



Low-cost ultrasonic based object detection and collision avoidance method for autonomous robots

Jawad N. Yasin¹ · Sherif A. S. Mohamed¹ · Mohammad-Hashem Haghbayan¹ ·
Jukka Heikkonen¹ · Hannu Tenhunen² · Juha Plosila¹

Received: 1 June 2020 / Accepted: 7 September 2020
© The Author(s) 2020

Abstract This work focuses on the development of an effective collision avoidance algorithm that detects and avoids obstacles autonomously in the vicinity of a potential collision by using a single ultrasonic sensor and controlling the movement of the vehicle. The objectives are to minimise the deviation from the vehicle's original path and also the development of an algorithm utilising one of the cheapest sensors available for very low cost systems. For instance, in a scenario where the main ranging sensor malfunctions, a backup low cost sensor is required for safe navigation of the vehicle while keeping the deviation to a minimum. The developed algorithm utilises only one ultrasonic sensor and approximates the front shape of the

detected object by sweeping the sensor mounted on top of the unmanned vehicle. In this proposed approach, the sensor is rotated for shape approximation and edge detection instead of moving the robot around the encountered obstacle. It has been tested in various indoor situations using different shapes of objects, stationary objects, moving objects, and soft or irregularly shaped objects. The results show that the algorithm provides satisfactory outcomes by entirely avoiding obstacles and rerouting the vehicle with a minimal deviation.

Keywords Collision avoidance · Fault tolerance · Mobile robots · Ultrasonic · Unmanned vehicles

This work has been supported in part by the Academy of Finland-funded research project 314048 and Finnish Cultural Foundation.

✉ Jawad N. Yasin
janaya@utu.fi

Sherif A. S. Mohamed
samoha@utu.fi

Mohammad-Hashem Haghbayan
mohhag@utu.fi

Jukka Heikkonen
jukhei@utu.fi

Hannu Tenhunen
hannu@kth.se

Juha Plosila
juplos@utu.fi

¹ Autonomous Systems Laboratory, Department of Future Technologies, University of Turku, Vesilinnantie 5, 20500 Turku, Finland

² Department of Industrial and Medical Electronics, KTH Royal Institute of Technology, Brinellvgen 8, 114 28 Stockholm, Sweden

1 Introduction

An unmanned vehicle can be categorised as a mobile robot that has no human pilot onboard. This includes, for instance, Unmanned Aerial Vehicles (UAV), Unmanned Ground Vehicles (UGV), and Unmanned Surface Vehicles (USV). Their operation is controlled either manually using remote controllers or autonomously based on onboard computers [1].

An Unmanned Aerial Vehicle (UAV), or a drone, is an aircraft, often a multirotor which is basically a rigid body with mechanically moveable blades. UAVs are designed for dangerous and complicated missions as the replacements of manned aircraft. Unmanned Ground Vehicles (UGV) and Unmanned Surface Vehicles (USV) are counterparts of UAVs moving on the ground and water surface, respectively. In general, unmanned vehicles have different types of sensors that enable situational awareness and autonomous decision making depending on the tasks at hand [2]. Control can be manual, based on e.g. live video

received from a camera mounted on a vehicle (remote control); autonomous, based on feedback received from a mounted camera and other types of sensors indicating the approaching obstacles [3–5]; or something between these two extremes (a hybrid method, semi-autonomous). Bearing in mind the considerably low risk to human life, as well as improved durability for longer missions and accessibility in difficult terrains, the demand for such unmanned vehicles is increasing rapidly and their path planning in dynamic environments remains one of the most challenging issues to solve [6]. Due to their autonomy and ability to travel far from the base stations or their operators (the range naturally depends on the type and size of the vehicle), the need for having an onboard mechanism to avoid collisions with objects and other vehicles is obvious [7–10].

1.1 Motivation

Consider a swarm of mobile robots such as drones operating in different layers as shown in Fig. 1, where drones at Layer 2 act as masters for drones at Layer 1. Any layer has the ability to perform different tasks depending on the mission. The drones in a swarm communicate with each other as well as observe their surroundings. The organization can be cluster based and comprise multiple hierarchy levels so that each master drone has several followers at the next lower layer (forming a cluster), and, correspondingly, a group of masters at a given layer has a common leader at the next higher layer. The mission is typically to monitor and extract information from the targeted environment and to send this information to the base station for further processing and analysis.

To execute their assignments, drones are equipped with different types of sensors capable of collecting the required information [11]. Different drones in a swarm can have different sets of sensors, depending on the task. At the same time, all drones should be able to avoid collisions in real time based on the sensory data [12, 13]. Indeed, a dense swarm of drones emphasises the need for having an efficient embedded collision avoidance technique/methodology so that the UAVs can individually detect and avoid

each other and other objects/obstacles in their neighbourhood [14]. The drones move from the base station to the target area in a given formation, execute the required monitoring task, and then return back to the base station. During the flight, they can encounter both stationary and moving obstacles and objects that need to be safely and reliably evaded [15].

In this paper, we develop a sonar-based collision avoidance algorithm for unmanned vehicles and test it on a set of stationary objects. We verify our algorithm using a simple mobile robot (LEGO MINDSTORMS). The algorithm is able to:

- guide the vehicle on how to proceed by calculating the distance between itself and the detected obstacle,
- control the speed of the vehicle by slowing it down and bringing its horizontal speed to zero, making it hover on its position,
- force the vehicle to turn in order to avoid collisions.

1.2 Related work

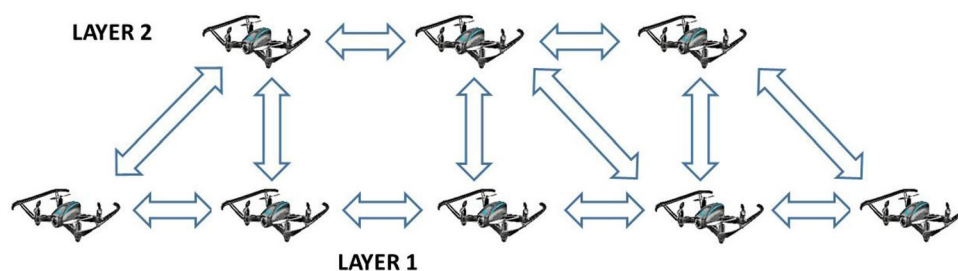
Various sensors can be utilised in the implementation of collision avoidance mechanisms for unmanned vehicles [16]. In this section, we discuss some of the approaches found in the literature.

