Improved Bug Algorithms for Robot Navigation in Unknown Environments

Clement Olufayo Oni and Haiying Wang

School of Computing, Ulster University, Belfast

Abstract – Automation is at the core of current technology development, and it is becoming increasingly prevalent, hence the need to investigate and develop simple systems/algorithms for dependable and efficient implementations. Autonomous robots designed to navigate unfamiliar environments successfully are often needed in automation systems. There have been many approaches to autonomous robot navigation systems. These include single-robot navigation and multi-robot coordination systems. Simplicity, effectiveness, and efficiency are paramount factors to be considered as this field grows. Multi-robot coordination is now a popular subject in robotism. This research conducted a constructive investigation of bug algorithms, and then developed and implemented a technique for optimizing some of the current algorithms. Two advanced simple navigation algorithms, with high effectiveness and efficiency, were thereby developed by this project. One is a single-robot algorithm which is an improvement to an existing algorithm. The other is a dual-robot navigation system (based on two different algorithms) that could be implemented for any number of robots.

Keywords: Bug algorithm, robot navigation, multi-robot

1. Introduction

Automation is central to the current technology development terrain, and robotics' role in automation cannot be overemphasized. Today, its importance and application are visible in virtually all human endeavors and the mode of growth has promised greater pervasiveness for tomorrow [1] [2]. Hence there stands an obvious need to develop simple, effective, and efficient implementation techniques. Due to its versatility and essentiality to current development: *simplicity* is essential to keep the system light and accommodate other tasks [3], *effectiveness* is paramount as it is becoming closely united with daily living, and *efficiency* is the economics that makes it a better choice.

One of the big subjects in robotics is navigation which has been attempted and implemented in different ways. One of the popular approaches is the Bug algorithm due to its comparative simplicity [4] [5]. Bug algorithms are online, non-heuristic algorithms that depend on the robot sensor input for local knowledge of the environment, on which navigation action is based. In general, bug algorithms are characterized by the following: knowledge of the starting point and the goal, straight-line movement toward the goal (Go-to-Point), obstacle detection/avoidance, and wall-following behaviors. Many versions of Bug Algorithms exist. Some of these are the Common Sence Algorithms (Com and

Com1), Bug1, Bug2, Alg1, Alg2, DistBug, Rev1, Rev2, and Distance Histogram Bug (DH-Bug). Based on their operation strategies, they are sometimes classified as Angle-Bugs, M-Line-Bugs, Distance and/or Hit-Point Bugs, Gange-Bugs, and Special Bugs. The recent development in this field is directed toward multi-robot (swarm) coordination to achieve a common goal [6] [7] [8].

The main contributions in this paper include:

1. The proposal of an innovative bug algorithm that improves the existing Rev2 algorithm, hereafter called **Rev3**.
2. The proposition of an approach for improving the performance of some existing bug algorithms.
3. The development of a simple but efficient multi-robot navigation algorithm hereafter called **Com1-Rev3**.

The paper has five sections. Section 1 introduces the concept of the paper. A review of related works is given in section 2. Section 3 presents the techniques behind the developed algorithms, while section 4 presents practical applications of the algorithms. Lastly, section 5 gives the summary and conclusion.

2. Related Works

There are many bug algorithms, but the following are briefly reviewed as they will be referenced later.

- *Bug0* (also known as Com) [9] moves the robot towards the goal and stops when the goal is reached. If an obstacle is encountered, it turns left or right and goes around the obstacle until heading towards the goal is possible again.
- *Bug1* [9] is a non-greedy algorithm that drives towards the goal and circumnavigates any obstacle it contacts to get the closest point to the goal along the obstacle boundary. It then returns to the nearest point to depart from the obstacle boundary and move towards the goal.
- *Bug2* [9] generates an imaginary line called 'm-line' connecting the starting and goal points along which the robot moves towards the goal until an obstacle is encountered. It departs from the obstacle when the line is re-encountered at a point closer to the goal, otherwise, the algorithm fails.
- *Com1* [4] drives towards the goal. When an obstacle is encountered, it follows the boundary until heading towards the goal is possible and the robot is closer to the goal than the last hit-point.
- *Rev2* [4] [10] moves the robot towards the goal and stops when the goal is reached. If an obstacle is encountered, it alternates its turning mode and follows the boundary until heading towards the goal is possible again.

Coordination of multiple robots to achieve a common goal is becoming popular by the day and it has been achieved via different approaches. A recent implementation, based on Bug1, carried out the circumnavigation by two robots traveling in opposite directions along the contour of the obstacle. They both keep track of their closest points to the goal and on meeting, they compare their closest points to know the actual closest point. They then return to the closest point to leave the obstacle and go straight to the goal. The travel time is reduced due to the two robots' involvement and the total distance travel is reduced due to a reduction in the distance back to the point, on the contour, closest to the goal [6]. In another development, multi-bug path planning (MBPP) in a known static environment was introduced using distance transform, line of sight, and m-line principles while generating a new robot at every hit-point [7]. Similarly, swarm algorithms based on Com (SwarmCom), Bug1 (SwarmBug1), and Bug2 (SwarmBug2) were implemented and SwarmBug1 was found to have the best performance among them [8].

3. Methodology

3.1 Discovery and Implementation of Rev3 Algorithm

Fig. 1 shows a navigation environment with four start and goal points (A, B, C, and D) to illustrate findings from studies on some bug algorithms and improvements implemented. The obstacle is the shape in blue, while navigation paths are black arrows and lines. We use a 1m clearance between the robot and the wall during wall following as the robots should not collide with the obstacle at any point. This means that when a robot is 1m from the obstacle, its navigation state changes to wall-following. The figure assumes a perfect (90^o) turning at every hit point and corner. It also assumes a perfectly straight-line movement between two points. All illustrations use a left-turning robot except where the algorithm implements both turnings. Table 1 presents the bug algorithms' navigation paths and approximate distance traveled. The shaded cells represent the instances where the bug failed to reach the goal.

