Multi-sensor fusion with radar and ultrasound for obstacle avoidance on Capra Hircus 1.0

Benjamin Damsgaard, Simon S. Gaasdal, Søren Bonnerup

Department of Materials and Production, Aalborg University
Fibigerstraede 16, DK-9220 Aalborg East, Denmark
Email: {bdamsg19, sgaasd19, sbonne19}@student.aau.dk,
Web page: http://www.mechman.mp.aau.dk/

Abstract
A key feature in mobile robotics is obstacle avoidance and perception of the environment. As such, this project creates a sensor fusion algorithm with multi-sensor input, in order to perform obstacle avoidance. The project is made in collaboration with Capra Robotics ApS and will be based on their Hircus 1.0 platform. The platform has two integrated ultrasonic sensors, and a RadarIQ-M1 will be installed for the purpose of this project. The sensor fusion will be made through a particle filter, which is a probabilistic method capable of detecting multiple objects. To perform obstacle avoidance, an algorithm based on the PointBug will evaluate the sensor fused data, and move the robot through unknown environments with both static and dynamic obstacles.
To test the sensor fusion, comparisons will be made of the output from the individual sensors and output from the sensor fusion. From this, a qualitative assessment will be made. The obstacle avoidance algorithm, will be tested in various settings and finally be compared to the obstacle avoidance of the MiR100.

Keywords: Sensor fusion, path planning, obstacle avoidance, radar, ultrasound, outdoor mobile robot

1. Introduction

Obstacle avoidance is a fundamental requirement for mobile robotic systems to operate safely and efficiently in dynamic environments. The challenge lies in the uncertainty of the movements of dynamic obstacles, which makes it difficult for navigation systems to compute collision-free paths [1]. To tackle this issue, a common approach in mobile robotics is to rely on sensors such as laser radar, Light Detection and Ranging (LiDAR), and camera-based sensors. However, these sensors have their limitations, particularly in extreme outdoor weather conditions and heavy occlusion.

Radar sensors offer an advantage in detecting obstacles in inclement weather conditions, but often produce a sparse interpretation of its environment [2]. Ultrasonic sensors, have the capability of detecting static and are widely used for obstacle detection in mobile robotics due to their low cost and high detectability for different textures [3].

To provide accurate and reliable obstacle detection and avoidance in various outdoor environments, a novel approach has been proposed to fuse ultrasound and radar data for mobile robotic systems. The aim is to overcome the limitations of individual sensors and to create a feasible concept for full implementation. This approach is being developed in collaboration with Capra Robotics ApS, using their mobile robotic platform, Capra Hircus 1.0 which can be seen in Figure 1. The platform has two integrated ultrasonic sensors and for the purpose of this project, a RadarIQ-M1 has been installed.

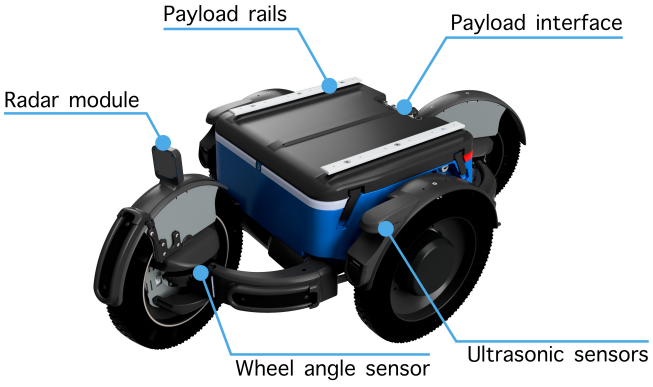

Fig. 1 Render of the front left of the Capra Hircus 1.0, with some of the key features highlighted.

2. Related work

A thorough review of related work and research is critical for developing an effective obstacle avoidance sys-

tem for mobile robots that utilises ultrasonic sensor and radar technology. As such, the following section will introduce key studies in the field of obstacle avoidance based on sensor fusion. To achieve a comprehensive review, each part of the field will be described individually in the following sections, i.e. obstacle avoidance and sensor fusion, and at last be summarised and concluded upon.

