Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

VFH+D: An Improvement on the VFH+ Algorithm for Dynamic Obstacle Avoidance and Local Planning ⋆

Daniel Díaz * **Leonardo Marín** *

* *Electrical Engineering School, University of Costa Rica, Ciudad de la Investigación, 11501 San Pedro, San José, Costa Rica. (e-mail: {daniel.diazmolina, leonardo.marin}@ucr.ac.cr).*

Abstract: This paper highlights some limitations of the VFH+ algorithm on the domain of local obstacle avoidance. An enhanced algorithm dubbed VFH+D is proposed, which considers a different way of calculating the obstacle vector magnitude and a decay algorithm for dynamic obstacle avoidance. Experiments were conducted to compare both algorithms on two different mecanum wheeled robots, VFH+D achieved higher average speeds and lower distance traveled to reach the goal.

Keywords: mobile robots, obstacle avoidance, navigation, path planning, autonomous robotic systems.

1. INTRODUCTION

Obstacle avoidance algorithms are an essential aspect in mobile robot navigation, as they allow the robot movement in the environment while preventing collisions when following a path or achieving a goal. These algorithms are used by the path planning methods, along with the robot position and the movement area map, to calculate and follow the best route in order to reach the target while avoiding obstacles. This behavior is essential in multiple navigation applications such as car collision avoidance for automated road safety (Adla et al., 2013; Dahl et al., 2019), autonomous vehicles (Lefebvre et al., 2004; Houjie Jiang et al., 2016; Wang et al., 2019), unmanned marine vessels (Chen and Li, 2017), mobile robots, both individual (Miró et al., 2008; Alajlan et al., 2015; Almasri et al., 2016) and for cooperative groups (Soriano et al., 2015; Alonso-Mora et al., 2018).

In order to safely navigate the environment, path planning algorithms usually divide the problem in two main components: A Global navigation planner that uses optimization techniques to obtain the best route to the goal in the available map of the environment (high level planning) (Marder-Eppstein et al., 2010; Pandey et al., 2017; Patle et al., 2019), and the Local navigation planner, that generates velocity commands to follow the global plan as closely as possible, while considering the kinematic constraints and using the sensor data to dynamically detect changes in the environment and obstacles in order to avoid them (Miró et al., 2008; Patle et al., 2019). Among the first and most simple obstacle avoidance methods used as local planners are the bug algorithm and its variants (Lumelsky and Stepanov, 1987; Lumelsky and Skewis, 1990; Ng and Bräunl, 2007; Marín et al., 2010) that avoid static obstacles by following their perimeter until the path is clear. There are also the pure reactive Braitenberg inspired algorithms (Braitenberg, 1986) that will turn in the presence of static obstacles, but some versions (Shayestegan and Marhaban, 2012) also implement additional rules to retake the path to the objective.

More advanced local planners have been developed that can consider the kinematic, dynamic and other constraints of the mobile platforms while being fast and robust in the obstacle evasion (Berns and von Puttkamer, 2009), such as the Dynamic Window Approach (DWA) (Fox et al., 1997), the Timed Elastic Band (TEB) (Roesmann et al., 2012), the Follow the Gap Method (FGM) (Sezer and Gokasan, 2012) and the Vector Field Histogram (VFH) (Borenstein and Koren, 1991) algorithms. All of these methods define, in real time, the required robot velocities, using different strategies to represent the environment from the sensor data, in order to extract from it the obstacle information and use it to determine the best robot movement to avoid the obstacles and transit the available free space (Shim and Kim, 2018; Cybulski et al., 2019).