- The choice of turning direction at each hit-point is a significant factor in a bug algorithm performance.
- Bug1 is known for its exhaustive (all possibilities checking) behavior, making it virtually dependable (effective) in all scenarios, though with an efficiency trade-off. Some other single-robot bug algorithms, like Com1, can compete with it in terms of effectiveness, with better time and distance travel efficiency. Bug1, Bug2, and Com1 are effective in all scenarios including the ones not presented here. The distance traveled and time taken by Bug1 is always the highest.
- **No navigable path condition** (contacting the last hit-point) was added to Com1 to suit the need of this research work.
- Rev2 traveled comparably short distances in its efficient environments. Nevertheless, it failed occasionally. For instance, it is **trapped** in the structure while navigating from A to D in Fig. 1 (see Table 1 for the navigation path). A closer look was taken at Rev2 algorithm to investigate an approach to failure elimination or alleviation. This was realized by adding memory to the algorithm to

keep the last two hit-points and if any of them is re-encountered, the navigation state changed to what we call '**wall-following to a better take-off**'. In this state, the robot follows the obstacle contour until its distance to the goal is less than that of the closest recorded point to the goal. **No navigable path condition** was also established: contacting the last-hit point while in the 'wall-following to a better take-off' state.

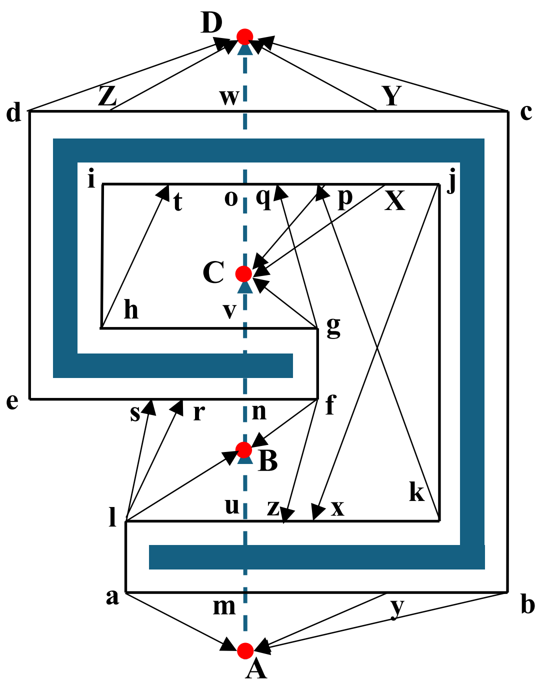


Fig. 1. A navigation environment illustration

- The resultant navigation algorithm is **Rev3** as shown in Fig. 2. The navigation paths for Rev3 are also given in Table 1.
- Rev3 behaves exactly like Rev2, except where Rev2 fails.
- By extension, Bug0 could also be likewise improved. Let it be noted that the last hit point will not be sufficient most of the time, but the previous two are adequate for memory consideration. For instance, using only the last hit point on Rev2, navigating from A to D, in Fig. 1 will not change anything. Likewise, using the previous hit-point only on Bug0, navigating from B to C will require the robot to orbit the obstacle twice, but only once when the last two hit points are used.
- With respect to Fig. 1 map, Rev3 has around 16% better efficiency than Com1, the best of the existing single robot algorithms assessed here.

Table 1. Paths taken, and distance traveled by bug algorithms for different scenarios in Fig. 1. Columns 2 to 7 headings were formed with the start point, an arrow, and the goal; and they give the sequence of points passed through during navigation scenarios and the approximate distance traveled. For instance, A → B is the heading for navigation from point A to point B. P = Obstacle Perimeter traced by the robot. The table is divided into parts I and II for space and clarity.

Part I: Starting points A and B

	A → B	**A → C**	**A → D**	**B → A**	**B → C**	**B → D**
Bug0 path	AmalB	Amalredcbals….	AmalsedD	Bukjxk…	Bnedcbalsedwc….	BnedD
Bug0 distance	17.83	**Failed**	43.59	**Failed**	**Failed**	32.49
Bug1 path	AmPmaluB	AmPmalkjihvC	AmPmalkjihgfedwD	BuPukjihgfedcbmA	BnPnedcbalkjihvC	BnPnedwD
Bug1 distance	159	203	243	272	273	174
Bug2 path	AmaluB	AmalunedcbalkjoC	AmalunedwD	BukjihgfedcbmA	BnedcbalkjoC	BnedwD
Bug2 distance	20	141	55	133	121	35
Com1 path	AmalB	AmalredcbalkjpC	AmalsedD	BukjihgfedcbyA	BnedcbalkjpC	BnedD
Com1 distance	17.83	132.09	43.59	130.25	119	32.49
Rev2 path	AmalB	AmalrfgC	Amalsfgqihtjkpi….	BukjxlaA	BnedcbalrfgC	BnedD
Rev2 distance	17.83	29.41	**Failed**	57.36	99.41	32.49
Rev3 path	AmalB	AmalrfgC	AmalsfgqihtjklabcD	BukjxlaA	BnedcbalrfgC	BnedD
Rev3 distance	17.83	29.41	137.39	57.36	99.41	32.49

Part II: Starting points C and D

	C → A	**C → B**	**C → D**	**D → A**	**D → B**	**D → C**
Bug0 path	Cvgfzkjxk…	CvgfB	Coihti…	DwcbA	DwcbalB	Dwcbalredw…
Bug0 distance	**Failed**	11.61	**Failed**	45.7	58.83	**Failed**
Bug1 path	CvPvgfedwcbmA	CvPvgfnB	CoPoihgfedwD	DwPwcbmA	DwPwcbaluB	DwPwcbalkjihvC
Bug1 distance	226	152	203	188	200	244
Bug2 path	CvgfnukjihgfedcbmA	CvgfnB	CoihgfedwD	DwcbmA	DwcbaluB	DwcbalkjoC
Bug2 distance	146	13	64	49	61	92