The author in [17] proposes a 3D reactive obstacle avoidance technique. The algorithm detects an obstacle in a UAV's path, makes the craft hover on its position, calculates the best escape route, and then instructs the UAV accordingly. The method is demonstrated using stereo and laser-based sensing techniques.

In [18] the author proposed a solution in which the paths are based on different line colours. A robot having the ability to distinguish between numerous colours can select the desired line autonomously to reach the target. This system is not viable in dynamic environments nor in bad lighting conditions. Furthermore, the robot is totally dependent on the visibility of the lines and does not take into account the presence of an obstacle on the line itself, lacking dynamic capabilities completely.

In [19] the author proposed to equip the vehicles with adaptive cruise control along with the collision avoidance

Fig. 1 Swarm of UAVs with two layers



system in such a way that collisions are avoided by braking for vehicles at slower speeds and by steering with vehicles at higher speeds.

The authors in [20] proposed an Artificial Potential Field for finding the shortest path between starting and destination points. A robot is repelled from and attracted towards the obstacle and target points respectively, due to the repulsive and attractive forces generated by the respective points. Based on the repulsive and attractive forces, the robot calculates the aggregate amount of force. The drawback of this method is that, for symmetric environments, it is very sensitive to local minima and therefore does not necessarily lead to a globally optimised solution [21].

In [22], forward-looking cameras are used for real-time obstacle detection and avoidance. The presented fuzzy control based method is in principle applicable to different types of unmanned vehicles, in the paper, it experiments on a small quadrotor UAV. The authors use a camera that is mounted in front of UAV to avoid collisions via visual servoing through image processing. In this approach, the collected data is wirelessly sent to a laptop for further processing, where obstacles are marked with specific colours, and this information is then employed to guide the UAV around the obstacles. The algorithm avoids the obstacles by pushing them to either the left or right side of the image. A potential problem in this setup is that communication delays between the drone and the controlling computer can lead to an accident in the case of a close obstacle or an obstacle moving towards the UAV.

In [23], the authors propose a methodology which uses two cameras for detecting the obstacles in the range of 30 to 100 meters and up to the speed of about 22kmph. In order to differentiate between the sea and sky, this approach relies on the sea-sky line and assumes that the obstacles are moving in a regular manner. Different filters are applied to detect the obstacles. A limitation of the scheme is that it does not take into account rough sea waves, haphazardly moving obstacles and overcast situations.

In the method presented in [24], LiDAR (Light Detection and Ranging) is used, by generating light in the form of a pulsed laser, to measure the distances to the earth. In [25], a simulated UAV mounted with a LiDAR sensor is inspected using a feed-forward based algorithm. The UAV is mainly controlled by the operator, and the algorithm estimates the path of the UAV by using the current inputs from the operator and the future for a predefined period of time. The algorithm checks for any possible collisions with objects and diverts the UAV from the original path when needed by keeping it as close to the operator's input as possible.

The authors in [26] used computer vision technique for the detection of animals and avoiding collision with them. They used over 2200 images to train their system and performed tests using video clips of animals on highways. The algorithm provided satisfactory results with 82.5% accuracy and detected the animals in order to avoid a collision. However, their proposed solution is highly speeding dependent and will not help in preventing collisions at speeds over 35km/h. In fact, at higher speeds, it may not be able to detect objects at all. Furthermore, the provided solution will provide bad to worst results especially in bad weather conditions, in bad or too bright light, in foggy situations, as well as in shiny surroundings.

The authors in [27] used five US sensors along with a predefined neural network module in MATLAB to triangulate and detect the precise position and shape of the objects. The authors used only three different shaped objects for their testing. Furthermore, the five US sensors used in their solution are more than required as the precise location of the detected objects can be found using only two US sensors and the third dimension (depth) can be found by adding the third US sensor. Moreover, their results are satisfactory when the objects are regular shaped, but for irregular objects, they report that their network is not able to correctly identify the objects.

Our collision avoidance algorithm, proposed in this paper, approximates the shape of an object by rotating the ultrasonic sensor while approaching and passing the object, avoiding it completely. The rotational movement of the sensor helps in identifying the exact location and corners (if in visible range of the sensor) of the detected obstacle without deviating from original path. Moreover, a fault tolerance method is deployed for ensuring correct operation and eliminating any false readings.

1.3 Organisation

The rest of the paper is organised as follows. The equipment used in our work is described in Sect. 2. In Sect. 3, the development of the proposed algorithm is described. Experiments on different cases with corresponding results are presented in Sect. 4. Lastly, Sect. 5 concludes the paper.

2 Equipment and programming interface

Using a wheeled robot or UGV is very feasible in order to simulate and study the collision avoidance algorithm in real-time without jeopardising the equipment and the surroundings. A UGV behaves, namely, precisely in the same way as a UAV in a 2-D plane. Moreover, UGVs have benefits such as cost efficiency and the availability of

equipment. Hence, LEGO MINDSTORMS NXT 2.0 can be used for designing and testing the algorithm.

The LEGO MINDSTORMS packages, which are widely available, consists of a programmable intelligent micro-computer brick NXT 2.0, interactive servo motors, Ultrasonic Sensor (US), touch sensor, sound sensor, light sensor, LEGO building blocks/parts, and an NXT rechargeable battery. The parts are easy to assemble and it is convenient to modify the design if required for re-testing. The NXT brick consists of a 32-bit microprocessor, 4 input ports and 3 output ports, a large matrix display for displaying the required outputs/messages, and it connects via USB or Bluetooth links.