In the field of local navigation, obstacle avoidance is done in cooperation between multiple aspects. A fundamental part of obstacle avoidance is the underlying algorithm responsible for decision making. Depending on the algorithm, decisions can vary in complexity and abstractness. A decision derived from an obstacle avoidance algorithm can both be represented as a trajectory in a complex system, or it can be represented as a right/left command in a simple system [4]. With offset in reviews of different approaches to obstacle avoidance [5], [6], methods and theories in this field will be investigated. These methods include fuzzy logic controllers [7], [8], bug algorithms [9], [10], [11], [12], trajectory and map based [5], [6], [13], [14], [15] and finally moving obstacles [16], [17], [18], [8].

According to the results of current research in the area of algorithms for obstacle avoidance, a range of methods with different and unrelated bases have proved effective in avoiding obstacles of various complexity. Some approaches have gained even better results by fusing multiple algorithms, exploiting only the best of each algorithm [6]. It is also possible to conclude that the most optimal choice of obstacle avoidance algorithms largely depends on the application, as the criteria for sensor accuracy and the computational load vary greatly among different methods.

Sensors have a great influence on the obstacle avoidance algorithms as the decisions are based on a representation of the input data delivered by the sensors. As such, the following investigates how sensor fusion has been performed previously, to obtain a more thorough understanding about how the sensors used in this project should be fused.

This project focus on sensor fusion of imperfect data from an ultrasonic sensor and a radar on the Hircus platform. As such, only papers using sensor fusion of imperfect data has been investigated. Specifically, fuzzy sensor fusion [19], [7], [3] and probabilistic methods including variations of Kalman filtering [20], [21], [22], [23], and particle filtering [24], [25], [26], [27], [28], [29] , [30], [31], [32] have been investigated.

From the studies investigated, it is observed that Kalman filters, particle filters and fuzzy sensor fusion can all be applied in mobile robotics. However, they are often used in different settings. For example, it is observed that the Kalman filter and fuzzy sensor fusion is applied in a variety of tasks, such as single object detection and target tracking, localisation, pose estimation and range estimation [20], [21], [22], [23], [7], [3]. Where particle filters are primarily used in localisation tasks [28], [29], [27] and multi-object detection [25], [26].

Based on the investigated studies and the sensor data available with two ultrasonic sensors and a RadarIQ-M1, it has been determined that particle filtering can provide the best estimation of the environment. The obstacle avoidance algorithm will be based on principles from the PointBug algorithm, which is one of the most recently developed bug algorithms. This algorithm has shown to be easy to implement and have the necessary capabilities for further development and expansion of the algorithm.

3. Methodology

As the Hircus platform is currently capable of driving between Global Positioning System (GPS) coordinates in a straight line, the aim of this project is to add a local planner to the existing global planner. The local planner will start whenever obstacles are detected within $1.9\,\mathrm{m}$ of the robot. If no obstacles are detected, the robot will only rely on the existing global planner.

3.1 Sensor fusion

As mentioned in section 2, the sensor fusion method which will be used in this project is a particle filter using the input from the two integrated ultrasonic sensors and the RadarIQ-M1. The data obtained from the ultrasonic sensors is range measurements which only state the distance to the nearest obstacle in the sensors' Field Of View (FOV). The data from the radar is a sparse point cloud, transformed according to the position and angle of the front wheel which the radar is mounted on.

The algorithm has been divided into the 4 different steps of particle filtering. These steps include: Initialisation, prediction, update and resampling. In the initialisation step, the particles of the filter are initiated with an $[x, y]$ position and a weight. The position of the particles must be within the FOV of at least one of the three sensors. In the prediction step, the particles are applied with some variance in both the x and y direction. Furthermore,

$33\,\%$ of the particles are randomly distributed again to ensure objects are detected when they appear in the FOV. In the update step, the weight of each particle is updated based on how close they are to the ranges of the ultrasonic sensors and how close they are to each point of the radar point cloud. The likelihood of belonging to each range and point is calculated based on the equation:

$$f(\Delta, \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\Delta-\mu}{\sigma}\right)^2} \tag{1}$$

Where Δ is the euclidean distance between the particle and point or range, μ is the mean, which has been set equal to zero as the difference has already been calculated, and σ is the standard deviation. All likelihoods of one particle are then multiplied to calculate the weight.

