Real-time obstacle avoidance navigation strategy in unknown environments

Despina E. Athanasaki[1], Spyros Panagiotakis[1], Nikos Pinikas[1], Zacharias G. Sifakis[2], Andreas Vlisidis[1]

[1]Department of Informatics Engineering,
Technological Educational Institute of Crete, Heraklion, Greece,
[2]Department of Computers Science,
University of Hertfordshire, United Kingdom

Emails: despinadev@gmail.com; spanag@teicrete.gr; npinikas@hotmail.com; zacharias.sifakis@gmail.com; anvli@ie.teicrete.gr

***Abstract*- The goal of this research is the development of a detection and obstacle avoidance real-time algorithm for autonomous mobile robot (mobot) navigation in unknown environments. The above mobot will be implemented on low cost hardware, open source software and low processing power platform. The algorithm is designed on using the Pythagorean Theorem to describe the obstacle (determining the size, shape and the plurality of potential obstacles in the environment). A basic feature of the specific implementation is simplicity, not only in calculations but also in hardware requirements. Our algorithm requires only one ultrasound sensor for its operation, while provides very small blind zone and limited energy consumption. The particular algorithm could be applied in unmanned vehicles, for exploration of hazardous environments such as wreckages, volcanos, shipwrecks, radioactive areas, space exploration and generally in places that are hard to human existence.**

Keywords— Obstacle avoidance, Autonomous navigation, Mobile robot, Real-Time Algorithm, Unknown Environment

I. INTRODUCTION

Mobile Robotics is a field that involves many scientific disciplines such as electrical & electronics engineering, informatics engineering, mechanical engineering as well as social and human sciences. Each discipline individually is responsible for the proper development of robotics science.

Robots that are used most often, nowadays, are the autonomous mobile robots, which their essential feature, is their obstacle avoidance ability, in order to be regarded as autonomous.

Occasionally, the avoidance algorithms have preoccupied many researchers. Although the technology is constantly advancing and both software and hardware are improved, a major issue, for the implementation of an autonomous mobile robot are both material and energy cost [1][2]. So, it is important to create an algorithm, which will be used in a platform which needs as less processing power as possible. This will be achieved by using the fewest possible modules and sensors, in order to have both power efficiency and even results.

The specific algorithm we propose, is based on simple mathematical equations, in order to be applied in non-great requirements hardware. The main concept of the algorithm is based on the obstacle detection, degrees calculation and distance calculation, in order to bypass the obstacle. Required the starting position and direction of the vehicle, so that it can return to its original orientation. The vehicle model has four drive wheels, which angular velocities (in pairs) are controlled independently. The rest of the paper is organized as follows: section II and III describe the proposed methodology, section IV discusses some similar works, section V shows the results of the implementation and section VI concludes the paper.

II. THE PROPOSED MATERIALS FOR IMPLEMENTATION

This work is developed in a prototyping platform such as the Arduino Uno but certainly can be used by other platforms. Two DC motors are used for the wheel motion, an H-Bridge circuit, which can drive the motors in forward and reverse, and two batteries, which feed the motors and the board. Assuming that 0 degrees is straight, a sonar (ultrasound sensor) scans 180 degrees to the left and 180 degrees to the right. The blind zone depends on the measuring angle of the sonar. We propose a sonar with measuring angle more than 45^o in order to achieve a small blind zone. The vehicle shifting is calculated using either two IR-distance sensors or an ultrasound wheel encoder.

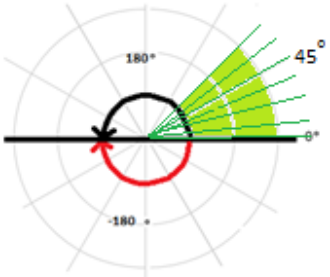

Figure 1: Sonar Range

III. THE PROPOSED METHOD

The quickest way for any predetermined target even in simple roaming conditions, is a straight line. The vehicle will move in straight, meaning that the angular velocities per pair of wheels (left and right) are equal. If it encounters an obstacle, it will

circumnavigate it, until overtake it and come back to its original orientation. In this instance, the vehicle changes direction (right or left) simply by changing the angular velocity of the respective wheels. To move left, the rightist wheels velocity should be higher than that leftist and respectively, for dextrorotatory motion, the velocity of the left wheels should be higher. For spot left turn, the velocity of the rightist wheels should be positive (forward), while leftist should be negative (backward) and conversely, for the spot right turn.

