LadyBug. An Intensity based Localization Bug Algorithm

1st Athanasios Lentzas
School of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
alentzas@csd.auth.gr

2nd Dimitris Vrakas
School of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
dvrakas@csd.auth.gr

***Abstract*—Localization and navigation is a crucial task of autonomous robots. This paper introduces LadyBug a novel bug algorithm. Using the Received Signal Strength Indication (RSSI) of an electromagnetic signal, the algorithm is able to accurately calculate the position of the beacon emitting the aforementioned signal. Various experiments were performed with a simulated robot equipped only with local sensors. The proposed algorithm was compared with similar approaches with very promising results.**

***Index Terms*—robot navigation, robot localization, RSSI, bug algorithm, path planning**

I. Introduction

Recently mobile robots have become more commonplace in commercial and industrial settings. Self localization is a crucial task for all autonomous robots. Localization relying on odometry or inertial sensors, although it can provide robust results against environmental changes, is susceptible to error accumulation. In order to overcome that problem, solutions based on beacons have been proposed in the literature [1]. Despite the fact that localization improves with the usage of beacons, complex beacon placement and expensive sensors needed, could increase the cost of such solution. In this paper an approach based on the Received Signal Strength Indication (RSSI) of a beacon that emits any electromagnetic signal is proposed for localization.

Localization is important for robot navigation. Indoor navigation of robots is an area gathering more scientific interest, as it can be exploited in various scenarios, e.g. search and rescue, industrial settings, assistant robots etc. Although simultaneous localization and mapping (SLAM) techniques are extensively used [2], their high complexity and computational cost is a drawback when the use of a small, simple robot is needed. Bug algorithms are ideal for deployment on small, resource limited robots. They are characterized by low complexity and minimum computational needs while they remain effective.

In this study, a novel bug algorithm (LadyBug) is presented. For localization a light source was selected as the source of the electromagnetic signal. Among the several available beacons the particular one was chosen based on the easily available sensors and the relevant low cost. RSSI based localization is a subject extensively researched in the literature [3]. Most published papers rely on a network of beacons. In our approach, a single beacon was employed and the algorithm was able to accurately locate its position.

II. Related Work

Bug algorithms is a topic extensively researched. The first bug algorithm introduced was the *Common sense algorithm* (*Com*) [4], also known as *Bug0*. A robot using *Bug0* will move towards a target until an obstacle is found. When an obstruction is detected, it will move around it, until there is no obstacle between the robot and its target.

Although *Bug0* could solve many mazes, the algorithm is not complete, as there are scenarios where it can't reach the goal. In order to overcome the problems encountered, *Bug1* was introduced [4]. When *Bug1* detects an obstacle, it will circumnavigate it completely. While moving around the obstacle, it keeps track of the point closest to the target. When the robot reaches the point where it found the obstacle, it will continue to the point closest to the target, and there it will move towards the goal.

Bug1 is a complete algorithm but the path found is not optimal. Therefore a new algorithm was proposed, called *Bug2* [5]. *Bug2* draw an imaginary straight line between the obstacle and the goal. The robot will follow the obstacle until the line is found at a point closer to the goal than the hit point.

A variation of the *Com* algorithm, named *Com1*, was proposed in the literature [6]. The proposed algorithm, compared to the previous version, remembers the hit point's distance to the target. As a result it will leave the obstacle only when the path to the goal is clear and the distance to goal is smaller than the hit point. All the aforementioned approaches rely on contact sensors to perceive the obstacles found on their path.

Relying on the logic of *Com1*, an intensity bug algorithm (*I-Bug*) was proposed [7]. *I-Bug* navigates towards a beacon based on the signal strength. The authors assume that a tower orientation sensor is used, allowing the agent to know whether its facing the source or not. When an obstacle is found the intensity is saved and circumnavigation begins. While navigating around the perimeter, the intensity is compared to the one on the hit point as well as the intensity a time step back. If the strength of the signal decreases after increasing then a local maximum has been detected and the robot leaves the obstacle. It is worth mentioning that in order to leave the obstacle and navigate to the source, the intensity must be larger than the hit point, i.e. closer to the target.

Although the majority of Bug Algorithms proposed offer a simplistic approach to robot navigation, they rely on an accurate positioning system [8].

III. Motivation

The key idea of *I-Bug* is very interesting and it could prove to be very important in many real world scenarios where the environment is GPS denying and the robot cannot afford to navigate around using SLAM (e.g. time critical situations or limited on board resources). Our motivation was to build an approach where the robot would still use local sensing alone, but have the ability to extract the exact location of the goal (in robot-centric coordinates system) in real time.

The benefits of such an approach would be numerous: a) The robot would be able to follow an optimized path, since both the starting and goal points would be known. b) The calculated trajectory would be smoother, since I-Bug has to stop every now and then and rotate in place in order to face the target. c) When travelling around obstacle the robot would be able to leave the circumnavigation behavior earlier resulting in shorter paths. d) It would be more noise resilient since the robot could apply averaging techniques in order to estimate the goal position using past measurement and e) it would facilitate the use of more advanced navigation algorithms, like *Tangent Bug* [9].

