than d_{lost}. Due to the wind direction measurement error, the actual upwind angle is $\boldsymbol{\alpha_1}$.
2) Moves along an Archimedean spiral until the plume is found again. The wind direction measurement here ($\boldsymbol{\alpha_2}$) only serves to decide whether to start the spiral towards left or right.

The *surge-spiral* algorithm does not have any failure condition[2]. Under the ideal assumptions taken here and the fact that the spiral increases, the robot will eventually reacquire the plume. The three distributions therefore are

$$\boldsymbol{d_t} = \boldsymbol{l} + r_l(d_{\text{gap}}, d_{\text{lost}}\boldsymbol{b}\sin|\boldsymbol{\alpha_1}|) \tag{21}$$

$$\boldsymbol{d_u} = \boldsymbol{l}\cos\boldsymbol{\alpha_1} + r_y(d_{\text{gap}}, d_{\text{lost}}\boldsymbol{b}\sin|\boldsymbol{\alpha_1}|) \tag{22}$$

$$\boldsymbol{s} = 1 \tag{23}$$

where

$$\boldsymbol{l} = \begin{cases} d_{\text{lost}} & \text{if } \boldsymbol{\alpha_1} < 0 \\ \frac{w}{\sin\boldsymbol{\alpha_1}} + d_{\text{lost}} & \text{otherwise} \end{cases} \tag{24}$$

$$\boldsymbol{b} = \begin{cases} -1 & \text{if } \boldsymbol{\alpha_2} < \boldsymbol{\alpha_1} < 0 \\ -1 & \text{if } 0 < \boldsymbol{\alpha_1} < \boldsymbol{\alpha_2} \\ 1 & \text{otherwise} \end{cases} \tag{25}$$

$r_l(d_{\text{gap}}, x)$ and $r_y(d_{\text{gap}}, x)$ are the trajectory length resp. the upwind component of the spiraling maneuver with spiral gap d_{gap} and distance x from the plume. As these values are difficult to calculate analytically, we numerically integrated over a spiral trajectory to find them. This is much more precise than approximating the spiral with a circle and straightforward to implement.

Since $\sin\boldsymbol{\alpha_1} \to 0$ for small $\boldsymbol{\alpha_1}$, it is clear that a good wind direction sensor will significantly increase the upwind step length, and therewith significantly improve the performance of the algorithm.

The rest of the calculation is exactly the same as introduced in Section V.

C. Results

Figure 9 shows the distance overhead for the *surge-spiral* algorithm. Despite the high variance of the simulation results, the overall match between simulation and theory is pretty good. Both capture the drop in performance for small spiral gaps, and both predict a fairly constant performance over a wide range of larger gap distances.

As opposed to the *casting* algorithm, the distribution generated by *surge-spiral* is almost exponential. While most runs yield a good distance overhead value, some runs are very bad. Indeed, a number of outliers can be observed the simulation and real-robot results (which are available for $d_{\text{gap}} = 58\,\text{cm}$ only).

For large d_{gap} values, the theoretically derived distance overhead distribution becomes slightly bumpy. This is a result of the discrete number of iterations that the algorithm performs. The number of real-robot and simulation runs is too small to observe the same effect there.

VII. Performance of the Surge-Cast Algorithm

A. Ideal Wind Direction Sensor

For the same reasons as for the *surge-spiral* algorithm, the upwind overhead for the *surge-cast* algorithm under ideal conditions is

$$d_o(d_{\text{cast}}) = 1 \tag{26}$$

[2]In the simulation and real robot experiments, the only condition for failure was when the robot touched the arena wall. This, however, was very unlikely even with the real robots and never happened.

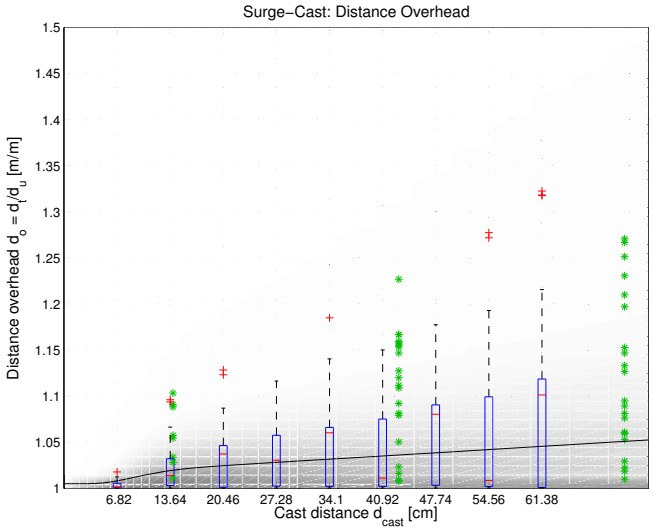


Fig. 10. Theoretical and experimental performance of the *surge-cast* algorithm. *Green stars:* Real-robot experiments (20 runs each). *Blue boxes and red crosses:* Box-plot of the simulation results (50 runs each). The box shows the lower/upper quartile and the red line denotes the median. Red crosses stand for outliers. *Gray shading:* Theoretically derived distribution of the upwind overhead. *Black line:* Expected mean upwind distance, calculated from the distribution.