Obviously we cannot set default values for the velocity and the angles of the wheels, since these parameters depend on the implementation materials, the sensors, the supply voltage, the total weight of the vehicle, the data processing speed of the platform and several other factors. This means that in different vehicles, these parameters will have different values.

It should be mentioned that, the vehicle has returned to its original orientation only when the sum of the angles, that its path makes is equal to 0 (Figure 2).

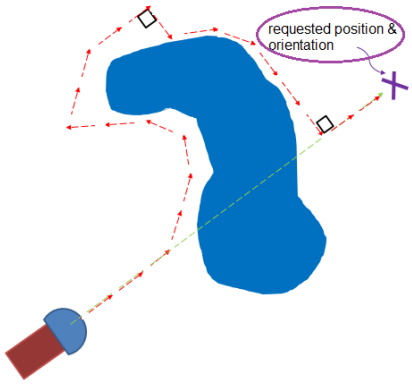

Figure 2: Estimated path

Below is an analysis of the most important parts of the algorithm. This project can be implemented in any compatible programming language.

1st Part

Originally the mobot is traveling in a predetermined direction. It is the first approach to the obstacle and it must decide if it will overtake it from the right or from the left.

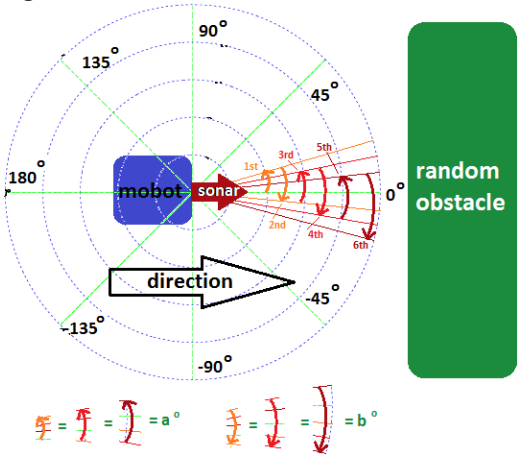

Figure 3: First approach

It should be noted that, initially, the distance that it has to be moved, and the degrees that it needs to turn, are completely unknown to the mobot, therefore, theoretically, the distance is infinite, and the angle is180^o, (opposite movement direction).

Values initialization of variables that are used: **deg=0** *(keeps the degrees that the mobot has digress from its initial route).* **obst=0** *(it is a Boolean variable and indicates whether there is obstacle (obst=1) or not (obst=0)).* **a** *(keeps the selected degrees that sonar will check to the left (positive)) and* **b** *(keeps the selected degrees that sonar will check to the right (negative)).* The initial values of both **a** and **b** angles must be odd numbers, so that the angle calculation algorithm not to *get into an infinite loop. The even difference value of the angles can return the vehicle at 0 degrees. That is because of* sum and difference of two odd numbers *is always even. The proof of this statement is:*

Suppose n and m are odd integers.
- Then $n = 2k + 1$ and $m = 2l + 1$ for some k, l ∈ Z, (by the definition of an odd integer).
- Therefore $n + m = (2k + 1) + (2l + 1) = 2(k + l + 1)$.
- Since k and l are integers, so is $k + l + 1$.
- Hence $n + m = 2p$ with $p = k + l + 1 \in Z$.
- By the definition of an even integer, this shows that n+m is even. [3].

In any other combination of angles, it will never be at 0 degrees, thus their differential will be odd number (odd-even), or it will get into an infinite loop (even-even), since their difference after some checks, may be one of them.