The standard software that comes with the package is NXT-G, which is a graphical programming interface (LabVIEW) having a drag and drop environment. LeJOS replaces the firmware on Lego bricks, and LeJOS NXJ is the programming interface widely used for programming in the Java language. The LeJOS-based robot Jitter flew around the International Space Station in December 2001 [28, 29]. Some of other third party programming interfaces are MATLAB and Simulink, NXTGCC, Robotics. NXT, Lego NXT, ROBOTC, RoboMind, and ruby-nxt.

In this work, Eclipse is used as the programming framework to accomplish the tasks. This is a customised/integrated development environment (IDE) for Java. Eclipse is the most commonly and widely used tool for Java IDE that itself a high-level open source language.

3 Proposed approach

There are various methods and techniques available for object detection and collision avoidance. These methods and techniques use a wide variety of different sensors such as radars, laser range finder, sonar sensors, cameras for detecting surroundings [30, 31]. Depending on the application area, a different set of sensors and algorithms will be used. In this paper, we present a collision avoidance algorithm for a UGV (a wheeled robot) that utilises an ultrasonic (US) sensor. A more thorough description of the algorithm and its investigation is given in [32].

The proposed collision avoidance algorithm works on a relatively simple model. It works in the xy plane keeping the z-axis fixed; in other words, the algorithm is responsible for controlling the horizontal speed to avoid collisions in a 2-D plane keeping the height fixed. The algorithm performs the following actions when a US sensor shows the presence of an obstacle:

- Object detection: responsible for alerting the presence of objects in the vicinity,

- Speed Control and Decision: responsible for slowing down the speed if a UGV is approaching the object and stop the UGV when the object is within a critical range,
- Locate and edge detection: responsible for scanning the object in front and determining the edges of the object,
- Fault tolerance: responsible for removing any false readings,
- Triangulation: responsible for calculating the exact angles and distances of the object,
- Coordinates determination and decision: search for alternative routes to avoid the object and re-route the UGV if needed.

The basic flowchart of the algorithm is shown in Fig. 2. The operation starts by initialising the US sensor by setting the maximum detection distance and the minimum and maximum angles for the rotation of the US sensor. In the experiment, the values are 170 cm, 45° and 135° , respectively. The algorithm does not restrict the maximum rotation angle; it depends on a UGV, its locomotion type and the location of US sensors. The US sensor is rotated in between the given angles and thus enabling continuous sweep of the surroundings.

Figure 3 shows the field of view of a US sensor. By default, a US sensor can only detect an object in its field of

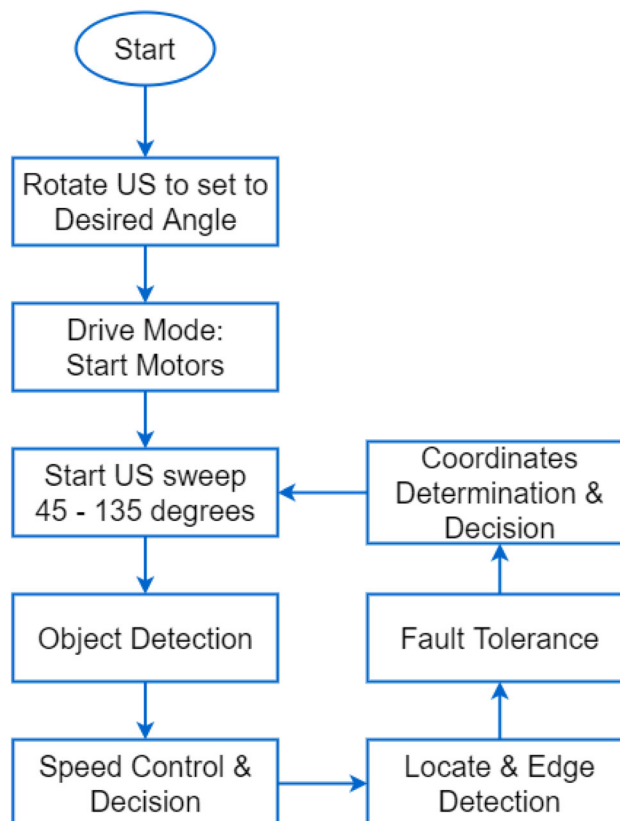


Fig. 2 Overall flowchart of object detection and collision avoidance algorithm

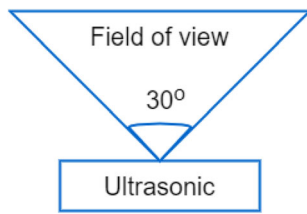


Fig. 3 Ultrasonic field of view

view. So we need to develop a methodology to find the exact location of the detected object. The US sensor starts sweeping between the defined angles and scanning for any objects in its vicinity. When the minimum angle of 45° is reached, the sensor actually covers 15° making the coverage from 30 60 . Object Detection is then initiated to determine if there are any objects or obstacles in the range.

If there are any objects detected in the range of the US sensor, the Speed Control module is called and works sequentially in a loop with the sensor rotation module. This part of the algorithm is responsible for continuously slowing down the speed of the UGV as the object gets closer. The flow chart diagram of the Speed Control module is shown in Fig. 4. This module works by reading the range from the US sensor and then going through multiple conditions accordingly.

Once the detected object enters the critical range of the US sensor, the UGV is stopped and the Locate and Edge Detection part of the algorithm is activated (see Fig. 5). This module locates and determines the exact angles at which the detected object lays and how far the detected object continues.

The Locate and Edge Detection block (LaED) is very important as the US sensor itself cannot locate the exact angle at which the object lies, since it has a field of view of 30 degrees. Anything within the detecting range of 30 degrees is detected. Thus, LaED is designed to help the US sensor not only to get the angle of the object along with its distance but also to detect where the object ends i.e. the front edges of the object. LaED starts by copying the object's distance in a temporary variable and compares it with the initial detected distance of the object. The US sensor is rotated degree by degree to the left side and the new distance reading is compared with the original reading. If the edge is found, and the left edge flag is raised and the US sensor is rotated back to the initially angle of detection and starts rotating to the right side.