Com1 path	Cvgfzkjih gfedcbyA	CvgfB	CoihgfedZD	DwcbA	DwcbalB	DwcbalkjX C
Com1 distance	138.97	11.61	61.8	45.7	58.83	88.25
Rev2 path	Cvgfzkjxl aA	CvgfB	Coihtjkpih…	DwcbA	DwcbalB	Dwcbalrfg C
Rev2 distance	66.08	11.61	**Failed**	45.7	58.83	70.41
Rev3 path	Cvgfzkjxl aA	CvgfB	CoihtjklabcY D	DwcbA	DwcbalB	Dwcbalrfg C
Rev3 distance	66.08	11.61	108.51	45.7	58.83	70.41

Note: Navigation path is the sequence of points passed through during a navigation scenario. For instance, 'AmalB' is the navigation path taken by Bug0, Com1, Rev2, and Rev3 while traversing from point A to point B.

Total distance by Bug1 = 2,537m, total distance by Bug2 = 930m, total distance by Com1 = 880.41m, and total distance by Rev3 = 735.03m.

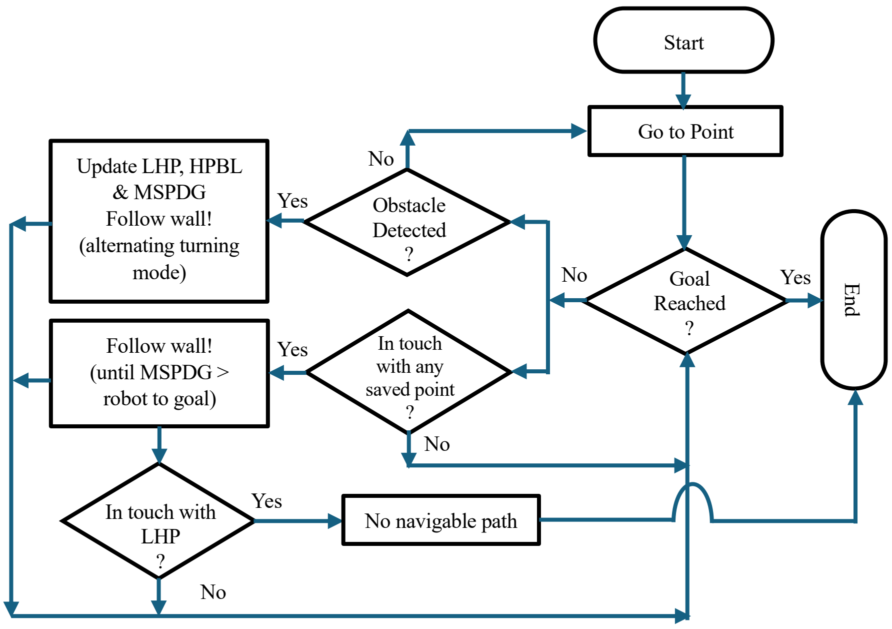

Start
Go to Point
Goal Reached ?
Yes
End
No
Obstacle Detected ?
No
Yes
Update LHP, HPBL & MSPDG Follow wall! (alternating turning mode)
In touch with any saved point ?
Yes
No
Follow wall! (until MSPDG > robot to goal)
In touch with LHP ?
Yes
No navigable path
No

Fig. 2. Rev3
LHP – Last Hit-Point
HPBL – Hit Point Before LHP
MSPDG – Minimum Saved Point Distance to Goal

Rev3 is a new single-robot algorithm that drives the robot toward the goal until an obstacle is encountered. It alternates its turning mode at each hit-point. If one of the last two hit-points is encountered, it switches its navigation mode to "follow-wall-to-a-better-take-off", the state in which it continues to follow the wall until there is a free heading towards the goal and it is closer to the goal than the closest saved point to the goal. Rev3 is an improved version of Rev2 that is effective in most situations where Rev2 fails by keeping track of its last two hit points. It has high efficiency in most situations.

3.2 Implementation of Com1-Rev3

We desire a simple, highly effective, and efficient navigation algorithm for dynamic environments. In the context of this paper, these terms are defined thus [4, 5]:

- The *simplicity* of an algorithm is the degree to which the algorithm is straightforward, light (less bulky), easy to understand, and implement.
- The *effectiveness* of a navigation algorithm is the probabilistic measurement of its successful navigation from any starting point to any target in an environment. This measurement is 1 (100%) if the algorithm always succeeds. For a single scenario, the effectiveness of a bug algorithm is 1, if the algorithm navigates successfully from the starting point to the goal; but 0, if the algorithm fails.
- The *efficiency* of a navigation algorithm is the comparative measurement of how little are the resources (computation, memory, training, travel time, distance travel, energy expenditure et cetera) needed to accomplish a given navigation. In this context, this term is just a comparative quantity that states which of two algorithms is preferred with respect to the time taken, distance traveled, and energy expenditure. It is assumed that the three parameters are directly proportional. Considering a single scenario, an ineffective algorithm has no (zero) efficacy because it will not reach the goal no matter how long it runs. In other words, an algorithm must be effective in a scenario to be efficient.

In this paper, we implement two robots operating with two different algorithms, Com1 and Rev3 due to the following:

- Keeping the design simple is important, so increasing the number is unnecessary if we can get a two-algorithm combination suitable for virtually all environments. Also, the fewer robots moving at a time, the lower the robot's collision chances.
- Com1 and Rev3 are greedy algorithms that navigate comparably short paths in their efficient environments.
- The Com1-Rev3 combination reduces the possibility of robot collision because they behave alike when departing from obstacles.
- Rev3 alternates its local turning direction at every new hit-point while Com1 maintains a single turning mode (left turning in this study). Based on this, at least one of them will be highly efficient in most (if not all) cases.
- The combination was tested in different environments. It performs outstandingly in most cases.