d parameter indicates predetermined distance until the next test of sonar. It has standard value that does not change during the duration of the algorithm. **cnt1=0** (a counter which will hold the number of the checks that sonar has done, in order to calculate the degrees that the mobot will start moving to avoid the obstacle). It should be mentioned that, the mobot must keep moving to the side which was its initial selection, that is, if it chooses to overtake the obstacle from the right, it should continue the right path and vice versa, otherwise, there is a great chance to get in an endless loop. **sonar (deg)** is the function which is used by the ultrasound sensor. Its return value is 1 or 0 (if there is an obstacle or not in specific degrees **deg**). Commonly its return value is kept by the **obst** variable, **obst=sonar (deg**). The functions are also used are, **stop (),** which stops the robot, **move (d),** which is used to move the robot as long as the parameter d indicates and **turn ()** function, which turns the mobot (not the sonar) so many degrees as its parameter keeps.

```
if (obst=1)
{
  stop ();
  cnt1=cnt1+1;
  if (cnt1 mod 2=0)
  {
```

```
        sonardeg= (cnt1/2)*(a+b);
        obst=sonar (sonardeg);
 }
 else if (cnt1 mod 2!=0)
 {
        sonardeg= ((cnt1+1)/2)*a+ (((cnt1+1)/2)-1)*b;
        obst=sonar (sonardeg);
 }
}
else (obst=0)
{
        deg=deg+sonardeg;
        turn (deg);
        move (d);
        obst=sonar (sonardeg);
}
```

This part of the algorithm is running just once at the beginning of the project. Until the vehicle decides from which side it will begin overtaking the obstacle.

2nd Part

The next part of the algorithm checks, the way that the robot will be moved. It has already spot the obstacle and has decided by which side will overtake it. This procedure is accomplished by checking each time the existence of an obstacle in the direction of its movement. As there is no obstacle, the function **move (d)** is called.

Additional variables used, ***h=0*** which keeps the height that the mobot travels in each check, ***dist=0*** which keeps the total distance that the mobot has traveled and ***Total_h=0*** which keeps the total height that the mobot has moved. All three variables' initial values are 0.

```
dist=0;
Total_h=0;
deg=deg+sonardeg;
turn (deg);
obstacle=sonar(0);
while (obst=0)
{
        h=0;
        move (d);
        dist=dist+d;
        if (deg>90)
                h=sin(180-deg)*dist;
        else
                h=sin(deg)*dist;
        obst=sonar(0);
}
Total_h=Total_h+h;
```

When the parameter of the **sonar ()** function is 0, then the ultrasound sensor is aligned to the vehicle.

Every time the **move (d)** function is called, the shift increases by **d**.

Figure 4 depicts the second part of the algorithm. Each red arrow shows the orientation and indicates the distance **d** that the mobot travels, each time the function **move (d)** is called. The green arrow indicates the direction that mobot had in previous control, so that the phase difference can be calculated. The brown arrow indicates the check that is carried out every **φ** degrees, until the appropriate path will found. **φ** is the result of the 1^{st} part of the algorithm (**φ** =sonardeg). **θ** is the angle that is used to calculate the drift if the sum of all **φ** angles exceeds 90 degrees. **h** is the amount that it is shifted at every change of direction and **Total_h** is the overall shift (height).

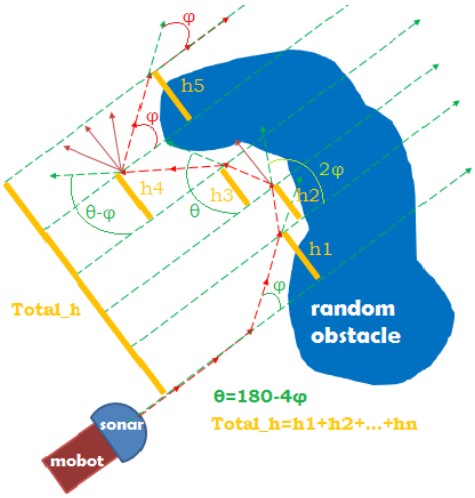

Figure 4: Path computation

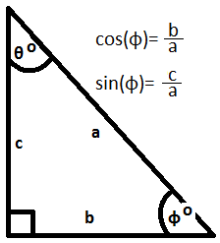

Figure 5: Trigonometric equations

The height is calculated by the formula of the Pythagorean Theorem. In this way, the overall height that the vehicle has gone up, is held in the variable (***Total_h)***, in order to descend at the same value.