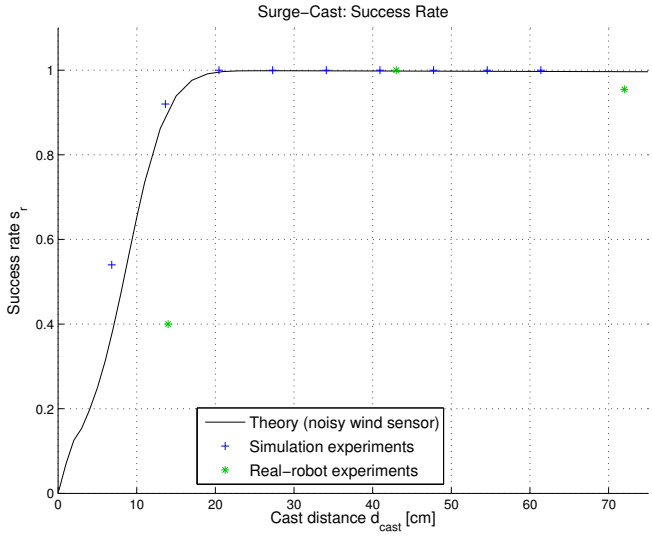


Fig. 11. Theoretical and experimental success rate of the *surge-cast* algorithm.

B. Noisy Wind Direction Sensor

Since *surge-cast* and *surge-spiral* only differ their plume reacquisition strategy their equations look very similar:

$$\boldsymbol{d_t} = \boldsymbol{l} + \boldsymbol{c_t}(\boldsymbol{\alpha_1}, \boldsymbol{\alpha_2}, d_{\text{lost}}) \quad (27)$$

$$\boldsymbol{d_u} = \boldsymbol{l}\cos\boldsymbol{\alpha_1} + \boldsymbol{c_u}(\boldsymbol{\alpha_1}, \boldsymbol{\alpha_2}, d_{\text{lost}}) \quad (28)$$

$$\boldsymbol{s} = \begin{cases} 1 & \text{if } d_{\text{lost}}\frac{\sin\boldsymbol{\alpha_1}}{\cos\boldsymbol{\alpha_2}} < d_{\text{cast}} \\ 0 & \text{otherwise} \end{cases} \quad (29)$$

Instead of the spiraling maneuver, however, the *surge-cast* algorithm casts to reacquire the plume. The corresponding equations for c_t and c_u are:

$$\boldsymbol{c_t} = d_{\text{lost}}\frac{\sin|\boldsymbol{\alpha_1}|}{\cos\boldsymbol{\alpha_2}} + \boldsymbol{b} \quad (30)$$

$$\boldsymbol{c_u} = d_{\text{lost}}(\cos\boldsymbol{\alpha_1} + \sin|\boldsymbol{\alpha_1}|\tan\boldsymbol{\alpha_2}) \quad (31)$$

$$\boldsymbol{b} = \begin{cases} d_{\text{cast}} & \text{if } \boldsymbol{\alpha_2} < \boldsymbol{\alpha_1} < 0 \\ d_{\text{cast}} & \text{if } 0 < \boldsymbol{\alpha_1} < \boldsymbol{\alpha_2} \\ 0 & \text{otherwise} \end{cases} \quad (32)$$

Note also that the *surge-cast* algorithm fails if the robot does not find the plume by casting backward and forward.

The rest of the calculation is again the same as introduced in Section V.

C. Results

The distance overhead and the success rate of the *surge-cast* algorithm are plotted in Figure 10 resp. Figure 11.

The match between simulation and theory is excellent for both the distance overhead and the success rate. The exponential distribution predicted by the theory is visible on the outliers of the simulation results.

The real-robot results are only slightly worse, but follow the same trends. While a few very good runs can be observed with the real robots, their performance does not exactly follow an exponential law. Closer inspection of the individual runs suggests that this is mainly due to the odometry bias which makes the robot turn slowly when it intends to go straight. Such errors do not have a big impact on a bad run, but makes a perfect run very unlikely.