IV. LadyBug Algorithm

The LadyBug algorithm proposed in this paper has two distinct parts. The first part is localization, where the position of the beacon in the robot-centric coordinates system are calculated. The second part is navigation. where the robot navigates to the beacon while avoiding obstacles found on it's way.

The navigation part of our algorithm is similar to the one employed by *Com1* algorithm. The main idea of *Com1* is expanded by: a) the local sensing, localization scheme, b) the circumnavigation behaviour that uses distance sensors instead of bumpers and c) an efficient yet simple controller for moving in empty spaces. The complete *LadyBug* algorithm can be seen on Alg.(1).

At each cycle the algorithm employs the localization scheme in order to estimate the position of the source. This estimation is combined with the estimation made in the previous cycle in order to amend for occasional faulty readings due to noise. The previous cycle estimation contains information from all the previous readings, thus using it provides more robust results. The next step is to use the projection of the source to the ground (thus the elimination of the z coordinate) to calculate the distance to the target. If the distance is larger than a threshold (C_{zero}) *LadyBug* checks if the way towards the target is blocked or not in order to choose between the Circumnavigate and the MoveToGoal behaviours. Note that when the algorithm senses an obstacle (hit point) and employs circumnavigation it stores the distance to the target (d_L) in order to ensure that the leave point will be closer to the target than the hit one.

Algorithm 1: LadyBug Algorithm

```
1  $(x, y, z) \leftarrow find\_Beacon\_Location()$
2  $(x, y, z) \leftarrow (x', y', z') * 0.9 + (x, y, z) * 0.1$
     // (x',y',z') is the beacon's
     location calculated on the previous
     step
3
4  $dist \leftarrow \sqrt{x^2 + y^2}$
5  if $dist < C_{zero}$ then
6      $stop()$                          // goal reached
7  end
8  $\theta = atan2(y, z)$  // find angle towards the
     goal
9
10 if readings of all sensors in$(\theta - C_{ang}, \theta + C_{ang}) >$
     $C_{safe} and dist < d_L$          // Leave obstacle
11  then
12     $d_L \leftarrow \infty$
13     $MoveToGoal()$
14 end
15 else
16     if $d_L = \infty$                    // Hit obstacle
17      then
18         $d_L \leftarrow dist$
19     end
20     Circumnavigate()
21 end
```

A. Localization

The localization scheme of the robot is based on RSSI. Given an electromagnetic wave (light, radio, Bluetooth etc.), the localization sub system can calculate the coordinates (in the robot-centric system) of the source in 3d space. This removes the requirement for a universal localization system. Additionally, our approach requires sensors that are easily accessible (light, radio, Bluetooth etc) as long as they return the intensity of the received signal.

1) Basic Scheme: The main idea is to use the RSSI measurements of four sensors in order to calculate the position of the electromagnetic source in the 3d space. These sensors (S_0, S_R, S_L, S_F) should be placed on the perimeter of a circle with radius *r* as seen on Fig. 1. Each sensor returns a measurement E, defined by (1), which is proportional to the intensity of the source (W) and the angle of incidence (ϕ) and inversely proportional to the square of the distance between the sensor and the source.

$$E = \frac{W}{4\pi d^2} \cos\phi \tag{1}$$

The triangle formed by the sensor, the source and the projection of the source onto the sensors plane can be seen on Fig. 2. The cosine of the angle of incidence is equal to the

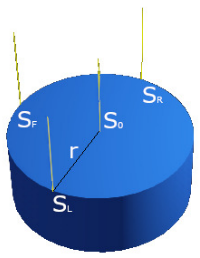


Fig. 1: The position of the four sensors on the robot.

sine of the angle formed by d and b ($\cos\phi = \sin(90-\phi)$). Thus equation (1) can be written as:

$$E = \frac{W}{4\pi d^2}\sin(90-\phi) \Rightarrow E = \frac{W}{4\pi d^2}\frac{h}{d} \Rightarrow E = \frac{Wh}{4\pi d^3} \quad (2)$$

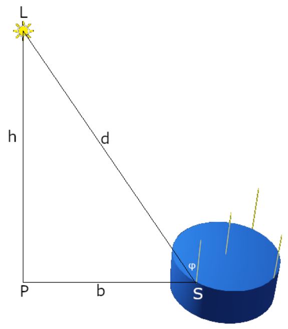


Fig. 2: The triangle formed. L is the source, S is the sensor and P is the projection.

Solving equation (2) for d, we come up with equation (3) where $\beta = \frac{Wh}{4\pi}$.

$$d = \frac{\beta^{\frac{1}{3}}}{E^{\frac{1}{3}}} \quad (3)$$

Working on the robot-centric system, the coordinates of each sensor are as seen on Table I, where (x_s, y_s, z_s) is the position of the source. Using equation (3), the distance between the sensors and the beacon are shown below, with (E_0, E_R, E_L, E_0) being the measurement of the corresponding sensor.