3rd Part

It should be noted that, the sensor will check so many degrees, as they calculated in the first part of the algorithm and have been kept in the variable **sonardeg**. Variable **Total_deg** keeps the total number of degrees the sonar has turned. There is, also, a second counter, which keeps the number of times the sonar scanned. Its initial value is equal to 0.

```
if (obst=1)
{
        obst=sonar(sonerdeg);
        Total_deg=Totaldeg+sonardeg;
        cnt2=cnt2+1;
```

```
}
else if(obst=0)
{
        turn(sonardeg*cnt2);
   obst=sonar(0);
 }
```

In a few words, this part of the algorithm allows the robot to check if there is free space to access by turning the sonar, using **sonar ()** function. The verification is done every **sonardeg** degrees. Once the robot find passage **(obst=0)**, the algorithm calls the **turn ()** function, which turns the entire vehicle, (**sonardeg* cnt2)** degrees. The sonar returns to 0 degrees that is straight relative to the robot. Once this part of the algorithm is finished, the second part is re-executed. *(figure 6, figure 7)*

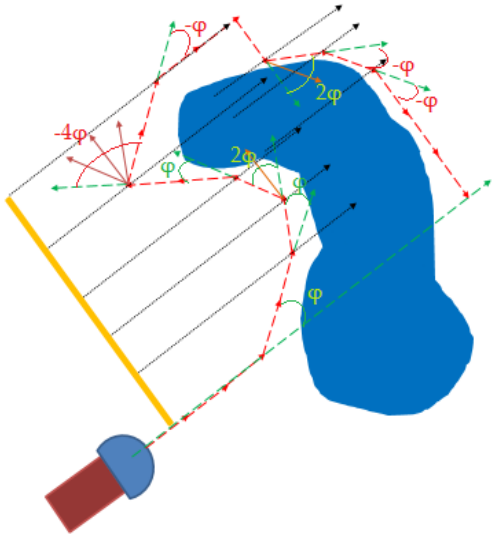

Figure 6: Angles Computation

4th Part

At regular intervals, this part of the algorithm, which checks whether the robot can turn in its original direction, scilicet at 0^o, will be called.

Since the total degrees that the robot has turned are kept in **Total_deg** variable, it is feasible, the robot gets same degrees back. *(figure 8)*

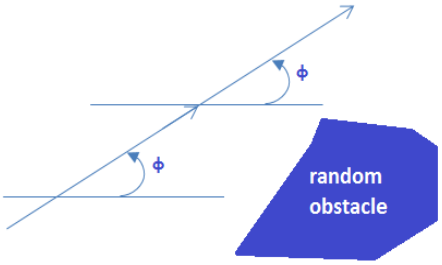

Figure 7: Angle & distance check

The following part, as well as the previous parts of the algorithm, can be implemented as a recursive function. The function's name will be **Turnback ().** Two new variables will be used, **L_deg** with initial value equal to **Total_deg,** which keeps the sets of degrees that are remaining, until the robot returns to its original orientation and **obst2,** which keeps the return value of the **sonar ()** function, in any contingent verification. If the value of the variable **obst2** is equal to 0, then the value of the variable **obst will be** equal to 0 too, thus, the function **Turnback ()** will be recalled by itself, otherwise, the 2^{nd} Part of the algorithm will be re-executed. This will occur, until there are no remaining degrees, up to the initial orientation.

```
Turnback ()
{
        L_deg=Total_deg;
        do
        {
                L_deg=L_de-sonardeg;
                obst2=sonar(-sonardeg);
                if (obst2=0)
                {
                        obst=obst2;
                        turn(-sonardeg);
                Turnback ();

                }
                else if (obst2=1)
                {
                        // Call 2nd Part…….
                }
                while (L_deg>0)
        }
}
```

5th Part

This part of the algorithm is used after the robot has been oriented (L_deg=0) and its distance from its original position is equal to **Total_h.** So it should come back as much as it was shifted, after turning 90^o.

A new variable (**L_dist**) is used which keeps the distance that is remaining to the final position. Its initial value is equal to **dist.** *(figure 8)*

```
L_dist=dist;
if (L_deg=0)
{
        obst= sonar(90);
        if (obst=0)
        {
                turn(90);
                mode (d);
                L_dist=L_dist-d;
        }
        turn (-90);
        move(d);
        obst=sonar(0);
}
```