The two robots are coordinated so that one implements left turning at the start, while the other implements right turning (or both start with the same turning mode, as in the example here). Likewise, the counterpart is summoned to stop moving when the robot that will reach the goal first is known to reduce energy expenditure. To be certain a robot will get to the goal before its counterpart:

- It must be in the 'go-to-point' state (free-heading toward the goal).
- There must be no obstacle between the robot and the goal.
- It must be closer to the goal than its counterpart.

For starting point, A and goal point, D in Fig. 1, both travel along points A, m, a, l, and s. At point s, the Com1 robot turns left (best in this situation), while the Rev3 robot follows the right. When the Com1 robot gets to point d, the three conditions above are satisfied, Rev3 robot stops. Com1 continues to point D and the navigation process ends.

3.3 Collision control

Collision control was introduced to the Com1-Rev3 algorithms as shown in Fig. 3. Ordinarily, each robot sees its counterpart as an obstacle and navigates around it, the collision control is necessary for the Rev algorithm since it changes turning direction at every hit-point. Not to see the com1 robot as an obstacle (a condition that will alter its performance and make it unpredictable), when the Rev3 robot is less than 1.5m from the com1 robot, it stops moving to allow the com1 robot to pass. It then continues moving when they are 2m apart. It should be noted that having enough space for the two robots to travel without encroaching on one another is essential.

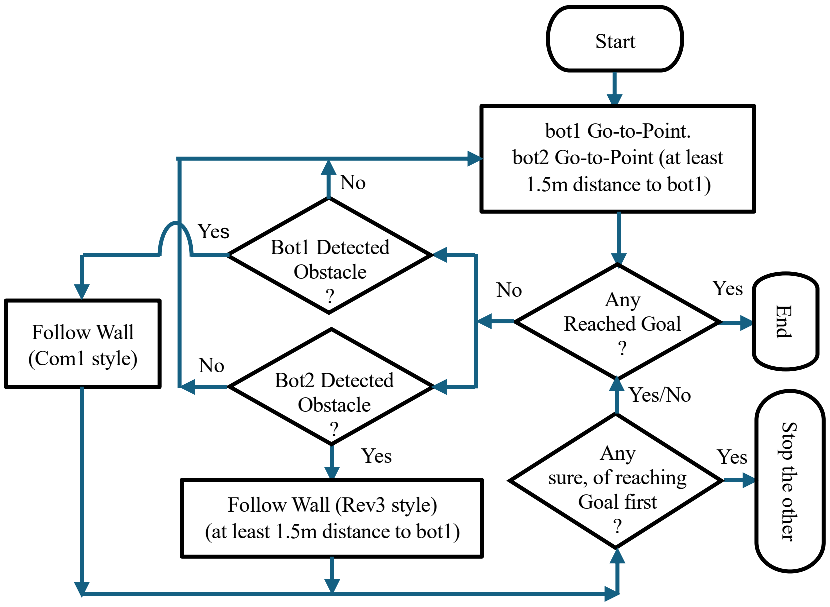
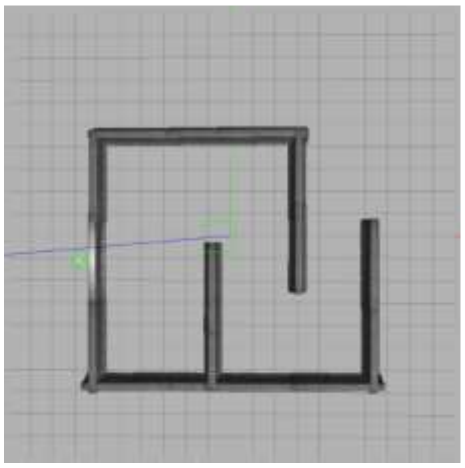


Fig. 3. Com1-Rev3

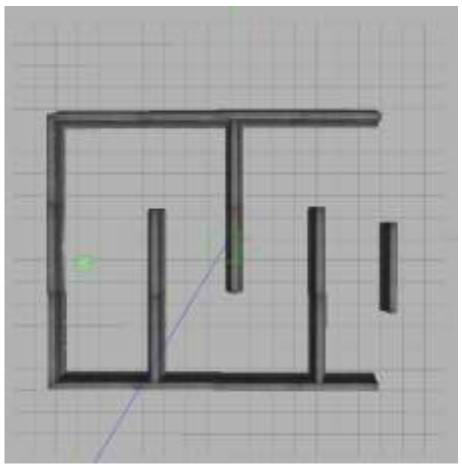
4. Results and Discussions

4.1 analysis of Single Bug Algorithms

To further confirm the performance of the Rev3 algorithm, Bug2, Com1, Rev2, and Rev3 were tested with the Gazebo simulated maps shown in Fig. 4 with three start and goal points (A, B, and C). The distance traveled in each navigation scenario and the aggregate for each algorithm are given in Table 2.



(a)



(b)

Fig. 4. Simulated maps for algorithm testing

Tables 2. Distance traveled by bug algorithms

Map (a): A = (9, -1), B = (-4, -1), C = (-9, -1)

	A → B	**A → C**	**B → A**	**B → C**	**C → A**	**C → B**	**Total**
Bug2	79.0	39.3	58.2	64.8	41.7	52.5	**335.5**
Com1	146.4	37.6	56.3	63.1	36.2	52.4	**392.0**
Rev2	70.4	37.5	23.3	Failed	35.3	44.3	∞
Rev3	70.7	37.1	23.3	58.5	35.2	44.9	**269.7**

Map (b): A = (9, -1), B = (-6, -1), C = (-10, -1)

	A → B	**A → C**	**B → A**	**B → C**	**C → A**	**C → B**	**Total**
Bug2	102.9	46.9	74,6	73.5	61.3	67.0	**426.2**
Com1	182.9	41.6	56.4	72.2	41.6	66.2	**469.9**
Rev2	38.9	Failed	36.4	Failed	40.4	Failed	∞
Rev3	39.1	106.7	36.5	72.6	40.1	124.7	**419.7**

It is observed that:

- Overall distance traveled: Bug2 = 761.7m, Com1 = 861.9m, Rev3 = 689.4m.
- Rev2 and Rev3 distance traveled were approximately the same, except where Rev2 failed.
- Among the three algorithms that succeeded in all scenarios, Rev3 has the highest distance efficiency, followed by Bug2. In Fig.1, Com1 takes the second position. Rev3 is 31.2% and 10.7% more efficient than Com1 in maps (a) and (b) respectively. Considering both maps, Rev3 is 20.0% more efficient than Com1. Rev3 is 19.6% and 1.5% more efficient than Bug2 in maps (a) and (b) respectively. Considering both maps, Rev3 is 9.5% more efficient than Bug2. Considering the map in Fig. 1, Rev3 is 16.5% and 21.0% more efficient than Com1 and Bug2 respectively. This shows that bug algorithms' efficiencies are map-dependent but for these maps and others implemented, Rev3 has better efficiency.
- Though Rev3 most of the time performed better than Com1, there are instances where Com1 traveled shorter distances. For instance, navigating from point A to C in map (b). This is also true for Bug2.

4.2 Comparing Com1-Rev3 with other Multi-robot Systems

In his 2020 presentation, Kandathil et al [6] developed a multi-robot system inspired by the Bug1 algorithm. Let us compare it with Com1-Rev3. Consider Fig. 5 for multi-robot systems with only two robots for illustrations. The arrows represent directions and paths. It should be noted, in practical demonstrations, that robots' navigation paths are not perfectly straight lines and the two are not tracing the same line. The starting point P and goal Q could be shifted left or right. Let all robots travel at the same speed, then travel time is directly proportional to the distance covered. Let us also assume that the distance covered is directly proportional to energy expenditure.

For case (a), the best is the situation where L (the closest point to the goal) and M (the meeting point) coincide. This happens when the line from start (S) to goal (G) divides the obstacle into two equal parts. It is the worst scenario for case (b) because if the points move away from S and G (to P and Q for instance), the distance traveled by each robot in (b) reduces.

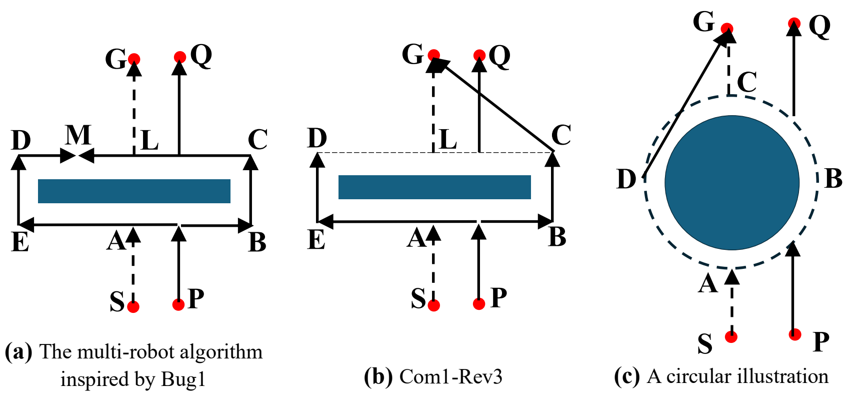

G
Q
D
M
L
C
E
A
B
S
P
(a) The multi-robot algorithm inspired by Bug1
G
Q
D
L
C
E
A
B
S
P
(b) Com1-Rev3
G
Q
C
D
B
A
S
P
(c) A circular illustration

Fig. 5. Paths taken by multi-robot systems

We will use points S and G (the best for case (a)). Let the approximate path traveled by the two robots be X_1 and X_2 for (a) and Y_1 and Y_2 for (b), then:

$$X_1 = SA+AB+BC+CL+LG \text{ - - - - - - - - - - - - - } (1)$$

$$X_2 = SA+AE+ED+DL \text{ - - - - - - - - - - - - - - - - } (2)$$

$$Y_1 = SA+AB+BC+CG \text{ - - - - - - - - - - - - - - - - } (3)$$

$$Y_2 = SA+AE+ED \text{ - - - - - - - - - - - - - - - - - - - } (4)$$

Note that

$$ED = BC$$

$$AB = AE = DL = CL = \text{half the length of the obstacle}$$

$$\text{Total Distance in (a)} = X_1+X_2 = SA+SA+AB+AE+CL+DL+BC+ED+LG$$

$$\text{Total Distance in (a)} = X_1+X_2 = 2(SA)+4(AB)+2(BC)+LG \text{ - - - - - - - - - - } (5)$$

$$\text{Total Distance in (b)} = Y_1+Y_2 = SA+SA+AB+AE+BC+ED+CG$$

$$\text{Total Distance in (b)} = Y_1+Y_2 = 2(SA)+2(AB)+2(BC)+CG \text{ - - - - - - - - - - } (6)$$

Comparing equations (5) and (6):

$$\text{Total Distance in (a)} - \text{Total Distance in (b)} = 2(AB)+LG - CG \text{ - - - - - - - - } (7)$$

The starting point and the goal could also be moved away or close to the obstacle. SA is not represented in equation (7), hence moving S does not affect it. The closer G is to the obstacle, the better for (a) but the contrary for (b). As LG tends to zero, CG tends to AB+LG, and equation (7) is reduced to:

$$\text{Total Distance in (a)} - \text{Total Distance in (b)} = AB \text{ - - - - - - - - - - - - - - - - } (8)$$

Since we have considered the best scenario for the multi-robot algorithm inspired by Bug1 which is the worst for Com1-Rev3 to derive equations (7 and 8), it follows that the total distance traveled by the two robots in Com1-Rev3 will always be less than that of multi-robot algorithm inspired by Bug1 by at least half the length of the obstacle. Hence Com1-Rev3 has a better energy expenditure.

$$\text{Average Distance in (a)} = \text{Total Distance in (a)} / 2 = SA+2(AB)+BC+LG/2 \text{ - - - - } (9)$$

$$\text{Average Distance in (b)} = \text{Total Distance in (b)} / 2 = SA+AB+BC+CG/2 \text{ - - - - - } (10)$$

$$\text{Average Distance in (a)} - \text{Average Distance in (b)} = AB+LG/2 - CG/2 \text{ - - - - - - } (11)$$

As LG tends to zero, CG/2 tends to AB/2 +LG/2. Hence, equation (11) reduces to:

$$\text{Average Distance in (a)} - \text{Average Distance in (b)} = AB/2 \text{ - - - - - - - - - - - } (12)$$

Equation (12) shows that the average distance traveled by each robot in a multi-robot algorithm inspired by Bug1 will always be at least one-quarter of the obstacle length greater than that of Com1-Rev3.