If the edge is not found and the physical limitation of rotation is reached, the edge flag is reset. Similar steps are taken for the right side edge detection. After this, the Fault Tolerance block (see Fig. 6) is initiated to cross-check the

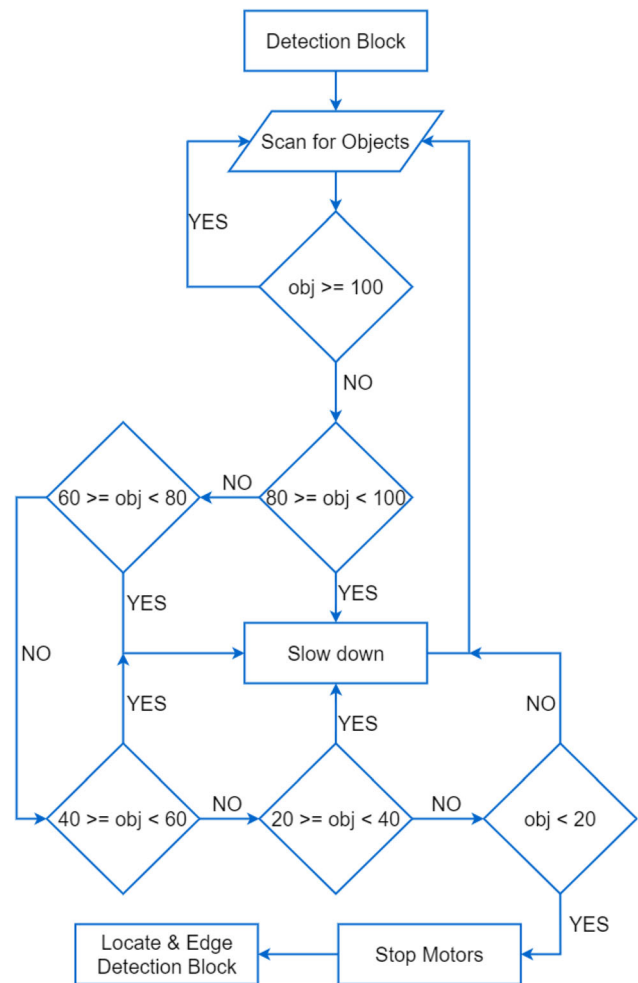


Fig. 4 Speed control decision flow chart

detected results and confirm if any of the detected edges was a false edge.

Upon successful detection of the edges or corners of the object, the algorithm calls the Fault Tolerance in order to cross-check any errors in the detected results from the Locate and Edge Detection module. A false edge can lead to wrong calculations and slow down the process of redirection. When LaED detects an edge, then this block crosschecks the result for the next three consecutive iterations (see Fig. 6). If the variables `error_check` and `edge_count` are equal and have been cross-checked three times consecutively, it means that there is an edge. On the other hand, if the edge is not found for three consecutive iterations, the algorithm will reset the edge flag indicating that the detected edge was a false edge. The flow-chart diagram for Fault Tolerance module is given in Fig. 6.

Then the Triangulation module is activated; this module of the algorithm uses the results from LaED to calculate the exact angles and distance at which the object lies from the