TABLE I: Position of the sensors on the robot-centric system

Sensor	Coordinates (x,y,z)
S_0	(0,0,0)
S_R	(-r,0,0)
S_L	(r,0,0)
S_F	(0,-r,0)

$$\frac{\beta^{\frac{2}{3}}}{E_0^{\frac{2}{3}}} = x^2 + y^2 + z^2 \quad (4)$$

$$\frac{\beta^{\frac{2}{3}}}{E_R^{\frac{2}{3}}} = x^2 + y^2 + z^2 + r^2 - 2rx \quad (5)$$

$$\frac{\beta^{\frac{2}{3}}}{E_L^{\frac{2}{3}}} = x^2 + y^2 + z^2 + r^2 + 2rx \quad (6)$$

$$\frac{\beta^{\frac{2}{3}}}{E_F^{\frac{2}{3}}} = x^2 + y^2 + z^2 + r^2 - 2ry \quad (7)$$

By adding equations (5) and (6) we come up with (8):

$$(5) + (6) => 2(x^2 + y^2 + z^2) + 2r^2 = \frac{\beta^{\frac{2}{3}}}{E_R^{\frac{2}{3}}} + \frac{\beta^{\frac{2}{3}}}{E_L^{\frac{2}{3}}} \quad (8)$$

Subtracting equation (4) from (8) we are able to express β based on sensor measurements and the radius of the robot as seen in (9).

$$(8) - (4) => \beta^{\frac{2}{3}} = \frac{2r^2}{\frac{1}{E_R^{\frac{2}{3}}} + \frac{1}{E_L^{\frac{2}{3}}} - \frac{2}{E_0^{\frac{2}{3}}}} \quad (9)$$

Knowing the value of β, we are able to calculate y (10) by subtracting equation (4) from (7) and x (11) by subtracting (5) from (6). Finally the value of z (12) is given by replacing the values of y and x on equation (4).

$$(7) - (4) => y = \frac{r^2 - \beta^{\frac{2}{3}}(\frac{1}{E_F^{\frac{2}{3}}} - \frac{1}{E_0^{\frac{2}{3}}})}{2r} \quad (10)$$

$$(6) - (5) => x = \frac{\beta^{\frac{2}{3}}(\frac{1}{E_L^{\frac{2}{3}}} - \frac{1}{E_R^{\frac{2}{3}}})}{4r} \quad (11)$$

$$(4), (10), (11) => z = \sqrt{\frac{\beta^{\frac{2}{3}}}{E_0^{\frac{2}{3}}} - x^2 - y^2} \quad (12)$$

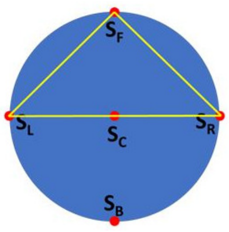
2) *Enhanced Scheme:* The basic localization scheme can be further enhanced by adding redundant sensors. The implementation of LadyBug uses a total of five sensors in order to create four groups (Group 1, Group 2, Group 3 and Group 4) as seen in Fig 3. Each one of these groups contains 4 sensors with the formation required by the *Basic Scheme* and is able to estimate the position of the source.

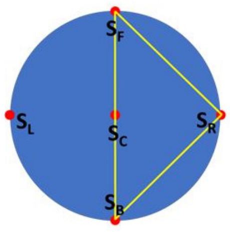
For instance, when Group 2 is considered, E_F comes from the reading of sensor S_R, E_L from S_F and E_R from S_B respectively because the virtual sensor triangle is rotated by 90° CW. The four estimates are then averaged in order to obtain a more precise estimation of the source's position. The complete localization algorithm can be seen on Alg.(2)

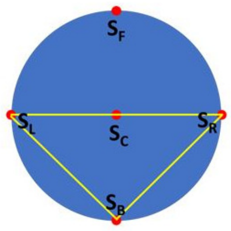
B. Navigation

Knowing the location of the beacon in the robot-centric system, the robot utilizes the navigation algorithm to reach it. While navigating to the source of the signal, the robot avoids obstacles found on its way. Obstacle avoidance assumes that the robot is equipped with a sensing medium, such as sonars, bumpers, IR sensors etc.

Two distinct behaviors were implemented on the navigation part of the algorithm: move to goal and circumnavigate. If no obstacle is detected, the robot moves towards the beacon.


(a) Group 1


(b) Group 2


(c) Group 3

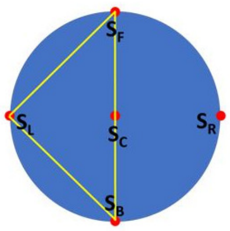

(d) Group 4

Fig. 3: Sensor groups

Algorithm 2: Find_Beacon _Location

```
1  $(x, y, z) \leftarrow 0$
2  foreach $SensorGroup \in Sensors$ do
3      $(E_0, E_F, E_L, E_R) \leftarrow read\_sensor\_intensity()$
               // Read intensity for center,
        front , left and right sensor
        respectively
4      $\beta^{\frac{2}{3}} \leftarrow (2 * radius^2)/((1/E_R^{\frac{2}{3}}) + (1/E_L^{\frac{2}{3}}) - (1/E_0^{\frac{2}{3}}))$
5      $y \leftarrow y + (radius^2 - \beta^{\frac{2}{3}} * (1/E_F^{\frac{2}{3}} - 1/E_0^{\frac{2}{3}}))/(2 * radius)$
6      $x \leftarrow x + (\beta^{\frac{2}{3}} * (1/E_L^{\frac{2}{3}} - 1/E_R^{\frac{2}{3}}))/(4 * radius)$
7      $z \leftarrow z + sqrt(\beta^{\frac{2}{3}}/E_0^{\frac{2}{3}} - x^2 - y^2)$
8  end
9  $(x, y, z) \leftarrow (x/4, y/4, z/4)$
10 return (x,y,z)
```