For both the *surge-cast* and the *surge-spiral* algorithms, the theoretical prediction with an ideal wind sensor does not provide an accurate model of the performance of upwind surge algorithms. At least in laminar flow, these algorithms highly depend on the wind direction sensor and its accuracy [20].

VIII. Conclusion

We derived the performance of three bio-inspired algorithms in laminar wind flow within a very simple theoretical model. For each algorithm, a deterministic equation for a ideal wind sensor and a probabilistic approach taking into account the wind sensor noise were introduced. The latter provides the distribution of the distance overhead, as well as the mean success rate.

The performance was compared with our previously published simulation [20] and real-robot [21] [22] results, and a good overall match was observed. The analysis at these three levels was very complementary:

- ▷ The **real-robot experiments** helped us finding out the accuracy of the sensors and the type (distribution) of errors they yield. They also pointed us to real-world problems such as odometry drift that are not commonly modeled in simulation programs. But most importantly, they helped us developing some intuition for the environment and parameters, which was crucial in the design of the simulation experiments and the theoretical model. Finally, they also served as real-robot validation of the results obtained in simulation and in theory.

- ▷ The **simulation experiments** allowed us to study the importance and influence of the algorithmic parameters. For our laminar flow scenario, we concluded that the plume lost distance does not have a big impact on the performance, while the the accuracy of the wind sensor did.
- ▷ The **theoretical results** finally allowed us to study the distribution of the distance overhead, as well as the expected mean distance overhead and success rate under ideal conditions. While the distribution of the distance overhead for the *casting* algorithm is almost normal (Gaussian), *surge-spiral* and *surge-cast* yield approximately exponential distributions. This could be an issue for real-world applications if predictability of the performance is more important than speed.

Altogether, the experiments provide a good overall picture of these three bio-inspired algorithms and demonstrate the interplay of the three underlying behaviors (casting, spiraling, and upwind surge) observed in nature. We showed that pure casting is inefficient for large upwind angles, and not very robust for small upwind angles. Upwind surge strategies have a big speed advantage, especially if the wind direction can be determined accurately. However, they need to be combined with a plume reacquisition strategy. Using a local search strategy (e. g., spiraling) to reacquire the plume yields very robust algorithms. Casting for plume reacquisition is faster if reliable wind direction information is available, such as in our experiments.

It is important to note that our findings presented here hold for a static scenario with laminar or quasi-laminar wind flow. Turbulence makes it intrinsically much harder to determine the wind direction, and may have a major impact on the results. In addition, obstacles will not only negatively influence the performance, but require to modify the algorithms [27]. Hence, our results should be viewed as an upper limit on the performance that can be achieved in quasi-ideal conditions, and not as a performance target in real-world conditions.

REFERENCES

[1] G. S. Settles, "Sniffers: Fluid-dynamic sampling for olfactory trace detection in nature and homeland security—the 2004 freeman scholar lecture," in *Journal of Fluids Engineering*, ser. Transactions of the ASME, vol. 127, 2005, pp. 189–218.

[2] D. W. Gage, "Many-robot MCM search systems," in *Proceedings of the Autonomous Vehicles in Mine Countermeasures Symposium*, April 1995, pp. 9.56–9.64.

[3] M. Long, A. Gage, R. Murphy, and K. Valavanis, "Application of the distributed field robot architecture to a simulated demining task," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, April 2005, pp. 3193–3200.

[4] H. Ishida, T. Nakamoto, T. Moriizumi, T. Kikas, and J. Janata, "Plume-tracking robots: A new application of chemical sensors," *Biological Bulletin*, no. 200, pp. 222–226, April 2001.

[5] M. Vergassola, E. Villermaux, and B. I. Shraiman, "'Infotaxis' as a strategy for searching without gradients," *Nature*, vol. 445, pp. 406–409, January 2007.

[6] A. J. Lilienthal, A. Loutfi, and T. Duckett, "Airborne chemical sensing with mobile robots," *Sensors*, vol. 6, pp. 1616–1678, October 2006.

[7] A. T. Hayes, A. Martinoli, and R. M. Goodman, "Distributed odor source localization," *IEEE Sensors Journal*, vol. 2, no. 3, pp. 260–271, June 2002.

[8] ——, "Swarm robotic odor localization: Off-line optimization and validation with real robots," *Robotica*, vol. 21, pp. 427–441, 2003.