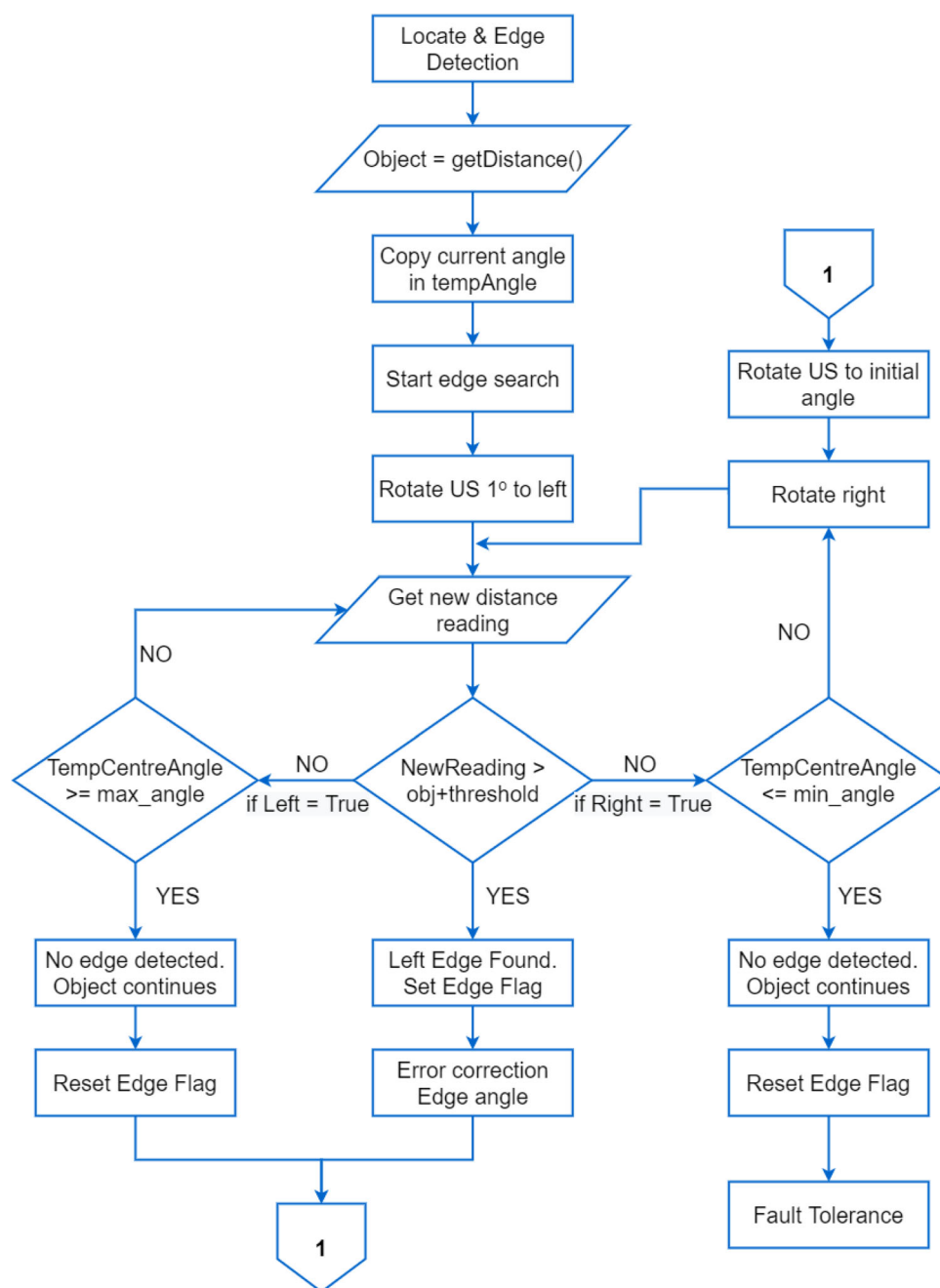


Fig. 5 Locate and edge detection flow chart

US sensor or the UGV (see Figure 7). If the detected edge of the object is at point P1 and angle 1, as shown in Figure 7, the distance from the US sensor to P1 is also known, h1. Then using the right-angled triangle equations, the rest of the sides of the triangle can be found [33], pp.3–4].

$$\sin\theta = \frac{\text{opposite}}{\text{hypotenuse}} \tag{1}$$

$$\cos\theta = \frac{\text{adjacent}}{\text{hypotenuse}} \tag{2}$$

$$\tan\theta = \frac{\text{adjacent}}{\text{opposite}} \tag{3}$$

The variables and their explanations are given below.

d = the width of the UGV

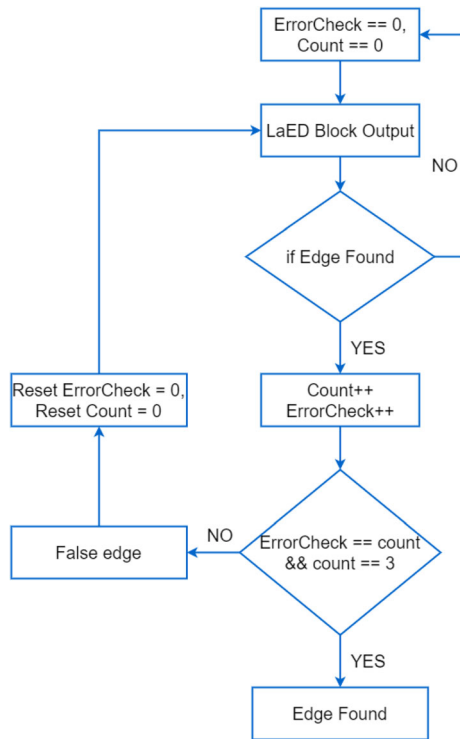


Fig. 6 Fault-tolerance flowchart

f_s = minimum safe distance allowed from the side of UGV

h_1 and h_2 = diagonal detected distance of the object (edges) on the left and right sides respectively

a_1 and a_2 = distance of the object on the left and right side of the UGV respectively, calculated from the centre of the UGV

o_1 and o_2 = distance of the object straight ahead of the UGV

P_1 and P_2 = Point where the object/edge was detected by the US sensor

θ_1 and θ_2 = angles of the edges

ϕ_1 and ϕ_2 = linearised angles

Now, we know the angles θ_1 and θ_2 , the diagonal distances h_1 and h_2 .

The unknowns are calculated by using the following equations:

$$\phi_1 = 180 - \theta_1$$

$$a_1 = h_1 \cos(\phi_1)$$

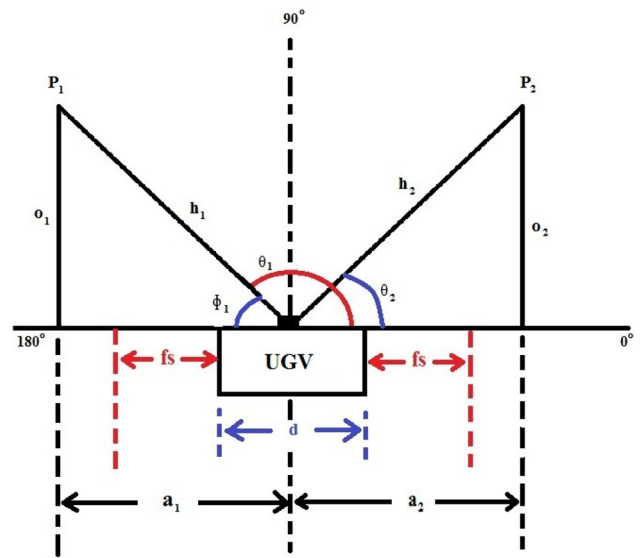


Fig. 7 Triangulation method used for calculating the exact horizontal and vertical distance of the edges of the object from the UGV [29]

$$o_1 = h_1 \sin(\phi_1)$$

$$\phi_2 = \theta_2$$

$$a_2 = h_2 \cos(\phi_2), \text{ and } o_2 = h_2 \sin(\phi_2)$$

Once the exact coordinates of the object have been calculated, all possible combinations of the object's location can be analysed and the decision is made accordingly.