When an obstruction is detected between the robot and its path to the goal, the robot saves its current location and begins circumnavigating the obstacle. This behavior terminates when the robot reaches a place where no obstacle exists between the robot and its target and the distance to the beacon is smaller than the point the robot found the obstacle.

The *MoveToGoal* behavior of LadyBug is a simple controller that leads the robot to the goal while maintaining a constant translational speed (C_2 m/sec) and a rotational speed that is proportional to the angle between the robot's direction and the goal. The controller pseudocode can be seen on Alg.(3).

Circumnavigate is a controller that drives the robot to follow the contour of the obstacle while maintaining a safe distance from it (D_{safety}). In order to achieve this, it calculates two angles ϕ_{lin} and ϕ_{rot} that are combined to produce the rotational speed. ϕ_{rot} is the offset of the robot regarding the

Algorithm 3: MoveToGoal Controller

```
1  $\theta \leftarrow atan2(y, x)$        // Angle to Goal
2  $V_{rot} = \theta * C_1$
3  $V_{tran} = C_2$
```

desired safety distance from the obstacle, transformed to the $[-\pi/2, \pi/2]$ range through the *arctan* function. The distance between the robot's body and the obstacle is obtained by taking the minimum reading of the distance sensors. ϕ_{lin} is an approximation of the obstacle's contour based on the sensors readings. Firstly the algorithm finds the sensor with the minimum reading (closer to obstacle) s_{min}. Based on the reading of this sensor (reading(s_{min})), its distance from the center of the robot's body (R(s_{min})) and the vertical rotation of the sensor (angle(s_{min})), *Circumnavigate* estimates the closest point of the obstacle (x_1, y_1) in the robot-centric coordinates system. The next step is to check if on of the two sensors next to s_{min} (one to the left and one to the right) were also able to detect an obstacle. If this was the case the algorithm uses the same formula to estimate a second point on the obstacle (x_2, y_2) and uses the line segment between these two points as a guide for the obstacle's contour. If neither of the sensors next to s_{min} detected an obstacle, then the algorithm computes a vector vertical to s_{min}'s orientation as a guide for the obstacle's contour. The controller's pseudo code can be seen on Alg.(4)

Algorithm 4: Circumnavigate Controller

```
1  $s_{min} \leftarrow argmax(reading(s_i)) : iinSonars$
2  $x_1 \leftarrow (R(s_{min}) + reading(s_{min})) * cos(angle(s_{min}))$
3  $y_1 \leftarrow (R(s_{min}) + reading(s_{min})) * sin(angle(s_{min}))$
4  $s + next \leftarrow nil$
5  if $reading(S_{min+1}) < \infty$ then
6      $s_{next} \leftarrow S_{min+1}$
7  end
8  else if $reading(S_{min-1}) < \infty$ then
9      $s_{next} \leftarrow S_{min-1}$
10 end
11 if $s_{next} = nil$ then
12     $\phi_{lin} \leftarrow atan2(x_1, -y_1)$
13 end
14 else
15     $x_2 \leftarrow$
         $(R(s_{next}) + reading(s_{next})) * cos(angle(s_{next}))$
16     $y_2 \leftarrow$
         $(R(s_{next}) + reading(s_{next})) * sin(angle(s_{next}))$
17     $\phi_{lin} \leftarrow atan2(y_2 - y_1, x_2 - x_1)$
18 end
19 $\phi_{rot} \leftarrow atan(reading(s_{min}) - D_{safety})$
20 $v_{rot} \leftarrow (\phi_{lin} + \phi_{rot}) * C_3$
21 $v_{tran} \leftarrow cos(v_{rot}) * C_4$
```

V. Experimental Results

In order to evaluate the performance of the *LadyBug* algorithm and provide an empirical comparison against the *I-Bug* algorithm, we implemented both algorithms in the Webots Open Source Robot Simulator by [10]. We used a generic disc shaped robot, with 40 cm diameter, equipped with differential drive kinematics and a belt of eight distance sensors located on the perimeter of the robot's body. The tower emitting signals that served as the goal of both robots was a simple point light placed at 1.5 m above the ground and all the tower related sensors were implemented using a group of light sensors placed on top of the robot body.