In the resample step, the particles are resampled with a probability proportional to the weight of each particle. Furthermore, by using K-nearest neighbour search method, the algorithm has been limited to sample maximum K points within a given area. The search point of the K-nearest search will be the particle with the highest weight and the neighbouring particles are stored in a temporary vector. Then, K particles are resampled proportional to the weight of the particles stored in the temporary vector, i.e. some particles may be sampled multiple times and some might not be sampled at all. As the K amount of particles have been resampled, the particles stored within the temporary vector are removed from the original vector of particles and the process is repeated. When finding the next search point for the K-nearest neighbour search, a minimum distance between the new search point and previous search points must be fulfilled and the new search point particle must have a weight above a certain threshold. If there are no particles left with a weight above the threshold and a minimum distance to previous search points, then particles are randomly sampled until the desired amount of particles have been resampled. The distance between search points and K are parameters which have been tested with different values to optimise the performance of the algorithm.

Once the desired amount of particles have been resampled, the prediction, update and resampling steps are repeated continuously. Furthermore, after each resampling step, clusters are determined in order to produce a point cloud of obstacles for the point bug algorithm. This is done by iterating through all particles and comparing distances to the nearby points by using a K-nearest search method, similarly to the resampling step. A cluster is defined as an obstacle when it has more than a certain amount of points located within a circle with a specific radius. The amount of points and radius of these clusters are parameters which have been tested with different values together with the distance between search points and K from the resampling step. Furthermore, the filter has been tested with different amounts of particles as well.

During parameter tuning, the best results were obtained with 800 particles, cluster size of 70 points, cluster radius of $5\,\mathrm{cm}$, 60 nearest particles and a minimum distance between search points of $12\,\mathrm{cm}$ in the resampling step.

3.2 Obstacle avoidance

The algorithm for obstacle avoidance is initialised with parameters specifying the physical properties of the robot and the desired behaviour of the robot. These specifications will be fixed for the entire implementation and will not change during the execution of the obstacle avoidance. After the initialisation, the point cloud obtained in subsection 3.1 will be transformed to follow the frame of the robot, and not only the angle of the front wheel. This allows a rolling map to be created. However, as the heading of the robot is only accessable for internal use, the rolling map will only include points captured within $1.5\,\mathrm{s}$. This timestamp have shown to perform the best.
Based on the distance between the robot and the content of the point cloud, i.e. obstacles, it will be determined if obstacle avoidance should be executed. If an obstacle is too close to the robot, it should be avoided by following the underlying algorithm of PointBug [9].

Using this algorithm, a target vector from the robot, pointing in the direction of the next GPS goal coordinate should be calculated, along with the heading of the robot. However, it is currently not possible to obtain the coordinates of the goal through the Capra Hircus, hence the target vector is calculated as the relative heading of the robot when an obstacle gets too close. The relative heading is represented as a direction in the point cloud, as the true heading of the robot is reserved for internal use by Capra Robotics.

The next step is then to determine every sudden point in the point cloud. This is done by clustering nearby points, that meet the criteria of a sudden point. A vector to each sudden point is then calculated and compared to the target vector. The vector closest to the target vector will determine which sudden point the robot

will move towards. As the target vector is infinite, due to the lack of GPS coordinates of the goal target, the length of the vectors will not be evaluated, but only the angle will be used to determine sudden points.

In order to overwrite the global planner, the obstacle avoidance algorithm calculates direct velocities. The direct velocities sent for avoiding the obstacles are based on the angle and distance to the sudden point and not a path towards the sudden point. This modification from the original proposal for the PointBug algorithm is necessary as neither the geographical heading or geographical target is possible to obtain from the current Capra Hircus software. However, with only minor changes, the algorithm could run with these parameters if the internal topics of the robot could be accessed.

4. Experiments and results

In order to test the performance of the sensor fusion and obstacle avoidance, the solution has been tested in various settings. The different settings and the measured parameters are based on earlier work done in the field of obstacle avoidance for mobile robotics [8][33][14].