Fig. 5(c) is a circular illustration of the comparison. Starting point P and Goal Q could be moved right or left. It is most favorable for the multi-robot algorithm inspired by Bug1 when line PQ passes through the center of the circle (starting point S and goal G), the instance at which the meeting point is the closest point to the goal. Using the same symbols as before:

$$X_1 = SA+AD+DC+CG \text{ - - - - - - - - - - - - - } (13)$$

$$X_2 = SA+AB+BC \text{ - - - - - - - - - - - - - - - - } (14)$$

$$Y_1 = SA+AD+DG \text{ - - - - - - - - - - - - - - - - } (15)$$

$$Y_2 = SA+AB \text{ - } (16)$$

Note that

$$AD = AB = DC = BC$$

$$\text{Total Distance inspired by Bug1} = X_1+X_2 = SA+AD+DC+CG+SA+AB+BC$$

$$\text{Total Distance inspired by Bug1} = X_1+X_2 = 2(SA)+4(AD)+CG \text{ - - - - - - - - - } (17)$$

$$\text{Total Distance by Com1-Rev3} = Y_1+Y_2 = SA+AD+DG+SA+AB$$

Total Distance by Com1-Rev3 = Y_1+Y_2 = 2(SA)+2(AD)+DG - - - - - -- - - - - (18)

Total Distance inspired by Bug1 – Total Distance by Com1-Rev3
= 2(AD)+CG – DG - - - - - - - - - - - - (19)

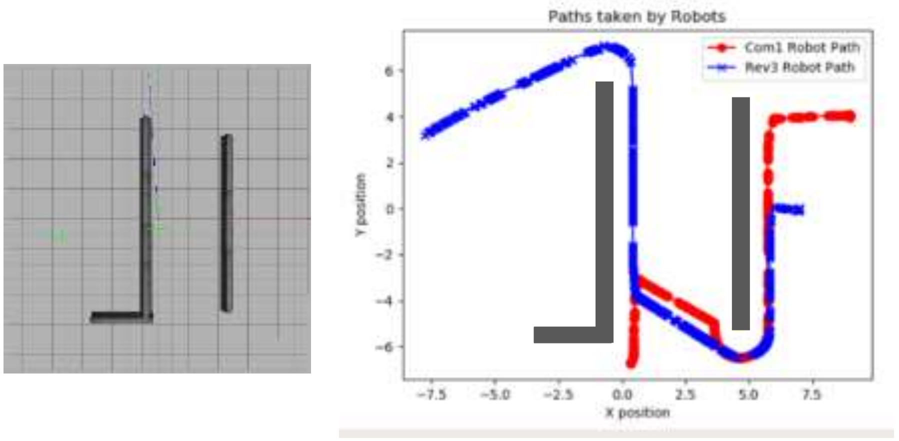
As CG tends to zero (that is G comes to C), DG tends to AD because AD = DC (best for the multi-robot inspired by Bug1). Hence, equation (19) reduces to:

Total Distance inspired by Bug1 – Total Distance by Com1-Rev3 = AD - - - - - (20)

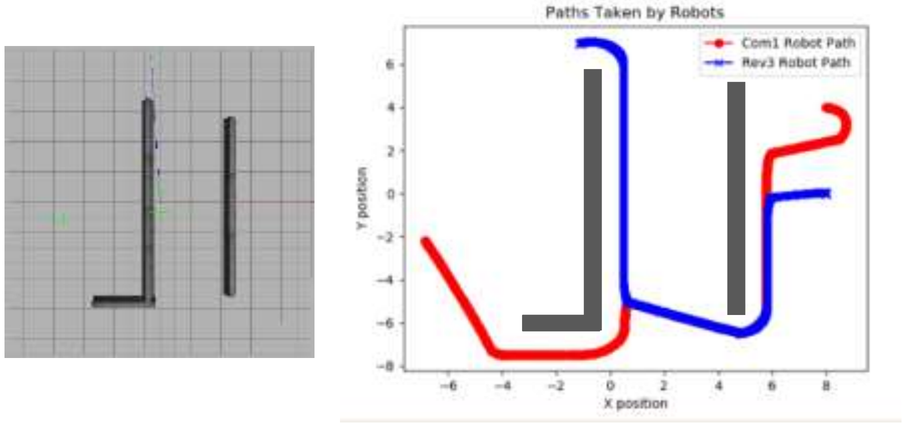
Equation (20) shows that for a circular obstacle, the total distance traveled by the multi-robot algorithm inspired by Bug1 is at least one-quarter of the obstacle circumference greater than the total distance traveled by Com1-Rev3. It also follows that the average distance traveled by each robot in the multi-robot algorithm inspired by Bug1 is at least one-eighth of the circumference greater than that of Com1-Rev3.