LadyBug used five light sensors: one at the center, one at the front, one at the back and two at the left and right side of the robot. *I-Bug* used the same distance sensors in order to navigate around the obstacles, instead of the naive approach using bumpers. The intensity sensor was implemented using a light sensor at the center of the robot and the tower alignment sensor was implemented using 12 light sensors located on the perimeter of a 2cm circle in order to simulate the original IR-Seeker with 9 regions plus the dead zones [11]. The virtual tower alignment sensor compares the reading of all 12 sensors and returns true when the highest intensity is read from sensor number 9 (the one facing at the front). In all experiments both robots were stopped when their distance from the projection of the light on the floor, would get below 40 cm (one body size).

Four experiments were conducted in order to validate the assumptions stated in the *Motivation* section.

A. Shorter Paths

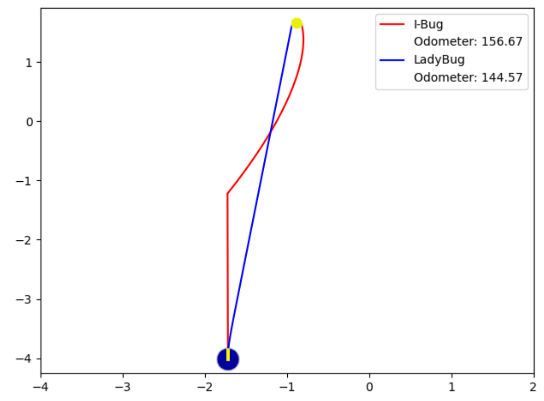
The first assumption was that due to *LadyBug* being able to know at real time both its starting and goal point it would follow a shorter path than *I-Bug* in worlds without obstacles. Since the robot is able to rotate in place (a behaviour extensively used by *I-Bug*) the best way to compare the total distance travelled by the robots were to use the odometer readings from both wheels and take the average.

We simulated three worlds with the robot placed approximately 6 meters away from the goal (projection of the point lamp). In all three worlds the starting and goal positions were the same and we just changed the relative direction towards the goal (World 1 at 10°, World 2 at 80° and World 3 at -50°). Figures 4a, 5a and 6a present the travelled paths and the average odometer readings for worlds 1,2 and 3 respectively.

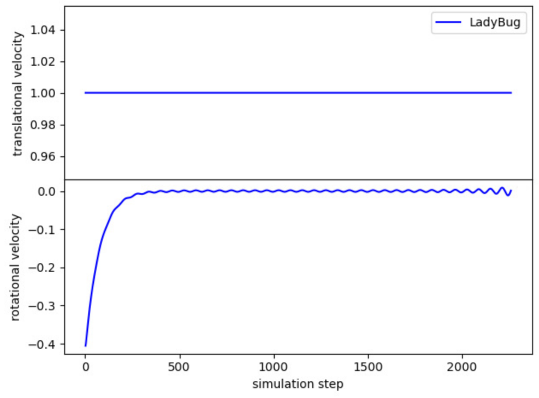
It is clear from the results that *LadyBug* travelled shorter paths in all three cases (approximately 7% shorter on average). The second conclusion drawn from the odometer readings is that the performance of *I-Bug* seems to depend more on the robot's initial orientation (The stdev of the *I-Bug* readings were approximately double compared to the ones by *LadyBug*).

B. Smoother Trajectories

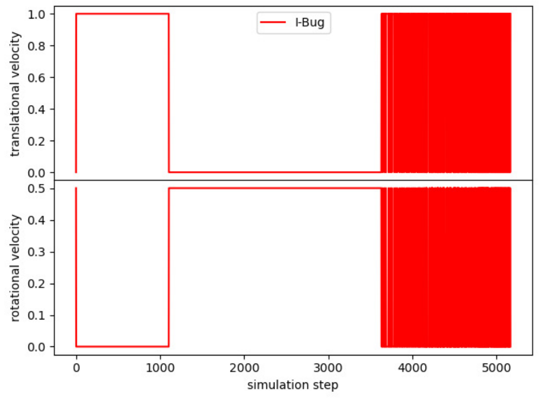
The second assumption was that apart from travelling shorter paths in obstacle-free worlds, *LadyBug* would be able to move in smoother trajectories than *I-Bug*. In order