4.1 Test A: Fusion

First, the sensor fusion has been tested by comparing the raw sensor input with the point cloud obtained through sensor fusion when the robotic platform is not moving. The test is performed with 3 different settings, where the wheel angle and amount of objects is changed. In setting 1, the wheel angle is $0°$, an obstacle is placed in the FOV of each of the ultrasonic sensors and the radar can see both obstacles. In setting 2, the wheel angle is $60°$, an obstacle is placed in the overlapping FOV of the ultrasonic sensors and an obstacle is placed in the radar FOV. In setting 3, the wheel angle is $60°$ and only one obstacle is placed in the FOV of the right ultrasonic sensor. The data from each of the sensors and the fused point cloud is measured 100 times. The setup and results from the three different settings can be seen in Figure 2, 3 and 4, where the the measurements from the radar, ultrasonic sensors and fused point cloud can be seen.

From setting 1, the mean error for the radar, left ultrasonic sensor, right ultrasonic sensor and fused output is $0.133\,\mathrm{m}$, $0.036\,\mathrm{m}$, $0.074\,\mathrm{m}$ and $0.125\,\mathrm{m}$, respectively. Even though the ultrasonic sensors have a lower mean error, they are still more inaccurate as the error is calculated based on the provided range and not a specific position. The results from setting 1 can be seen in Figure 2.

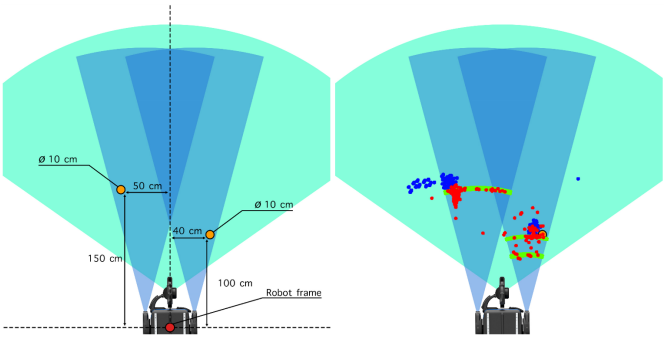

Fig. 2 Setting 1: Setup to the left with ground truth of obstacle marked orange. Results to the right with radar point cloud marked blue, ultrasonic measurements marked green and fused point cloud marked red.

From setting 2, the mean error for the radar, left ultrasonic sensor, right ultrasonic sensor and fused output is $0.094\,\mathrm{m}$, $0.019\,\mathrm{m}$, $0.053\,\mathrm{m}$ and $0.145\,\mathrm{m}$, respectively. From these results, it is observed that the sensors individually provide a more accurate result. However, they only describe one obstacle each, while the fusion point cloud is capable of describing both obstacles. Furthermore, the fusion point cloud gives a good estimation of the position of the obstacle detected by the ultrasonic sensors, while they only provide a range. The results from setting 2 can be seen in Figure 3.

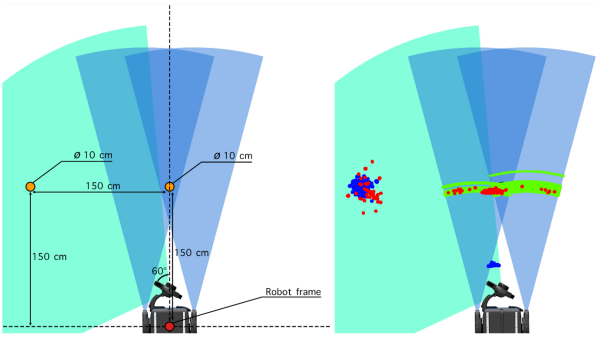

Fig. 3 Setting 2: Setup to the left with ground truth of obstacle marked orange. Results to the right with radar point cloud marked blue, ultrasonic measurements marked green and fused point cloud marked red.

From setting 3, the mean error for the right ultrasonic sensor and fused output is $0.016\,\mathrm{m}$ and $0.159\,\mathrm{m}$, respectively. The results from setting 3 can be seen in Figure 4.