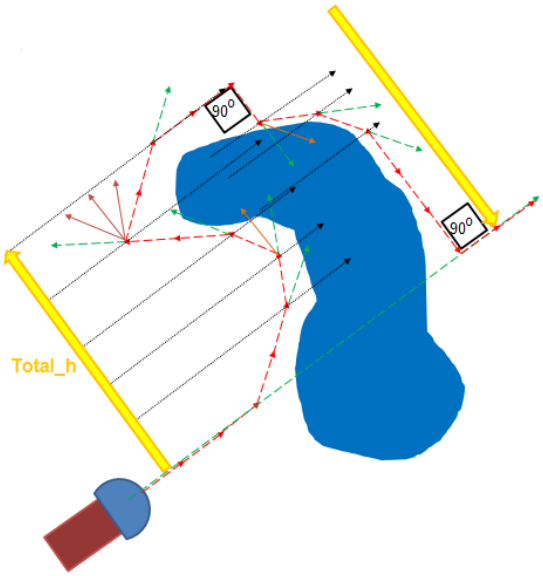

Figure 8: Return to initial direction

IV. RELATED WORKS

Various types of autonomous navigation robots and obstacle avoidance algorithms already exist.

A similar implementation, which uses five ultrasonic range finder sensors with a very small blind zone existence in the sensor arrangement, is described in [4]. The navigation of the mobile robot depends on the position of the mobile robot in the influence zone area. The formation of three layers of influence zones shows the optimized path planning without making any unnecessary obstacle avoidance presence. It is an interesting solution to the problem of moving a mobile robot through a two-dimensional workspace containing obstacles moving on unknown trajectories. The workspace and obstacles are convex polygons, and the robot is a point [5].

Also, an implementation for nonholonomic vehicles with acceleration constraints and limited sensing, is interesting. A bounded control law for nonholonomic systems of unicycle-type that satisfactorily drive a vehicle along a desired trajectory, while guaranteeing a minimum safe distance from another vehicle or obstacle, is proposed in [6]. The control law is comprised of two parts. The first is a trajectory tracking and set-point stabilization control law that accounts for the vehicle's kinematic and dynamic constraints (i.e. restrictions on velocity and acceleration) [6]. Also, a very common and affordable solution is the use of laser sensor instead of ultrasound sensor. This strategy takes a laser rangefinder as the main environment detecting sensor. A polar co-ordinate vector map (PMap) is utilized to store environmental information and a cluster ideology is used to describe obstacle features. The local virtual target comprise environment features and target information and it is used as a temporary target to guide the robot to track the real goal while accomplishing obstacle avoidance [7][8].

One of the best known algorithms for obstacle avoidance navigation is the BUG algorithm. It is a hybrid algorithm for the autonomous navigation of an Unmanned Ground Vehicle (UGV) using visual topological maps. [9] The Bug1 and Bug2 algorithms are among the earliest and simplest sensor-based planners with provable guarantees. These algorithms assume the robot is a point operating in the plane with a contact sensor or a zero range sensor to detect obstacles. When the robot has a finite range (non-zero range) sensor, then the Tangent Bug algorithm is a Bug derivative that can use that sensor information to find shorter paths to the goal [10]. A variation and evolution of the Bug algorithm is the IBA (Intelligent Bug Algorithm). The improved algorithm offers a goal oriented strategy by following smooth and short trajectory. This has been achieved by continuously considering the goal position during obstacle avoidance [11].

Algorithm	Sensors	Motors	Blind zone
[4]	5 sonars	4	Very small
[5]	2 sonars & 1 IR	4	small
[6]	1 sonar	2	small
[7][8]	1 laser & 1IR or 1 sonar	1	large
[9]	2 sonars or 1 sonar & 1 IR	2 or 4	small
[10][11]	2 sonars or 1 sonar & 1 IR	2 or 4	small
Our algorithm	1 sonar	2 or 4	small

Table 01: Hardware comparison between similar algorithms

As noted, most of the aforementioned algorithms use more complex hardware such as several sensor types or use a large number of ultrasound sensors, which makes them more energy-consuming than our algorithm.

V. IMPLEMENTATION & RESULTS

Below, the results of the implementation of the algorithm are available. The hardware consists of the materials that are referred in section II. Also, a L298N is used. It is a two-H Bridges chip, which it can move two DC motors, immediately. The HC-SR04 ultrasound sensor, which its measuring angle is almost 60^o in order to achieve small blind zone, is used in combination with a Torque TowerPro SG90 servo motor. Also they are used four wheels, which are bind to L298N bridge, and an ultrasound wheel encoder. It is the same sensor that is used for obstacle detection (HC-SR04), which uses a distance calculation function.