[9] J. H. Berlanger and M. A. Willis, "Adaptive control of odor-guided locomotion: behavioral flexibility as an antidote to environmental unpredictability," *Adaptive Behavior*, vol. 4, no. 3-4, pp. 217–253, August 1996.

[10] G. Ferri, E. Caselli, V. Mattoli, A. Mondini, B. Mazzolai, and P. Dario, "A biologically-inspired algorithm implemented on a new highly flexible multi-agent platform for gas source localization," in *Proceedings of the First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics (BIOROB 2006)*, February 2006.

[11] W. Li, J. A. Farrell, and R. T. Cardé, "Tracking of fluid-advected odor plumes: Strategies inspired by insect orientation to pheromone," *Adaptive Behavior*, vol. 9, no. 3-4, pp. 143–170, 2001.

[12] L. P. S. Kuenen and H. C. Rowe, "Cowpea weevil flights to a point source of female sex pheromone: analyses of flight tracks at three wind speeds," *Physiological Entomology*, vol. 31, no. 2, p. 103, June 2006.

[13] B. Webb, R. R. Harrison, and M. A. Willis, "Sensorimotor control of navigation in arthropod and artificial systems," *Arthropod Structure and Development*, vol. 33, pp. 301–329, May 2004.

[14] A. J. Lilienthal, D. Reiman, and A. Zell, "Gas source tracing with a mobile robot using an adapted moth strategy," in *Autonome Mobile Systeme (AMS), 18. Fachgespräch*. GDI, December 2003, pp. 150–160.

[15] W. Li, J. A. Farrell, S. Pang, and R. M. Arrieta, "Moth-inspired chemical plume tracing on an autonomous underwater vehicle," *IEEE Transactions on Robotics*, vol. 22, no. 2, pp. 292–307, April 2006.

[16] E. Balkovsky and B. I. Shraiman, "Olfactory search at high reynolds number," *PNAS*, vol. 99, no. 20, pp. 12 589–12 593, October 2002.

[17] K. A. Justus and R. T. Cardé, "Flight behaviour of males of two moths, cadra cautella and pectinophora gossypiella, in homogeneous clouds of pheromone," *Physiological Entomology*, vol. 27, no. 1, pp. 67–75, March 2002.

[18] W. Jatmiko, K. Sekiyama, and T. Fukuda, "A pso-based mobile robot for odor source localization in dynamic advection-diffusion with obstacles environment," *IEEE Computational Intelligence Magazine*, pp. 37–51, May 2007.

[19] L. Marques, U. Nunes, and A. T. de Almeida, "Particle swarm-based olfactory guided search," *Autonomous Robots*, vol. 20, no. 3, pp. 277–287, June 2006.

[20] T. Lochmatter and A. Martinoli, "Simulation experiments with bio-inspired algorithms for odor source localization in laminar wind flow," in *Proceedings of the The Seventh International Conference on Machine Learning and Applications (ICMLA 2008)*. San Diego, CA, USA: IEEE, December 2008, pp. 437–443.

[21] T. Lochmatter, X. Raemy, L. Matthey, S. Indra, and A. Martinoli, "A comparison of casting and spiraling algorithms for odor source localization in laminar flow," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA 2008)*, May 2008, pp. 1138–1143.

[22] T. Lochmatter and A. Martinoli, "Tracking odor plumes in a laminar wind field with bio-inspired algorithms," in *Proceedings of the 11th International Symposium on Experimental Robotics 2008 (ISER 2008)*, ser. Springer Tracts in Advanced Robotics (2010), Athens, Greece, July 2008, to appear.

[23] R. A. Russell, *Odour Detection by Mobile Robots*, ser. World Scientific Series in Robotics and Intelligent Systems. World Scientific Publishing Company, 1999, vol. 22.

[24] O. Michel, "Webots: Professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[25] J. A. Farrell, J. Murlis, X. Long, W. Li, and R. T. Cardé, "Filament-based atmospheric dispersion model to achieve short time-scale structure of odor plumes," *Environmental Fluid Mechanics*, vol. 2, pp. 143–169, 2002.

[26] T. Lochmatter and A. Martinoli, "Understanding the potential impact of multiple robots in odor source localization," in *Proceedings of the 9th Symposium on Distributed Autonomous Robotic Systems (DARS 2008)*, 2008, to appear.

[27] T. Lochmatter, N. Heiniger, and A. Martinoli, "Localizing an odor source and avoiding obstacles: Experiments in a wind tunnel using real robots," in *Proceedings of the 13th International Symposium on Olfaction and Electronic Nose (ISOEN 2009)*, April 2009, to appear.