The original VFH (Borenstein and Koren, 1991) searches for openings among obstacles that are big enough for the robot to travel and avoid collisions, while building a local map of the encountered obstacles using rangefinder sensors to estimate the certainty of their position in a grid. An improvement of this algorithm, VFH+, was proposed in (Ulrich and Borenstein, 1998) that takes into consideration the robot size and improves the steer angle selection using a polar histogram, that masks the sectors behind the detected obstacles so they are not assigned as open space. Another improvement, VFH*, was proposed in (Ulrich and Borenstein, 2000) that defines a look ahead direction search with the A* pathfinding algorithm, improving the obstacle detection in some cases, but also being closer to a global path planner without guaranteeing the optimality of the solution.

⋆ The financial support from the University of Costa Rica, under the grant 322-B8-298, is greatly appreciated.

As the VFH+ is fast and robust in avoiding obstacles, it is commonly used as a local planner (Yim and Park, 2014; Sary et al., 2018) in combination with other global planning methods to provide a reliable solution to the navigation problem (Pandey et al., 2017). The main disadvantage of the VFH+ algorithm is its inability to perform well in environments with dynamic obstacles (Miró et al., 2008). In order to overcome this limitation, the algorithm is usually combined with other techniques, such as the Kalman filter (Guo et al., 2010; Kim et al., 2010), in order to predict obstacle movement, the aforementioned showing promising results but demonstrated only by simulations.

In this work we analyze some of the limitations of the VFH+ algorithm and propose a new and improved version called Vector Field Histogram + Dynamic (VFH+D) that overcomes these deficiencies. The proposed algorithm has the capacity of avoiding dynamic obstacles, while also improving the robot speed and success rate in reaching the goal point while avoiding collisions, as shown by the experiments conducted on an omnidirectional robot that implements the proposed algorithm in the Robot Operating System (ROS).

2. VFH+ ALGORITHM AND LIMITATIONS

The VFH+ algorithm as defined on Ulrich and Borenstein (1998) uses five data structures, the data is processed sequentially from one data structure to the next: two 2-dimensional grids and three 1-dimensional polar histograms. The following is a summary of the data flow.

- The map grid or obstacle grid C: a 2-dimensional grid that represents the obstacles in the world reference frame, each cell holds a certainty value between 0 and c_{max}. A cell's certainty value is increased by 1 for each sensor reading that detects an obstacle in that cell.
- The active window C_a: a much smaller 2-dimensional grid that follows the robot. Each cell holds an "obstacle vector" that consists of a magnitude $m_{i,j}$ and direction $\beta_{i,j}$, where $m_{i,j}$ is a function of the cells distance to the robot's center $d_{i,j}$ and corresponding certainty value $c_{i,j}$ as described by (1), and $\beta_{i,j}$ is the direction from the robot's center to the cell.
- The primary polar histogram H^p: a 1-dimensional histogram of the angular sectors of width α around the robot. Each sector holds a *polar obstacle density* which is the sum of the magnitude of all the cells in C_a that fall within that sector. An enlargement angle for cells is also defined based on the robot's radius r_r and a parameter for minimum obstacle distance r_s, so a single cell can add to more than one sector.
- The binary polar histogram H^b: a 1-dimensional histogram that maps each sector on H^p to 0 (free) or 1 (blocked) based on its value H^p_k. Two thresholds τ_{low} and τ_{high} are defined, if $H^p_k < \tau_{low}$ then $H^b_k = 0$, if $H^p_k > \tau_{high}$ then $H^b_k = 1$, otherwise H^b_k remains unchanged from its previous value.
- The masked polar histogram H^m: additional sectors in H^b are blocked based on the robot's direction of movement and minimum steering radius.
- Consecutive free sectors in H^m are classified as wide or narrow valleys according to their size, and *candidate directions* for each valley are then added to a list.