In the final step, the decisions are made based on the coordinates of the detected object. After the exact coordinates, angles, and distance of the object are calculated, it is decided whether the UGV can continue in its original course or not. The decision takes into account the dimensions of the UGV as well. If rerouting is needed, it will be decided in what direction the UGV should be rotated to

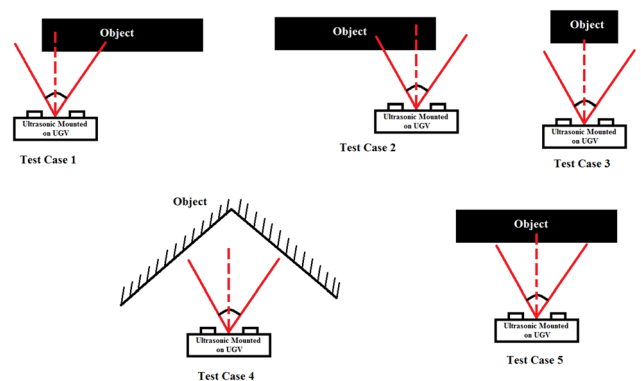


Fig. 8 Visual description of the test cases: object in front of the UGV **a** with only the left edge visible, **b** with only the right edge visible, **c** with both edges in detectable range, **d** with a V-shape, **e** with no edges in the detectable range

achieve the shortest possible deviation from the original path.

4 Experimental results

In order to validate the collision avoidance algorithm, a number of test cases using a UGV in real-time are considered. These cases cover all possible situations when the robot approaches an object and the object in front of the robot has:

1. a visible left edge, but no right edge,
2. a visible right edge, but no left edge,
3. both edges visible,
4. both edges closing in from the sides i.e. a V-shaped object, and
5. no visible edges.

It is important to note here that in all these cases the objects are stationary while the robot is moving. These cases are visualised in Fig. 8.

The robot scans for objects by sweeping the US sensor. When the object enters the visible range, the Speed Control block of the algorithm start to slow the robot down. As the object is close enough to appear in the defined critical range for safety, the robot stops and the Locate and Edge Detection block starts to determine the edges and the location of the detected object.

4.1 Test case 1: edge on the left side visible

Consider a scenario in which the robot approaches an object that has a visible edge or opening on its left side, but continues to the right outside the visibility range as shown

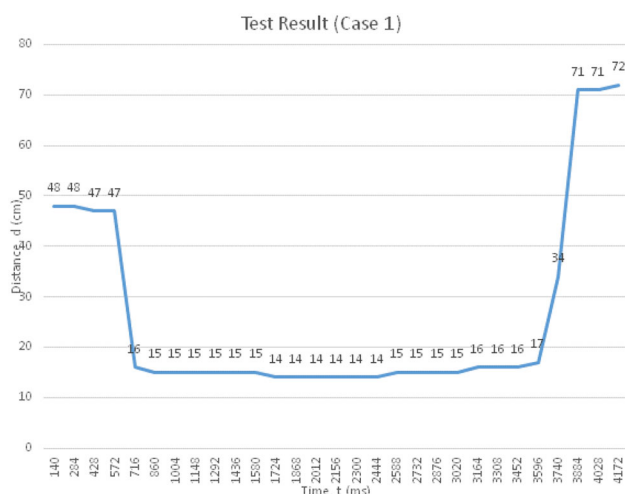


Fig. 9 Test result of case study 1

in Fig. 8a. The robot should be able to see the nearest opening to its left side and turn left to avoid the object by a minimum deviation from its original path.

In Fig. 9, the robot detects the edge at $t = 572$ ms as there is a drastic change in the distance to the object, which is further confirmed by the Fault Tolerance block of the algorithm. At $t = 3884$ ms, the physical limitation of the visibility range of the system is reached and the algorithm decides that there is no edge on this side. The exact distance to the object from the UGV is determined. Since the object's right edge is within the critical colliding distance of the UGV, the robot turns left to avoid the object and then moves forward.

4.2 Test case 2: edge on the right side

In the second test case (see Fig. 8b), the robot approaches an object that has an edge or opening on its right side but continues to the left outside the visibility range. The robot should be able to see the nearest opening on its right side and turn right to avoid the object by a minimum deviation from its original path.

It is significant to note in this case, as depicted in Fig. 10, that the value on the right side of the robot changed suddenly from 17 cm to 28 cm at $t = 4464$ ms and then 300 ms later to 36 cm. This is the threshold level for considering it as an edge. However, on the left side, it can be noted that the value is 255 when the physical limitation of the rotation of the US sensor is reached and it cannot be rotated further in that direction. This is due to the fact that the algorithm is designed to give value 255 if there is no edge and the object continues. The robot decides to turn right using the results of Coordinates Determination and Decision block's calculations.

4.3 Test case 3: both edges in visible range

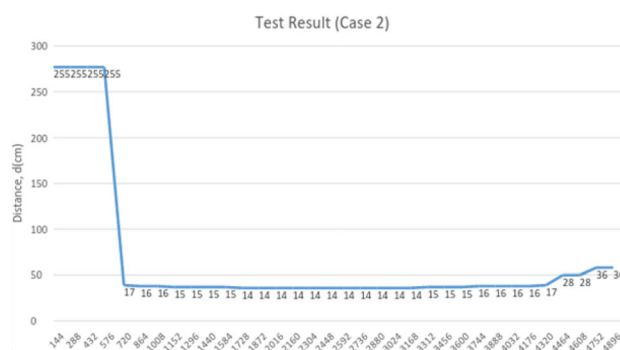


Fig. 10 Test result of case study 2

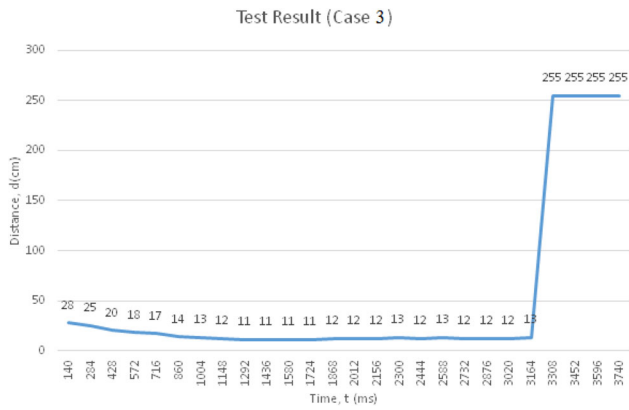


Fig. 11 Test result of case study 3

In the third case (see Fig. 8c), the robot approaches an object where both edges of the object are in the detectable range of the sensor. The robot should be able to detect the nearest edge of the object to its left and turn left to avoid the object by a minimum deviation from its original path.