4.3 Com1-Rev3 Practical Demonstrations

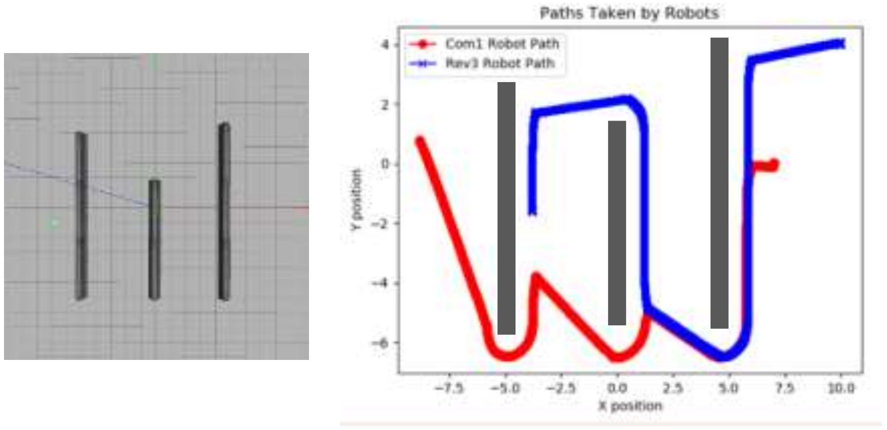
The algorithm was tested in different environments in the Gazebo simulation. The two robots cannot be at the same point at the start as they are not expected to be on top of one another. It is also a good practice to allow at least a 2m distance between them to take care of the collision control measure at take-off. For a specified starting point, the two could be 1m from the point at either side. Fig. 6 shows some pictures of paths taken by the robots in some of the scenarios.



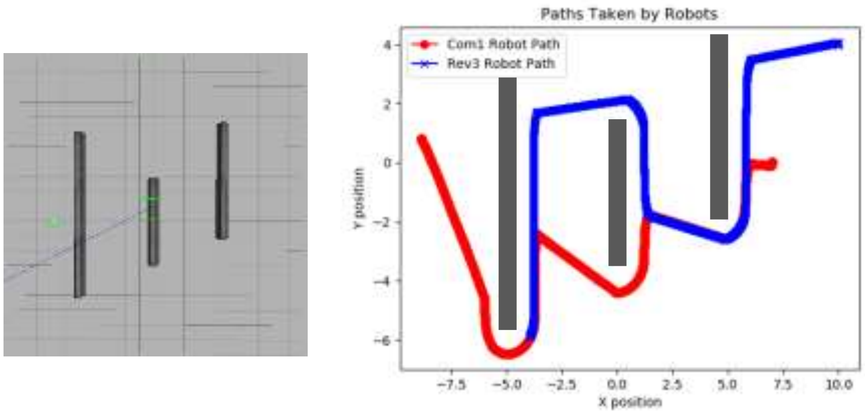
(a) Start: Com1 (9, 4), Rev3 (7, 0). Com1 stopped at (0.3, -6.7). Rev3 reached the target (-8, 3)



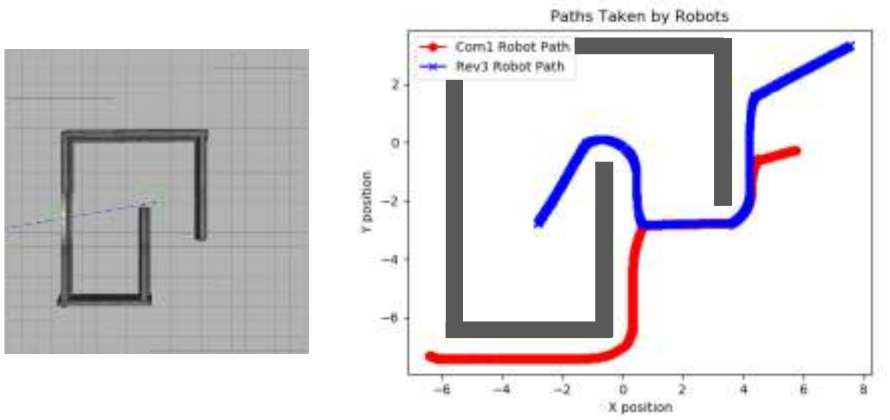
(b) Start: Com1 (8, 4), Rev3 (8, 0). Rev3 stopped at (-1.1, 7). Com1 reached the target (-7, -2)



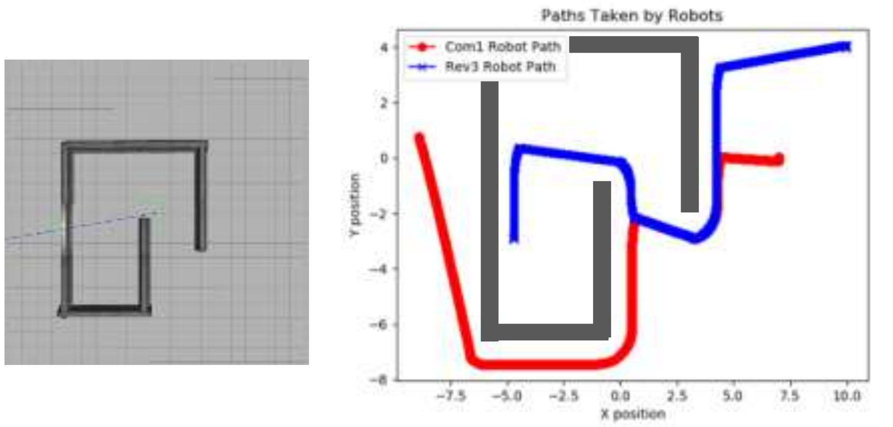
(c) Start: Com1 (7, 0), Rev3 (9, 4). Rev3 stopped at (3.8, -1.5). Com1 reached the target (-9, 1)