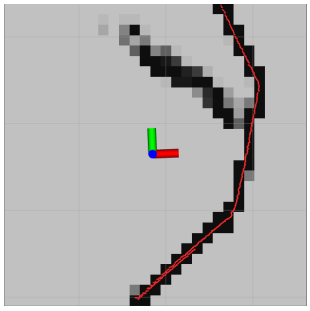

Fig. 1. Trail left by moving obstacle near a wall on the occupancy grid, darker cells have higher occupancy values

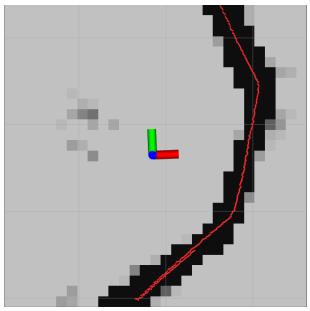

Fig. 2. Several grid cells with maximum occupancy value near a wall, red dots are the actual laser ranger readings at a given time.

The steering direction is determined by applying a cost function to all candidate directions and selecting the one with the least cost.

$$m_{i,j} = c^2_{i,j}(a - b \cdot d^2_{i,j}) \tag{1}$$

For equation (1), Borenstein suggests choosing the parameters a and b such that $m_{i,j}$ in border cells is equal to $c^2_{i,j}$.

While working with the VFH+ algorithm in practice several limitations became evident. The first was the inability of the algorithm to handle moving obstacles. Grid cells in the obstacle grid are updated to increase their certainty values when objects are detected, however, no mechanism for decreasing these same cells values is described. As such, in situations where there are moving obstacles the algorithm could deem large parts of the grid to be occupied, as shown in Fig. 1. The cells are actually free, but the "trails" of moving obstacles were left in the occupancy grid.

Another related problem happens when the robot's odometry has poor accuracy or is badly configured, or when the robot's wheels slip. Differences between the approximated and actual position can cause the occupied cells to "blur" into adjacent cells as demonstrated in Fig. 2.

In addition to the former, additional tests also brought other undesirable behaviour to light. Many of our tests were done in a relatively small (7×5 m) enclosed arena. We noticed that the algorithm tended to perform poorly close to the walls, especially when the direction of movement was perpendicular to the wall. Also, the robot often avoided walls that were far away as much as obstacles that were right next to it, adjusting parameters didn't seem to

correct this. After analyzing the effects of the obstacle grid on the polar histogram for different parameter configurations the team came up with some hypotheses to explain this behaviour.

First, the sensors used for obstacle detection should be considered, in our case, scanning laser rangefinders in the front and back of the robot. In general, cells tended to become saturated in regards to their occupancy value very quickly (in less than a second), such that all cells had either the maximum occupancy value or zero. In turn, **the magnitude of cells in the active window was usually the maximum value possible given the cells distance as described by equation (1)**. This is due to the high sampling rate and resolution of the laser rangefinders sensors used.

Since the *polar obstacle density* of each circular sector is calculated as the sum of the magnitudes of all cells inside that sector, and cells are arranged in a simple square grid, there are a lot more cells that contribute to the value of a sector further away from its origin. Because of this and the way H^p is calculated with (1), larger obstacles that are far away and closer to the border of the active window tend to drive the *polar obstacle density* of a single sector to values as high or higher than smaller obstacles much closer to the robot, this is considered as an undesired behaviour.

The former situation is exemplified in Figs. 3, 4 and 5, which correspond to a configuration with a wall to the left and an obstacle to the right of the robot (the white cells are occupied by the robot, the black cell is the robot's center). Notice in Fig. 4, the magnitude of obstacle vectors in the active window is lower on cells farther from the robot, as expected from equation (1). However, the polar histogram has a higher obstacle density in the direction of the wall, as show in Fig. 5.

3. PROPOSED ALGORITHM

For the purpose of this paper, the aforementioned problems can be summarized as follows:

(1) **The algorithm is not robust to changes or uncertainties in the obstacle grid, and cannot deal with moving obstacles.**
(2) **The algorithm performs poorly in the presence of walls, and in general obstacle proximity does not translate to a higher obstacle density in H^p.**

3.1 Obstacle decay rate

The first proposed change to the VFH+ algorithm is to add a "decay" to cells around the active window, such that occupancy values of all the cells inside the window plus an additional number of surrounding "guard band" cells will decrease over time. We define the following parameters:

- **Decay rate** (R_d): the frequency with which occupancy values are decreased.
- **Decay value** (d): the amount to subtract from each cell.
- **Decay guard band** (gb): additional cells to decay around the active window.