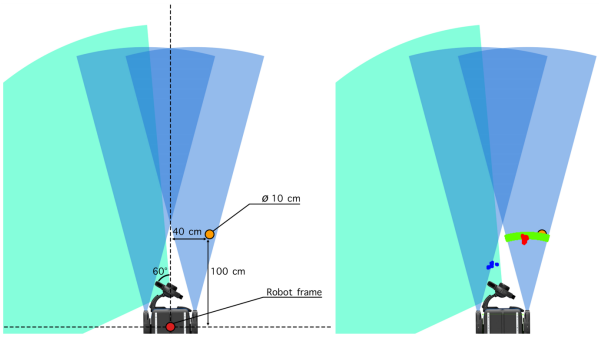

Fig. 4 Setting 3: Setup to the left with ground truth of obstacle marked orange. Results to the right with radar point cloud marked blue, ultrasonic measurements marked green and fused point cloud marked red.

From the three different settings, it can be concluded that the sensor fusion provides a more reliable point cloud than using the three sensors individually, especially when turning, as it is able to provide input from the different sensors with higher accuracy in a single point cloud than what is possible with the individual sensors. The mean errors for the three sensors and the fused point cloud can be seen in Table I.

Tab. I Mean error based on 100 measurements from the three sensors and the fused output, in all three settings.

Setting	Radar	Ultrasonic sensor left	Ultrasonic sensor right	Fused output
Setting 1	0.133 m	0.036 m	0.074 m	0.125 m
Setting 2	0.094 m	0.019 m	0.053 m	0.145 m
Setting 3			0.016 m	0.159 m

4.2 Test B: Simple static obstacle avoidance

With a standardised test environment, this test is designed to force a robot to do obstacle avoidance in order to reach a specified goal. At the same time, each obstacle can be traversed on a side that provides the shortest path and a side that provides a longer path. The test is specified and executed as illustrated in Figure 5.

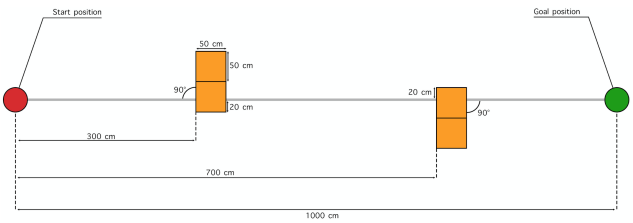

Fig. 5 Illustration of the test setup for Test B.

After running the test 10 times for both the Hircus and MiR100, the paths were mapped as seen in Figure 6 and the parameters were calculated as seen in Table II.

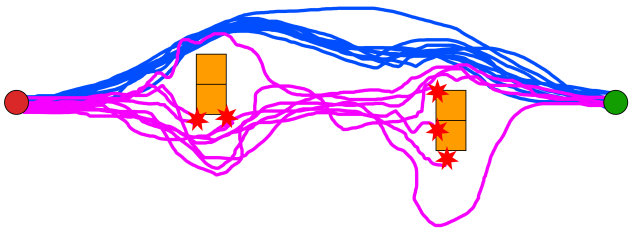

Fig. 6 The recorded path of the Hircus (purple) and MiR100 (blue), with collisions marked red.

From the traversed paths and the measured parameters, it can be concluded that the Hircus have multiple collisions. Only one of the collisions do not happen at a corner of the obstacle. Looking further into this run reveals that the robot do not converge on which side the obstacle should be traversed. The collisions at the corners could potentially could be neglected by further tuning of the algorithm. Comparing the successful runs of the Hircus with the MiR100 shows that the Hircus spent less time avoiding the obstacles than the MiR100, but it travelled a greater distance. This is because the Hircus do not change lower the speed as much as the MiR100, while avoiding obstacles.

Tab. II Average deviations from the test. The "Time deviation" and "Distance deviation" are averaged from the successful runs. The "Successful runs" shows how many of the 10 runs that were not terminated by collision.

	Time deviation	**Distance deviation**	**Successful runs**
MiR100	38 %	7 %	100 %
Hircus	33 %	33 %	40 %

4.3 Test C: Obstacle of complex geometry

This test environment is identical to the test presented in subsection 4.2, with the only exception being the geometry of the obstacles. In this test the two robots will encounter at least 4 corners for every obstacle. This increase in obstacle complexity provides more information to base the decisions on, and sets greater demands for the sensor representation of the surroundings. This environment is illustrated in Figure 7

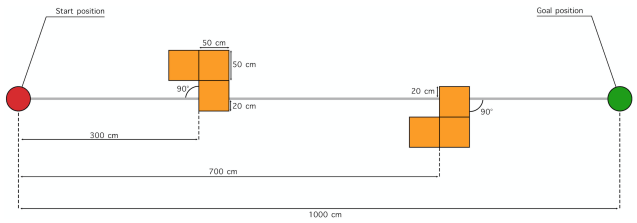

Fig. 7 Illustration of the test setup for Test C.