Using the results of the blocks Triangulation, as well as Coordinates Determination and Decision, the algorithm calculates the exact coordinates for the edges of the object and decides to turn the robot left. From the test result in Fig. 11, it is visible that the change in the distance became significant enough for an edge from $t = 140$ ms to $t = 860$ ms.

4.4 Test case 4: V-shaped obstacle

Test Case 4 (see Fig. 8d) describes a test scenario where the robot approaches a V-shaped obstacle. When the robot goes deeper into the V-shape and the walls of the obstacle come closer to its sides, it starts to slow down. The robot stops and the US sensor starts scanning the obstacle when the distance to either side of the obstacle is equal to the allowed minimum safe distance to the side. In this case, the

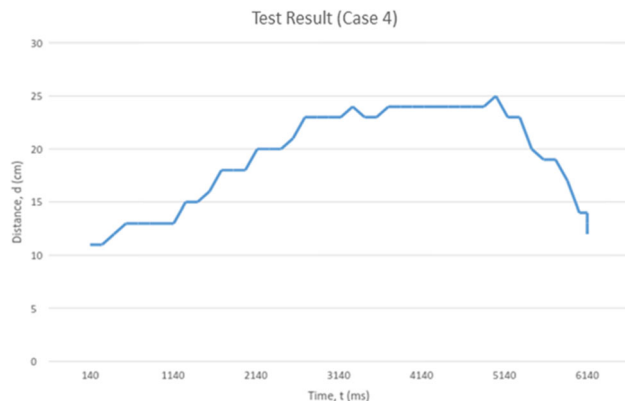


Fig. 12 Test result of case study 4

robot reverses to the point where it can take a turn and find a better alternate route.

The test result is shown in Fig. 12, where it can be seen that the distance increases above the threshold level for detecting an edge, in the centre of the graph. The robot does, however, not detect it as an edge, because the condition for critical safe distances to the sides is not satisfied. The condition indicates that the object is in the critical colliding range to the side of the UGV. Since the side safe distance condition is not met, the robot decides that it is not safe to move forward and reverses instead.

4.5 Test case 5: obstacle with no edges in visible range

In the last case study, shown in Fig. 8e, the robot approaches an obstacle where none of the edges of the object is in the detectable range of the sensor. When the sweeping is completed and none of the edges of the obstacle is found, the robot turns left to avoid the obstacle using prioritised turning. In prioritised turning, there are either no edges in the visible range or both of the edges are located at exactly the same distance.

From the test outcome in Fig. 13, it is clearly visible that the distance to the obstacle is almost constant, except at both ends of the graph where the value of the distance is 255. This indicates that no edges were found at the ends and that the US sensor could not be rotated further due to the physical limitations of the hardware design. However, another significant point to note here is that approximately at $t = 3300$ ms a false edge is detected by the US sensor. Upon detecting this gap, the algorithm initiates the Fault Tolerance block of the algorithm, which identifies it as a false edge and ignores it.

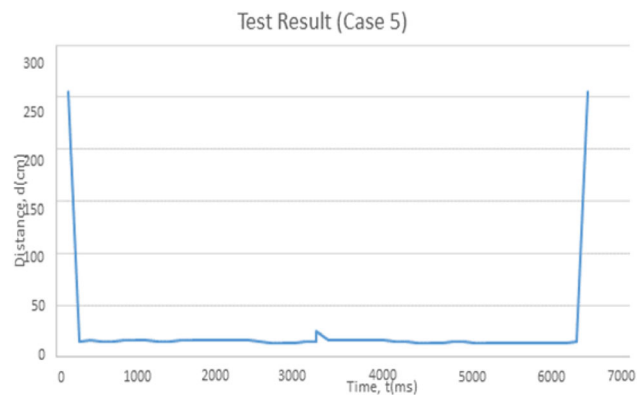


Fig. 13 Test result of case study 5

5 Conclusion

In this paper, we developed collision avoidance algorithm, independent of lighting conditions, using single ultrasonic sensor for shape estimation and collision avoidance for autonomous robots. We investigated the behaviour of our collision avoidance algorithm and tested it in real-time by using a simple UGV. All the performed experiments were successful and the UGV rerouted itself to avoid the collision while keeping the deviation to a minimum. However, if the detected object moves out of the visible range, the UGV is brought back to its original travelling speed. Moreover, the developed fault-tolerance function in this algorithm improves the outcomes by detecting false alarms within milliseconds and eliminates the chances of collisions. The proposed algorithm was also investigated with moving objects/obstacles, especially by walking in the way of the UGV and by moving objects such as a box, laptop, mobile phone, and other objects in its way. Consistently, the UGV detected all the moving objects in its way. However, this algorithm is studied and proved to be working well only for the indoor and controlled environment and its performance outdoors in more uncontrolled environments still needs to be inspected. Currently, this algorithm is being extended and tested in a swarm of robots for its performance and verification, along with its ability to successfully detect and avoid dynamic obstacles while keeping the formation.

Acknowledgements This work has been supported in part by the Academy of Finland-funded research project 314048 and Finnish Cultural Foundation.