The main control loop will then periodically call algorithm 1 at the defined rate.

Algorithm 1 Decay algorithm

Require: d: decay value, gb: decay guard band.
Ensure: C: obstacle grid, C_s: obstacle grid size, w_s: active window size, w_c: $\lfloor w_s/2 \rfloor$, i_0: robot's X position in C, j_0: robot's Y position in C.

```
1: procedure DecayActiveWindow(d, gb)
2:     i_s ← max(i_0 − (w_c + gb), 0)
3:     j_s ← max(j_0 − (w_c + gb), 0)
4:     i_f ← min(i_s + w_s + 2 · gb), C_s)
5:     j_f ← min(j_s + w_s + 2 · gb), C_s)
6:     for i ← i_s, i_f do
7:         for j ← j_i, j_f do
8:             if C[i][j] ≥ d then
9:                 C[i][j] ← C[i][j] − d
10:            else
11:                C[i][j] ← 0
12:            end if
13:        end for
14:    end for
15: end procedure
```

3.2 Magnitude equation

The second proposed change to the algorithm is the use of a different equation to determine obstacle magnitude, in order to improve on the poor performance of equation (1) in cluttered environments.

While considering alternatives, we minded the following important points:

- Distance should have a higher impact on vector magnitude, a linear or quadratic proportion is not enough.
- Cells with the maximum occupancy value in close vicinity to the robot should have a blocking effect on the polar histogram, even if it is just one or two cells.
- The equation's different parameters should affect independent aspects of the observed behaviour of the robot.

After considering the former aspects, we came up with equation (2). Let's assume $c_{i,j}^2 = 1$ in order to focus on the exponential term, which is what really differentiates this new function from equation (1). Figure 6 plots $m(d)$ for different E and B values, these will be the main parameters to adjust the shape of the function.

$$m_{i,j} = c_{i,j}^2 \cdot e^{-\frac{1}{B} \cdot \left(\frac{d_{i,j}}{D}\right)^E} \tag{2}$$

Notice how higher E values produce a more pronounced slope. In practice, the E value should be increased if the algorithm is not making enough distinction between close and distant obstacles. On the other hand, higher B values "spread" the slope along the x-axis. In practice, this would be done to increase the minimum distance to obstacles that determines a particular direction to be blocked.

Finally, parameter D is useful to "normalize" or change the function's scale on the x-axis and allow portability across different robots or use cases, for example by setting D to the robot's radius r_R. Other values that could be used are the width of the active window, the LIDAR's maximum range or some other significant reference value. In fact,