After running the test 10 times for both the Hircus and

MiR100, the paths were mapped as seen in Figure 8 and the parameters were calculated as seen in Table III.

Fig. 8 The recorded path of the Hircus (purple) and MiR100 (blue), with collisions marked red.

From the traversed paths and the measured parameters, it can be concluded that the Hircus has less collisions in this environment. The collisions only happen at corners in this test, which potentially could be neglected by further tuning of the algorithm.

Tab. III Average deviations from the test. The "Time deviation" and "Distance deviation" are averaged from the successful runs. The "Successful runs" shows how many of the 10 runs that were not terminated by collision.

	Time deviation	Distance deviation	Successful runs
MiR100	38 %	9 %	100 %
Hircus	25 %	24 %	70 %

4.4 Test D: Dynamic obstacle avoidance

In this test, only one obstacle is present. This obstacle is however dynamic. The motion of the obstacle is determined upon the speed of the robot, to ensure collision if the robot do not react upon the presence of the obstacle. This is further specified in Figure 9, where it can be seen that the obstacle will move in front of the robot, when it is 1.0 m away from where the centerline passes the path of the obstacle.

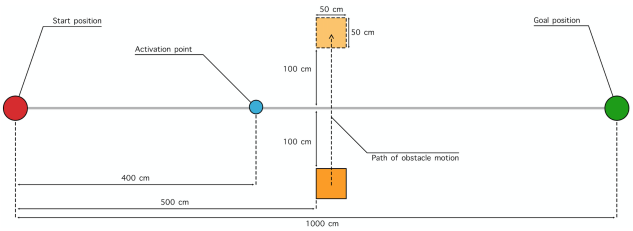

Fig. 9 Illustration of the test setup for Test C.

After running the test 10 times for both the Hircus and MiR100, the paths were mapped as seen in Figure 8 and the parameters were calculated as seen in Table III.

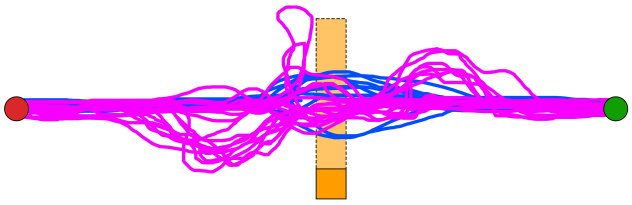

Fig. 10 The recorded path of the Hircus (purple) and MiR100 (blue).

From the traversed paths and the measured parameters, it can be concluded that the global planner of the Hircus makes the robot traverse in a loop when the heading of robot is more than 90° in positive direction of rotation. However, no collision happens, and the deviation in time is lower than the MiR100. These results are seen in Table IV.

Tab. IV Average deviations from the test. The "Time deviation" and "Distance deviation" are averaged from the successful runs. The "Successful runs" shows how many of the 10 runs that were not terminated by collision.

	Time deviation	Distance deviation	Successful runs
MiR100	33 %	3 %	100 %
Hircus	11 %	11 %	100 %

5. Conclusion

This project aimed to investigate obstacle avoidance on the Capra Hircus platform using fused sensor data and the PointBug algorithm. This was done by implementing an algorithm for fusing the ultrasonic sensor data and radar data by utilising a particle filter. The particle filtered data showed to represent obstacles better than only using the raw radar or ultrasonic sensor data. However, during testing of the entire system, it was found that the robot would collide with corners of the obstacles in some cases, and other times choose unnecessary long paths. This was found to happen because of two main shortcomings. Firstly, the lack of a robot heading made it difficult to update the global goal when encountering a new obstacle. Secondly, collisions with obstacle corners occurred because of the sparse implementation of a rolling map, which again was a result of the missing heading. Finally, the software and hardware were evaluated for integration into Capra Robotics' production. This showed that minimal changes were needed for the hardware integration and potential algorithm adjustments were needed for a full software integration.