Funding Open access funding provided by University of Turku (UTU) including Turku University Central Hospital.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Unmanned aerial vehicle (2020). https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle#Market_trends. Page Version ID: 956436479. Accessed 29 May 2020
2. Zhang W, Zelinsky G, Samaras D (2007) Real-time accurate object detection using multiple resolutions. In: 2007 IEEE 11th International Conference on Computer Vision, pp. 1–8
3. Shinde SN, Chorage S (2016) Unmanned ground vehicle. *Int J Adv Eng Manag Sci* 2:1659
4. Iqbal J, Pasha SM, Baizid K, Khan AA, Iqbal J (2013) Computer vision inspired real-time autonomous moving target detection, tracking and locking. *Life Sci J* 10(4):3338–3345
5. Rabah M, Rohan A, Talha M, Nam KH, Kim SH (2018) Autonomous vision-based target detection and safe landing for uav. *Int J Control Autom Syst* 16(6):3013–3025
6. Zhuge C (2017) A novel dynamic obstacle avoidance algorithm based on collision time histogram. *Chin J Electron* 26:522–529. <https://doi.org/10.1049/cje.2017.01.008>
7. Chao H, Cao Y, Chen Y (2007) Autopilots for small fixed-wing unmanned air vehicles: a survey. In: 2007 International Conference on Mechatronics and Automation, pp. 3144–3149
8. Yasin JN, Mohamed SAS, Haghbayan M, Heikkonen J, Tenhunen H, Plosila J (2020) Unmanned aerial vehicles (uavs): collision avoidance systems and approaches. *IEEE Access* 8:105139–105155
9. Vijayavargiya A, Sharma A, Anirudh Kumar A, Kumar A, Yadav A, Sharma A, Jangid A, Dubey A (2016) Unmanned aerial vehicle. *Impl J Interdiscip Res* 2(5):1747–1750
10. Yasin JN, Haghbayan MH, Heikkonen J, Heikkonen H, Plosila J (2019) Formation maintenance and collision avoidance in a swarm of drones. *Proceedings of the 3rd International Symposium on Computer Science and Intelligent Control, ISCSIC 2019*. Association for Computing Machinery, New York. <https://doi.org/10.1145/3386164.3386176>
11. Khan M, Hassan S, Ahmed SI, Iqbal J (2017) Stereovision-based real-time obstacle detection scheme for unmanned ground vehicle with steering wheel drive mechanism. In: 2017 International Conference on Communication, Computing and Digital Systems (C-CODE), pp. 380–385
12. Alejo D, Cobano JA, Heredia G, Ollero A (2015) Collision-free trajectory planning based on maneuver selection-particle swarm optimization. In: 2015 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 72–81
13. Yasin JN, Mohamed S, Haghbayan MH, Heikkonen J, Tenhunen H, Plosila J (2020) Navigation of autonomous swarm of drones using translational coordinates. *Advances on practical applications of agents and multi-agent Systems*. Springer, Berlin
14. Seo J, Kim Y, Kim S, Tsourdos A (2017) Collision avoidance strategies for unmanned aerial vehicles in formation flight. *IEEE Trans Aerosp Electron Syst* 53(6):2718–2734
15. Lin Y, Saripalli S (2015) Collision avoidance for uavs using reachable sets. In: 2015 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 226–235
16. Kwag YK, Chung CH (2007) Uav based collision avoidance radar sensor. In: 2007 IEEE International Geoscience and Remote Sensing Symposium, pp. 639–642
17. Hrabar S (2011) Reactive obstacle avoidance for rotorcraft uavs. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4967–4974
18. Hasan KM, Al-Nahid A, Reza KJ, Khatun S, Basar MR (2013) Sensor based autonomous color line follower robot with obstacle avoidance. In: 2013 IEEE Business Engineering and Industrial Applications Colloquium (BEIAC), pp. 598–603
19. Kanarachos SA (2009) A new method for computing optimal obstacle avoidance steering manoeuvres of vehicles. *Int J Veh Auton Syst* 7(1–2):73–95. <https://doi.org/10.1504/IJVAS.2009.027968>
20. Zohaib M, Pasha M, Riaz RA, Javaid N, Ilahi M, Khan RD (2013) Control strategies for mobile robot with obstacle avoidance. *J Basic Appl Sci Res* 3(4):1027–1036

21. Oroko J, Nyakoe G (2014) Obstacle avoidance and path planning schemes for autonomous navigation of a mobile robot: a review. In: Proceedings of Sustainable Research and Innovation Conference, pp. 314–318
22. Olivares-Mendez MA, Campoy P, Mellado-Bataller I, Mejias L (2012) See-and-avoid quadcopter using fuzzy control optimized by cross-entropy. In: 2012 IEEE International Conference on Fuzzy Systems, pp. 1–7
23. Wang H, Wei Z, Wang S, Ow CS, Ho KT, Feng B (2011) A vision-based obstacle detection system for unmanned surface vehicle. In: 2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM), pp. 364–369
24. US Department of Commerce NOAA (2020) Lidar, national ocean service. <https://oceanservice.noaa.gov/facts/lidar.html>. Library Catalog: oceanservice.noaa.gov. Accessed 29 May 2020
25. Bareiss D, van den Berg J, Leang KK (2015) Stochastic automatic collision avoidance for tele-operated unmanned aerial vehicles. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4818–4825
26. Sharma SU, Shah DJ (2017) A practical animal detection and collision avoidance system using computer vision technique. IEEE Access 5:347–358
27. De Simone MC, Rivera ZB, Guida D (2018) Obstacle avoidance system for unmanned ground vehicles by using ultrasonic sensors. Machines. <https://doi.org/10.3390/machines6020018>
28. Filming of space robot “jitter” assembled out of legos (2020). http://www.nasa.gov/mission_pages/station/research/experiments/468.html#publications. Accessed 21 Apr 2020
29. leJOS (2020). <https://en.wikipedia.org/w/index.php?title=LeJOS&oldid=956137626>. Page Version ID: 956137626. Accessed 21 Apr 2020
30. Meister O, Frietsch N, Ascher C, Trommer GF (2008) Adaptive path planning for a vtol-uav. In: 2008 IEEE/ION position, location and navigation symposium, pp 1252–1259
31. Moses A, Rutherford MJ, Valavanis KP (2011) Radar-based detection and identification for miniature air vehicles. In: 2011 IEEE international conference on control applications (CCA), pp 933–940
32. Yasin JN (2017) Collision avoidance algorithm for unmanned aerial or ground vehicles. Master’s Thesis, bo Akademi University
33. Trigonometrical ratios in a right-angled triangle (2020) <http://www.mathcentre.ac.uk/resources/uploaded/mc-ty-trigratios-2009-1.pdf>. Accessed 21 Apr 2020