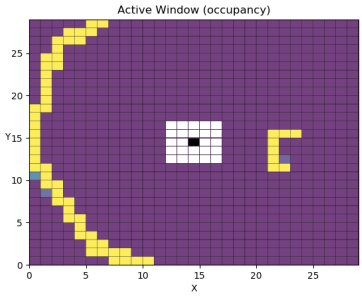

Fig. 3. Active window occupancy

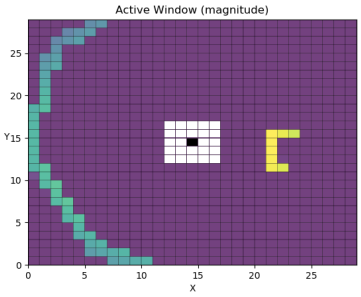

Fig. 4. Active window magnitude

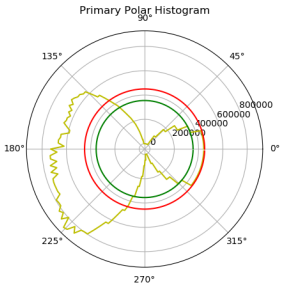

Fig. 5. Polar histogram

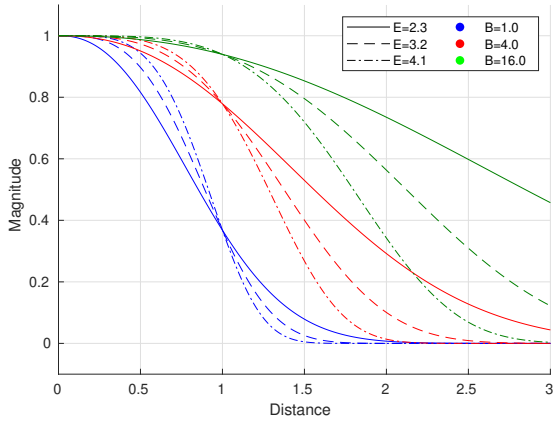

Fig. 6. Effect of parameters B and E on equation (2).

parameter D is actually redundant, but we found it to be very useful to simplify parameter tuning in practice.

Also, notice how $m(d)$ stays within the range $[0, 1[$, regardless of which values are selected. This also facilitates the parameter tuning process. For example, we used $B = 16.31$, $E = 3.2$, and set D to r_R. In Fig. 6, this would closely resemble the dashed green line, with the x-axis normalized to x times the Robot radius r_R.

Now, we could say that any obstacle closer than 1.5 times r_R to the robot's center will have almost the maximum magnitude, while obstacles more than 3 times r_R away from the robot's center will be negligible. Selecting the threshold values τ_{low} and τ_{high} is also easier. We can define them solely in terms of c^2_{max}, e.g. set the thresholds around $4 \cdot c^2_{max}$ to specify that there should more than 4 occupied cells in one particular direction to block it.

4. EXPERIMENTAL RESULTS

In order to evaluate the performance of the new algorithm we used the obstacle configuration shown in Fig. 7. The robot's goal was set 3.5 m ahead of its starting position. We measured the robot's position using an OptiTrack motion capture system. The tests were done on two different Mecanum wheeled mobile robots, the robots' dimensions are summarized on table 1. For more information on the robots used refer to Röhrig et al. (2010) and Marín (2018).

The parameters for VFH+D were selected using the aforementioned guidelines. In this configuration, the gap between $\boldsymbol{O_1}$ and $\boldsymbol{O_2}$ is too small for Robot A to pass through,

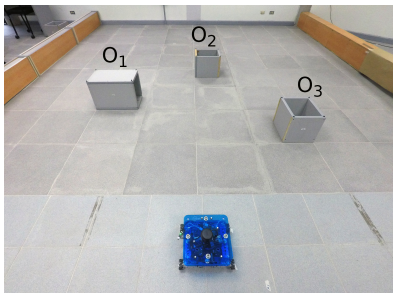

Fig. 7. Obstacle course and robot B

Table 1. Dimensions of the robots tested

	l_a	l_b	r
Robot A	29.41 cm	21.03 cm	10.15 cm
Robot B	15.5 cm	12.5 cm	3 cm

Table 2. Success rate on reaching the goal

Algorithm	Robot	Trials	Success Rate
VFH+	Robot A	10	30%
	Robot B	10	100%
VFH+D	Robot A	10	100%
	Robot B	10	100%

on the other hand $\boldsymbol{O_2}$ and $\boldsymbol{O_3}$ leave enough space for the robot to pass. Robot B, being much smaller, easily fits through both gaps.

Ten trials were conducted for each combination of robot and algorithm as shown in table 2, with the same initial conditions for all the tests. For the VFH+ algorithm with Robot A, only 3 out of 10 trials managed to reach the goal. Table 3 summarizes the results, excluding those trials where the robot could not reach the goal. We measured the Integral Absolute Error by defining the error $\epsilon(t)$ as the robot's Euclidean distance to the goal. Figures 8 and 9 show the path followed on one instance of each algorithm for robot A. Figures 10 and 11 do the same for robot B.