(a) Path followed

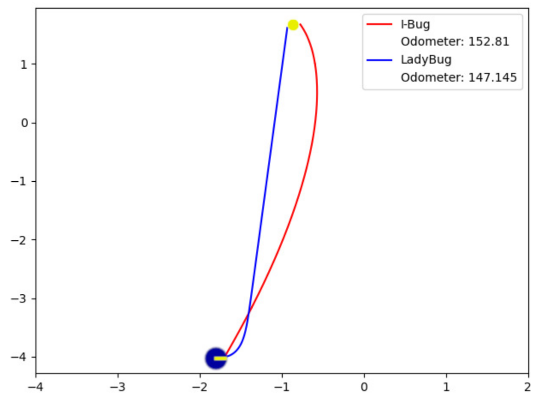


(b) LadyBug Velocities

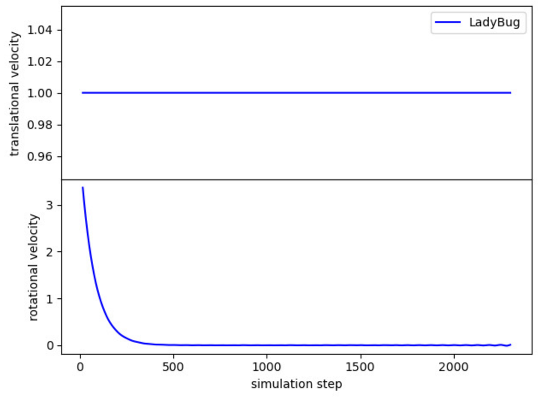


(c) I-Bug Velocities

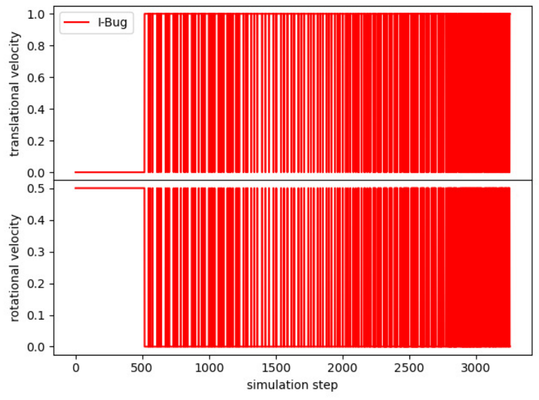
Fig. 4: Path and velocities for World 1



(a) Path followed

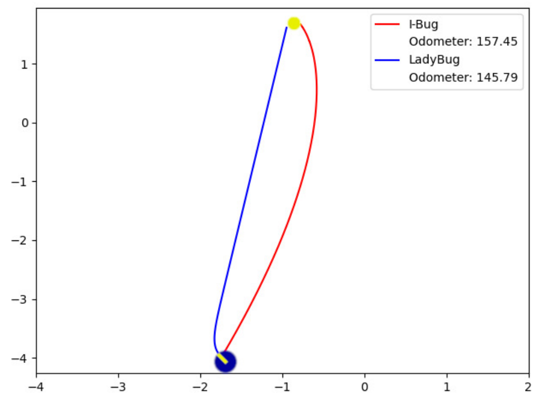


(b) LadyBug Velocities

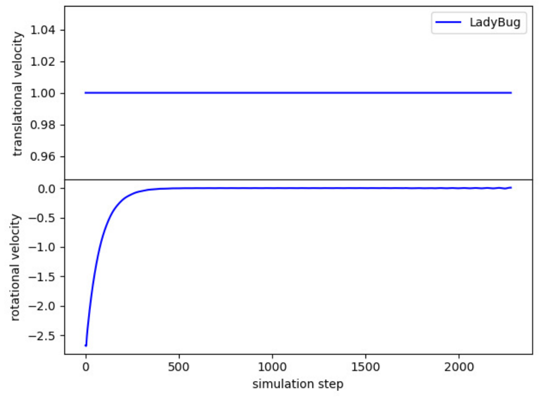


(c) I-Bug Velocities

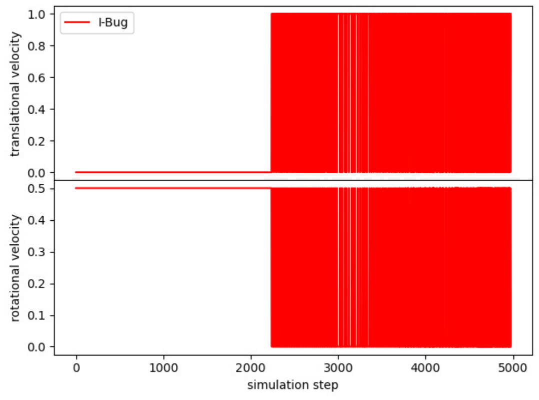
Fig. 5: Path and velocities for World 2



(a) Path followed



(b) LadyBug Velocities



(c) I-Bug Velocities

Fig. 6: Path and velocities for World 3

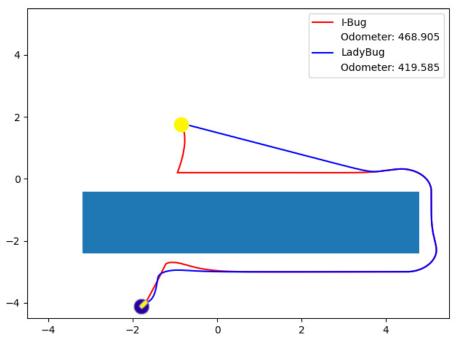


Fig. 7: Path for World 4

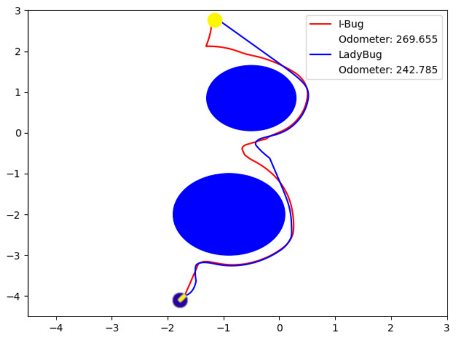


Fig. 8: Path for World 5

to validate this we recorded the rotational and translational velocities from both robots in the three worlds (World 1, World 2 and World 3) of the previous experiment. The resuts can be seen in Figures 4b, 5b and 6b for *LadyBug* and in Figures 4c, 5c and 6c for *I-Bug* respectively.