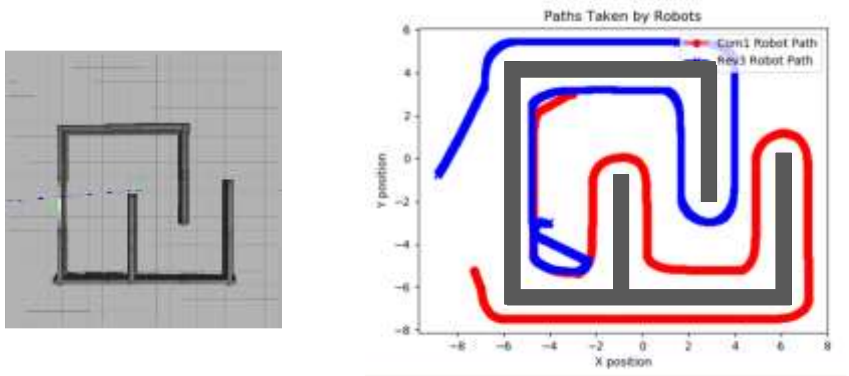
(d) Start: Com1 (7, 0), Rev3 (10, 4). Rev3 stopped at (-3.9, -5.9). Com1 reached the target: (-9, 1)



(e) Start: Com1 (6, 0), Rev3 (9, 4). Rev3 stopped at (-6.4, -7.3). Rev3 reached the target: (-3, -3)



(f) Start: Com1 (7, 0), Rev3 (10, 4). Rev3 stopped at (-4.7, -2.8). Com1 reached the target: (-9, 1)



(g) Start: Com1 (-3, 3), Rev3 (-4, -3). Com1 stopped at (-7.3, -5.3). Rev3 reached the target: (-9, -1)

Fig. 6. Robot paths in different simulations of Com1-Rev3

5. Conclusion

Two improved bug algorithms for autonomous robot navigation were developed in this paper. One is a single-robot algorithm (Rev3) which could be the preference where simplicity is the first consideration. The other is a dual-robot algorithm (Com1-Rev3) which would be the choice where efficiency is the priority.

The dual-bug algorithm is developed by implementing two different single-robot algorithms on two robots. To get this done, an existing algorithm, Rev2 was upgraded to Rev3. So, an existing algorithm, Com1, and the new one, Rev3 were used to develop the new dual-bug algorithm, Com1-Rev3. The advantage of Com1-Rev3 over the existing multi-robot algorithms include:

- Simplicity: The higher the number of robots, the bulkier the navigation system. Getting good performance even when the number of robots is just two keeps the system relatively simple.
- Effectiveness and efficiency: The effectiveness is high because it implements two highly effective algorithms that explore possibilities, but not irrationally like Bug1 (shown in Table 1). It is also proved above that Com1-Rev3 has better distance traveled and energy expenditure efficiencies.
- No waiting time: No robot waits for the other to accomplish a task. Waiting happens only when a collision is to be avoided. This makes Com1-Rev3 time efficient.
- Collision possibility alleviation: Robot collision possibility, one of the major challenges of multi-robot systems, is considerably alleviated by Com1-Rev3.
- No bug generation: Some multi-robot algorithms are characterized by bug generation as navigation progresses. This is a real live application challenge that is not present with Com1-Rev3.

By extension, Com1-Rev3 could be designed for any number of robots navigating together. It is recommended that the number be shared equally for the two algorithms, while half of Com1 are right-turning robots and the remaining half are left-turning robots. Then half of the Rev3 turns right at the first hit point while the remaining half turns left at the first hit point. These recommendations are subject to further studies for verification. Navigation space management is another established challenge to explore in multi-robot applications including Com1-Rev3.

References

1. Dhingra, V., Arora, A.: "Pervasive Computing: Paradigm for New Era Computing," *2008 First International Conference on Emerging Trends in Engineering and Technology*, Nagpur, India, 2008, pp. 349-354, doi: 10.1109/ICETET.2008.170.
2. Debashis, S.: Pervasive Computing: A Vision to Realize, Advances in Computers, Elsevier, Volume 64, 2005, Pages 195-245, ISSN 0065-2458, ISBN 9780120121649

3. Johansson, A., Markdahl, J.: "Swarm Bug Algorithms for Path Generation in Unknown Environments," *2023 62nd IEEE Conference on Decision and Control (CDC)*, Singapore, Singapore, 2023, pp. 6958-6965, doi: 10.1109/CDC49753.2023.10383598.
4. McGuire, K.N., de Croon, G.C.H.E., Tuyls, K.: "A comparative study of bug algorithms for robot navigation" 2019 Robotics and Autonomous Systems, Volume 121, 2019, 103261, ISSN 0921-8890,
5. Nandesh, S. S, D. A., R. Raman K, G. K, and R. R.: "An Investigation of Bug Algorithms for Mobile Robot Navigation and Obstacle Avoidance in Two-Dimensional Unknown Static Environments," *2021 International Conference on Communication Information and Computing Technology (ICCICT)*, Mumbai, India, 2021, pp. 1-6, doi: 10.1109/ICCICT50803.2021.9510118.

6 Kandathil, J. J., Mathew, R., Hiremath S. S.: Development and analysis of a novel obstacle avoidance strategy for a multi-robot system inspired by the Bug-1 algorithm. SIMULATION. 2020;96(10):807-824. doi:10.1177/0037549720930082

7 Bhanu, V., Chander, V., Asokan, T., Ravindran, B.: "A new Multi-Bug Path Planning algorithm for robot navigation in known environments," *2016 IEEE Region 10 Conference (TENCON)*, Singapore, 2016, pp. 3363-3367, doi: 10.1109/TENCON.2016.7848676.

8 Johansson, A., Markdahl, J.: "Swarm Bug Algorithms for Path Generation in Unknown Environments," *2023 62nd IEEE Conference on Decision and Control (CDC)*, Singapore, Singapore, 2023, pp. 6958-6965, doi: 10.1109/CDC49753.2023.10383598.

9. Lumelsky, V., Stepanov, A.: "Dynamic path planning for a mobile automaton with limited information on the environment," in *IEEE Transactions on Automatic Control*, vol. 31, no. 11, pp. 1058-1063, November 1986, doi: 10.1109/TAC.1986.1104175.
10. Horiuchi, Y., Noborio, H.: "Evaluation of path length made in sensor-based path-planning with the alternative following," *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, Seoul, Korea (South), 2001, pp. 1728-1735 vol.2, doi: 10.1109/ROBOT.2001.932860.