Our results suggest that the proposed VFH+D algorithm can have a higher success rate at traversing cluttered indoor environments. We believe this is mainly due to the improved mapping between obstacle distance and polar obstacle density. This allows the robot to correctly ignore obstacles that are too far away to impose restrictions on its movement. This was particularly hard to accomplish for VFH+, without considerably reducing the size of the active window.

Table 3. Algorithm performance comparison

Algorithm	**Robot**	**Distance traveled** [m]				**Average speed** [m/s]				**IAE** [m·s]			
		min	avg	max	stdev	min	avg	max	stdev	min	avg	max	stdev
VFH+ [1]	A	4.71	5.32	5.84	0,57	0.128	0.213	0.262	0,074	33.12	41.24	48.17	7,59
VFH+D	A	4.71	4.85	4.97	0,09	0.165	0.284	0.336	0,047	30.58	31.28	32.29	0,49
VFH+	B	3.84	3.86	3.89	0,02	0.094	0.217	0.253	0,047	28.89	30.00	36.25	2,28
VFH+D	B	3.71	3.78	4.15	0,13	0.264	0.277	0.306	0,011	26.31	27.15	27.94	0,51

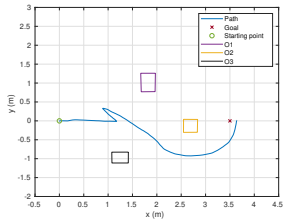

Fig. 8. VFH+ algorithm, Robot A

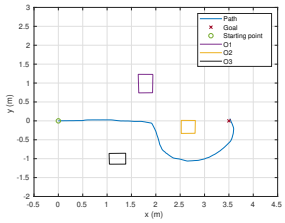

Fig. 9. VFH+D algorithm, Robot A

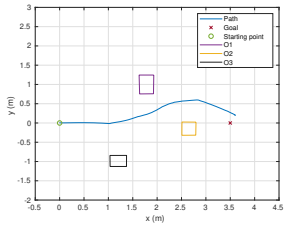

Fig. 10. VFH+ algorithm, Robot B

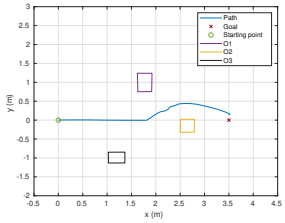

Fig. 11. VFH+D algorithm, Robot B

As for the trials where VFH+ could not reach the goal, we observed that the robot went for the opening between O_2 and O_3, but as it got closer that direction became blocked. In turn the robot backed up, until the obstacles were far away enough that once again the direction between O_2 and O_3 was unblocked, and the cycle repeated. Effectively, the robot was stuck in a "local minimum". Once the robot has failed in its first attempt to clear this opening, it's very unlikely that any further attempt will succeed, because of the blurring effect explained in section 2 and Fig. 2. We attributed the initial success or failure of the runs to slight variations in quantization of the obstacles' and robot's position to the discrete data structures of the grid.

VFH+D also achieved a higher average speed on both robots. This may be expected, if the speed is calculated as a function of the obstacle density in the robot's direction of movement (higher density means lower speed). Given that the obstacle density for VFH+D is lower for far away obstacles, the robot's movement speed when traveling in the direction of such obstacles will be higher. This is often the case for indoor environments, where walls are present in every direction and frequently fall within the active window.

Additionally, VFH+ as defined in Ulrich and Borenstein (1998) cannot deal with moving obstacles without becoming trapped. The addition of decay to the active window allows VFH+D to escape such situations.