It is obvious from the results that *I-Bug* had a lot of steep changes in both rotational and translational velocities, which occur when the robot realizes it is not facing the goal anymore and it has to come to a complete stop, activate the in place rotation and then stop the rotation and start moving again. On the other hand, *LadyBug* knowing the exact position of the goal is able to adopt a simple but efficient controller for moving towards the goal that produces much smoother changes in rotational speed and a constant translational one.

C. Earlier Leaving Points on Obstacles

The third assumption was that although both robots use the same circumnavigate behaviour when they come across obstacles, *LadyBug* would be able to leave the obstacle and travel towards the goal earlier than *I-Bug* and thus produce a shorter path in total.

I order to verify the above we simulated two worlds (World 4 and World 5) with obstacles of different size and shape and we recorded the paths travelled and the average odometer for both robots. The results presented in Fig. 7 and Fig. 8 show that *I-Bug* followed paths that were on average 11% longer than the ones by *LadyBug*. This is due to the fact that *I-Bug* has to detect a local maximum in the RSSI value in order to leave the obstacle, while *LadyBug* checks if the obstacle is blocking the line segment between the robot and the goal.

D. Noise Resilient Goal Detection

The last assumption was that a sensor returning the exact position of the goal offers more capabilities for logical redundancy in sensing than a tower detection sensor returning boolean values. Therefore *LadyBug* would present a more robust behaviour in noisy environments than *I-Bug*.

In order to evaluate the above, we created a simple obstacle-free world (similar to World 1) and we simulated both robots with 7 increasing levels of noise in the light sensors readings. We started with a noise-free environment and in the 7th run the noise in the sensors reading's was 50%. For each noise level we recorder the path followed by the two robots and the average odometers measurements. Fig. 9a and Fig. 9b present the results.

LadyBug clearly outperformed *I-Bug* in noisy environments since the latter followed paths that were, on average, 61% longer. This was due to the fact the the tower detection sensor was unable to correctly detect if the robot was facing the goal especially as it came closer to the light source, resulting in the spiral movement shown in 9b. On the contrary, noise seemed to affect the goal identification system of *LadyBug* especially when the robot was farther from the light source, resulting in paths diverging in their first half and converging at the second one.

The second conclusion that can be drawn from the odometer readings is that as the noise level increases the behaviour of *I-Bug* deteriorates in a more aggressive manner than *LadyBug*. Fig. 9c presents the average odometers of the two robots as the noise level increases and there is a clear difference in the slope of the two lines. The slope of *I-Bug* is 2.5 times larger than the one of *LadyBug* at the biggest level of noise.

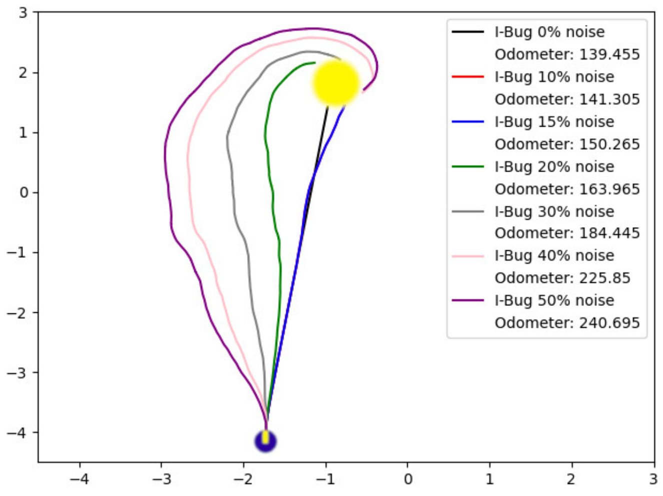
VI. Conclusions

In this paper a novel Bug Algorithm, called *LadyBug* was proposed. LadyBug has two distinct functions: Localization and Navigation. The algorithm is able to localize a source of an electromagnetic signal based on RSSI and navigate to it.

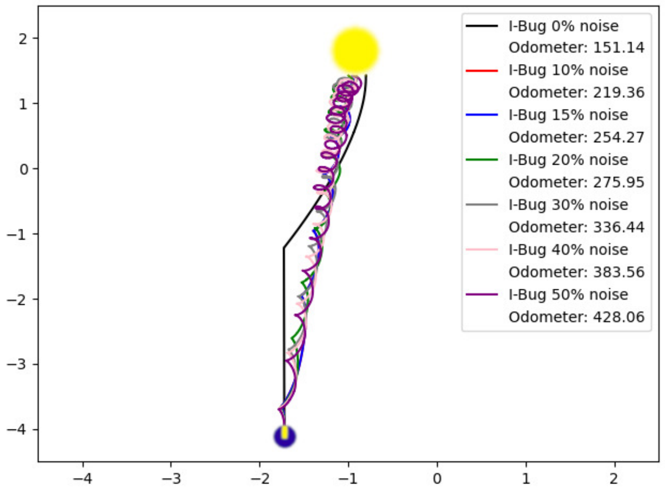
Our approach was compared on a simulated environment with *I-Bug*, a bug algorithm based on intensity as well. The proposed algorithm produced shorter paths and smoother trajectories. Even when noise was introduced on the experiments, *LadyBug* presented a more robust noise resilient behaviour and clearly outperformed *I-Bug*.