Concerning the software parameters, the initials values of the **a, b** angles are11^o, -35^o respectively. The value of **d** distance is 50, and lastly the speed of left wheels sets to 120, while the speed of right wheels sets to 110 (hardware malfunction). As already mentioned, the values of these parameters depend on several factors, such us the hardware and the environment.

There were approximately 70 tests to have legitimate results. Figures 9 to 12 illustrate the operation of our mobot.

Figure 9 :1st part of the algorithm-obstacle detection

Figure 10: 2nd part of the algorithm-path selection

Figure 11: 3rd part of the algorithm-return to original orientation

Figure 12: 4th and 5th part of the algorithm-return to original direction.

The vehicle does not stop in the final element, but continues its path, until to re-find an obstacle and do in the same way.

VI. CONCLUSION

This paper presented the development of an obstacle avoidance algorithm for the autonomous navigation of mobile robots in unknown environments. Our future plans include the evolution of this work to an involved algorithm which takes place in three dimensions, to be used in flying robots (drones) for easier space exploration. Moreover, it could be placed in service of the underwater archeology, a not very known but hazardous field of study, in which the robot will assist archaeologists during possibly dangerous and expensive shipwreck exploration missions. In such a case, the saving of processing power is much more important.

VII. REFERENCES

[1] Carlos M. Costa, Héber M. Sobreira, Armando J. Sousa and Germano Veiga, "3 DoF/6 DoF Localization System for Low Computing Power Mobile Robot Platforms", in book: Cutting Edge Research in Technologies, Edition: 2015.

[2] Kuo-Lan Su1, *, Yi-Lin Liao2, Shih-Ping Lin3 and Sian-Fu Lin3, "An Interactive Auto-recharging System for Mobile Robots", in International Journal of Automation and Smart Technology 4(1):43-53 · March 2014.

[3] A.J. Hildebrand, "Fundamental Mathematics Math 347", Department of Mathematics University of Illinois, 2015. Available from: http://www.math.uiuc.edu/

[4] R.N. Farah, 2N. Irwan, 3R.L. Zuraida, 4S. Amira and 5M.H. Hafiz, "Path Planning for Mobile Robot Based On Reactive Collision Avoidance Method", in Australian Journal of Basic and Applied Sciences. July 2014.

[5] James Gil de Lamadrid,"Avoidance of Obstacles with Unknown Trajectories: Locally Optimal Paths and Periodic Sensor Readings" The International Journal of Robotics Research December 1994.

[6] Erick J. Rodríguez-Seda, Chinpei Tang, Mark W. Spong, Dušan M. Stipanović, "Trajectory tracking with collision avoidance for nonholonomic vehicles with acceleration constraints and limited sensing". The International Journal of Robotics Research October 2014. Available from: http://ijr.sagepub.com/content/33/12/1569.abstract

[7] L.-M. Ren, Weidong Wang, Z.-J. Du," On a Laser Rangefinder-based Mobile Robot Obstacle Avoidance Strategy via Local Virtual Target", article in Lasers in 25(3):205-223 · January 2013.

[8] Cezary Kownacki, "A concept of laser scanner designed to realize 3D obstacle avoidance for a fixed-wing UAV", article in Robotica 34(02):243-257 · FEBRUARY 2016.

[9] Juan Pablo Fuentes Brea, Darío Maravall, Javier de Lope," Visual Bug Algorithm for Simultaneous Robot Homing and Obstacle Avoidance Using Visual Topological Maps in an Unmanned Ground Vehicle", Conference: IWINAC 2015 Bioinspired Computation in Artificial Systems Lecture Notes in Computer Science, At Elche, June 2015.

[10] Howie Choset, Kevin Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia Kavraki, and Sebastian Thrun," Principles of Robot Motion: Theory, Algorithms, and Implementation", ERRATA, August 23, 2010.

[11] Muhammad Zohaib, Syed Mustafa Pasha, Nadeem Javaid3, Jamshed Iqbal, "An Intelligent Bug Algorithm (IBA) to Navigate an Autonomous Mobile Robot", Conference: Springer's International Multi Topic Conference 2013 (IMTIC '13).