Another aspect which is not immediately obvious from our results is that VFH+D allows for better overall performance with smaller data structures. For example, while tuning VFH+ to allow it to somewhat successfully clear the obstacle course, we found that H^p needs at least 160 sectors, while our parameterization of VFH+D required only 100 sectors to successfully reach the goal on every iteration. Intuitively, we interpret this as a better or more efficient codification of the information necessary to correctly decide which path to follow.

5. CONCLUSIONS

This paper presents VFH+D, an improvement on VFH+ for local obstacle avoidance. The proposed changes are:

- The introduction of cell occupancy decay, which allows for dynamic obstacle avoidance (algorithm 1).
- A new obstacle vector magnitude equation for cells in the active window, as described by (2).

Also, it was shown that the parameter tuning for VFH+D is more intuitive and requires less iterations.

To measure the new algorithm's performance, a total of 40 trials with 2 different robots on one obstacle course with static obstacles were conducted. The results show that the new algorithm incurred in lower average distance traveled to reach the goal and higher average speed, and achieved consistently a lower IAE. For one of the robots tested, VFH+ failed to reach the goal on 7 out of 10 trials, while VFH+D reached the goal on every iteration.

REFERENCES

Adla, R., Al-Holou, N., Murad, M., and Bazzi, Y.A. (2013). Automotive collision avoidance methodologies sensor-based and its-based. In *2013 ACS International Conference on Computer Systems and Applications (AICCSA)*, 1–8.

Alajlan, A.M., Almasri, M.M., and Elleithy, K.M. (2015). Multi-sensor based collision avoidance algorithm for mobile robot. In *2015 Long Island Systems, Applications and Technology*, 1–6.

Almasri, M.M., Alajlan, A.M., and Elleithy, K.M. (2016). Trajectory planning and collision avoidance algorithm for mobile robotics system. *IEEE Sensors Journal*, 16(12), 5021–5028.

Alonso-Mora, J., Beardsley, P., and Siegwart, R. (2018). Cooperative collision avoidance for nonholonomic robots. *IEEE Transactions on Robotics*, 34(2), 404–420.

Berns, K. and von Puttkamer, E. (2009). *Autonomous Land Vehicles*. Vieweg+Teubner Verlag.

Borenstein, J. and Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3), 278–288.

[1] For VFH+ with robot A, includes only the 3 trials that reached the goal

Braitenberg, V. (1986). *Vehicles: Experiments in synthetic psychology*. MIT press.

Chen, Y. and Li, T. (2017). Collision avoidance of unmanned ships based on artificial potential field. In *2017 Chinese Automation Congress (CAC)*, 4437–4440.

Cybulski, B., Wegierska, A., and Granosik, G. (2019). Accuracy comparison of navigation local planners on ros-based mobile robot. In *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*, 104–111.

Dahl, J., de Campos, G.R., Olsson, C., and Fredriksson, J. (2019). Collision avoidance: A literature review on threat-assessment techniques. *IEEE Transactions on Intelligent Vehicles*, 4(1), 101–113.

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1), 23–33.

Guo, J., Zhang, S., Xu, J., and Zhou, S. (2010). Kalman prediction based VFH of dynamic obstacle avoidance for intelligent vehicles. In *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, volume 3, V3–6–V3–10.

Houjie Jiang, Zhuping Wang, Qijun Chen, and Jin Zhu (2016). Obstacle avoidance of autonomous vehicles with cqp-based model predictive control. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 001668–001673.

Kim, D., Kim, J., Bae, J., and Soh, Y. (2010). Development of an Enhanced Obstacle Avoidance Algorithm for a Network-Based Autonomous Mobile Robot. In *2010 International Conference on Intelligent Computation Technology and Automation*, volume 2, 102–105.

Lefebvre, O., Lamiraux, F., Pradalier, C., and Fraichard, T. (2004). Obstacles avoidance for car-like robots integration and experimentation on two robots. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 5, 4277–4282 Vol.5.