Although results from *LadyBug* were promising there is still further work we could do in order to improve its performance. The next step would be to implement the same localization technique in a navigation scheme that makes better use of distance sensor readings, i.e. similar to *tangent bug's* approach. This would reduce the time spent on following the boundaries of the obstacles and result in even shorter paths.

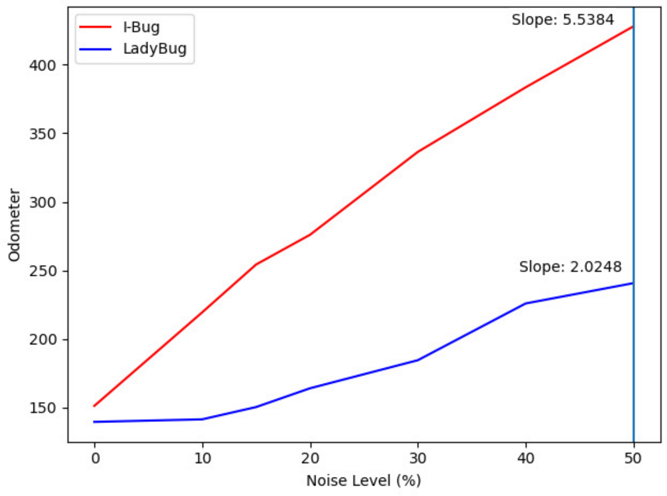
Our approach was only tested on a simulation environment. In near future an implementation of the *LadyBug* algorithm will be realized on a real robot. This will allow us to



(a) Path followed by LadyBug



(b) Path followed by I-Bug



(c) Odometer readings per noise levels

Fig. 9: Paths for noise

evaluate the performance of our algorithm in real conditions. Evaluating the localization part of *LadyBug* in non simulated environment is also critical. Real world has noise that can't be simulated easily (i.e. reflection of the electromagnetic signal, non-uniform noise etc.)

Additionally, small obstacles are harder to detect with the current configuration. Although laser sensors can be placed as low on the robot's body as possible, there will still be an area which is not covered. In order to detect obstructions located under the field of view, bumper sensors could be employed in future implementations.

Another important part of our future work is the generalization of the localization algorithm. In our experiments we used a source of light and a robot equipped with light sensors. Using those sensors on a real robot is tricky. A light wave is susceptible to reflection and can't pass through solid materials such as walls. Also the noise levels will make localization a difficult, if not impossible, task. For that reason, the generalization and experimentation with different sources (radio, Bluetooth etc.) lies in our goals.

References

[1] X. Luo, W. J. O'Brien, and C. L. Julien, "Comparative evaluation of received signal-strength index (rssi) based indoor localization techniques for construction jobsites," *Advanced Engineering Informatics*, vol. 25, no. 2, pp. 355 – 363, 2011, information mining and retrieval in design. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474034610000984

[2] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, Sep. 2017.

[3] M. Shahkhir Mozamir, R. B. A. Bakar, and W. I. S. W. Din, "Indoor localization estimation techniques in wireless sensor network: A review," in *Proceedings - 2018 IEEE International Conference on Automatic Control and Intelligent Systems, I2CACIS 2018*, 2019.

[4] V. Lumelsky and A. Stepanov, "Dynamic path planning for a mobile automaton with limited information on the environment," *IEEE Transactions on Automatic Control*, vol. 31, no. 11, pp. 1058–1063, November 1986.

[5] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1, pp. 403–430, 1987. [Online]. Available: https://doi.org/10.1007/BF01840369

[6] A. Sankaranarayanan and M. Vidyasagar, "A new path planning algorithm for moving a point object amidst unknown obstacles in a plane," in *Proceedings., IEEE International Conference on Robotics and Automation*, May 1990, pp. 1930–1936 vol.3.

[7] K. Taylor and S. M. LaValle, "I-bug: An intensity-based bug algorithm," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 3981–3986.

[8] K. McGuire, G. de Croon, and K. Tuyls, "A comparative study of bug algorithms for robot navigation," *Robotics and Autonomous Systems*, vol. 121, p. 103261, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889018306687

[9] I. Kamon, E. Rivlin, and E. Rimon, "A new range-sensor based globally convergent navigation algorithm for mobile robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, 1996, pp. 429–435 vol.1.

[10] O. Michel, "Webots: Professional mobile robot simulation," *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39–42, 2004. [Online]. Available: http://www.ars-journal.com/International-Journal-of- Advanced-Robotic-Systems/Volume-1/39-42.pdf

[11] K. Taylor and S. M. LaValle, "Intensity-based navigation with global guarantees," *Autonomous Robots*, vol. 36, no. 4, pp. 349–364, 2014. [Online]. Available: https://doi.org/10.1007/s10514-013-9356-x