Lumelsky, V. and Skewis, T. (1990). Incorporating range sensing in the robot navigation function. *IEEE Transactions on Systems, Man, and Cybernetics*, 20, 1058–1068.

Lumelsky, V. and Stepanov, A.A. (1987). Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2, 403–430.

Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., and Konolige, K. (2010). The office marathon: Robust navigation in an indoor office environment. In *2010 IEEE International Conference on Robotics and Automation*, 300–307.

Marín, L. (2018). Modular open hardware omnidirectional platform for mobile robot research. In *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*, 1–6.

Marín, L., Valles, M., Valera, A., and Albertos, P. (2010). Implementation of a bug algorithm in the e-puck from a hybrid control viewpoint. In *2010 15th International Conference on Methods and Models in Automation and Robotics*, 174–179.

Miró, J.V., Taha, T., Wang, D., and Dissanayake, G. (2008). An adaptive manoeuvring strategy for mobile robots in cluttered dynamic environments. *IJAAC*, 2, 178–194.

Ng, J. and Bräunl, T. (2007). Performance comparison of bug navigation algorithms. *Journal of Intelligent and Robotic Systems*, 50(1), 73–84.

Pandey, A., Pandey, S., and Parhi, D. (2017). Mobile robot navigation and obstacle avoidance techniques: A review. *International Robotics & Automation Journal*, 2(3), 00022.

Patle, B., L, G.B., Pandey, A., Parhi, D., and Jagadeesh, A. (2019). A review: On path planning strategies for navigation of mobile robot. *Defence Technology*, 15(4), 582 – 606.

Roesmann, C., Feiten, W., Woesch, T., Hoffmann, F., and Bertram, T. (2012). Trajectory modification considering dynamic constraints of autonomous robots. In *ROBOTIK 2012; 7th German Conference on Robotics*.

Röhrig, C., Heß, D., Kirsch, C., and Künemund, F. (2010). Localization of an omnidirectional transport robot using ieee 802.15.4a ranging and laser range finder. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3798–3803.

Sary, I.P., Nugraha, Y.P., Megayanti, M., Hidayat, E., and Trilaksono, B.R. (2018). Design of Obstacle Avoidance System on Hexacopter Using Vector Field Histogram-Plus. In *2018 IEEE 8th International Conference on System Engineering and Technology (ICSET)*, 18–23.

Sezer, V. and Gokasan, M. (2012). A novel obstacle avoidance algorithm: "follow the gap method". *Robotics and Autonomous Systems*, 60(9), 1123 – 1134.

Shayestegan, M. and Marhaban, M.H. (2012). A braitenberg approach to mobile robot navigation in unknown environments. In S.G. Ponnambalam, J. Parkkinen, and K.C. Ramanathan (eds.), *Trends in Intelligent Robotics, Automation, and Manufacturing*, 75–93. Springer Berlin Heidelberg.

Shim, Y. and Kim, G.W. (2018). Range sensor-based efficient obstacle avoidance through selective decision-making. *Sensors*, 18(4).

Soriano, A., Bernabeu, E.J., Valera, A., and Vallés, M. (2015). Collision avoidance method based on consensus among mobile robotic agents. *International Journal of Imaging and Robotics*, 15(1), 80–90.

Ulrich, I. and Borenstein, J. (1998). VFH+: reliable obstacle avoidance for fast mobile robots. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, volume 2, 1572–1577 vol.2.

Ulrich, I. and Borenstein, J. (2000). VFH*: local obstacle avoidance with look-ahead verification. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 3, 2505–2511 vol.3.

Wang, P., Gao, S., Li, L., Sun, B., and Cheng, S. (2019). Obstacle avoidance path planning design for autonomous driving vehicles based on an improved artificial potential field algorithm. *Energies*, 12(12).

Yim, W.J. and Park, J.B. (2014). Analysis of mobile robot navigation using vector field histogram according to the number of sectors, the robot speed and the width of the path. In *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, 1037–1040.