Acknowledgement
The authors of this work gratefully acknowledge Grundfos for sponsoring the 11th MechMan symposium.

References

[1] B. Guo, N. Guo, and Z. Cen, "Obstacle avoidance with dynamic avoidance risk region for mobile robots in dynamic environments," IEEE Robotics and Automation Letters, vol. 7, no. 3, pp. 5850–5857, 2022.

[2] H. Andreasson, G. Grisetti, T. Stoyanov, and A. Pretto, "Sensors for mobile robots," arXiv preprint arXiv:2206.03223, 2022.

[3] S. Adarsh and K. Ramachandran, "Neuro-fuzzy based fusion of lidar and ultrasonic sensors to minimize error in range estimation for the navigation of mobile robots," Intelligent Decision Technologies, vol. 14, no. 2, pp. 259–267, 2020.

[4] H. C. et al., Principles of Robot Motion. 5 Cambridge Center, Cambridge, England: Massachusetts Institute of Technology, 1. ed., 2005.

[5] B. P. et al., "A review: On path planning strategies for navigation of mobile robot," Defence Technology, vol. 15, pp. 582–606, 2019.

[6] H. ye Zhang et al., "Path planning for the mobile robot: A review," Symmetry, vol. 10, no. 450, 2018.

[7] A. D. Adhvaryu, S. Adarsh, and K. Ramchandran, "Design of fuzzy based intelligent controller for autonomous mobile robot navigation," in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 841–846, 2017.

[8] Z. L. et al., "Path planning of mobile robot with pso-based apf and fuzzy-based dwa subject to moving obstacles," Transactions of the Institute of Measurement and Control, vol. 44, no. 1, pp. 121–132, 2022.

[9] N. B. et al., "A simple local path planning algorithm for autonomous mobile robots," International Journal of Systems Applications, vol. 5, no. 2, pp. 151–159, 2011.

[10] V. Lumelsky and T. Skewis, "Incorporating range sensing in the robot navigation function," IEEE Transactions On Systems, vol. 20, no. 5, pp. 1058–1069, 1990.

[11] I. Kamon and E. Rivlin, "Sensory-based motion planning with global proofs," IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, vol. 13, no. 6, pp. 814–822, 1997.

[12] I. K. et al., "Tangentbug: A range-sensor-based navigation algorithm," The International Journal of Robotics Research, vol. 17, no. 9, pp. 907–1021, 1998.

[13] O. M. et al., "Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles," Expert Systems with Applications, vol. 42, no. 12, pp. 5177–5191, 2015.

[14] F. P. et al., "A comparison of local path planning techniques of autonomous surface vehicles for monitoring applications: The ypacarai lake case-study," Sensors, vol. 20, no. 1488, 2020.

[15] K. C. et al., "Thorough robot navigation based on svm local planning," Robotics and Autonomous Systems, vol. 70, pp. 166–180, 2015.

[16] F. K. et al., "A review on motion planning and obstacle avoidance approaches in dynamic environments," Advances in Robotics & Automation, vol. 4, no. 2, 2015.

[17] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," Robotics and Autonomous Systems, vol. 100, pp. 171–185, 2018.

[18] J. H.-A. et al., "Application of time dependent probabilistic collision state checkers in highly dynamic environments," PLoS ONE, vol. 10, no. 3, 2015.

[19] A. Shitsukane, W. Cheruiyot, C. Otieno, and M. Mvurya, "Fuzzy logic sensor fusion for obstacle avoidance mobile robot," in 2018 IST-Africa Week Conference (IST-Africa), pp. Page–1, IEEE, 2018.

[20] W. Farag, "Kalman-filter-based sensor fusion applied to road-objects detection and tracking for autonomous vehicles," Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, vol. 235, no. 7, pp. 1125–1138, 2021.

[21] G. Wan, X. Yang, R. Cai, H. Li, Y. Zhou, H. Wang, and S. Song, "Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes," in 2018 IEEE international conference on robotics and automation (ICRA), pp. 4670–4677, IEEE, 2018.

[22] Á. Odry, R. Fullér, I. J. Rudas, and P. Odry, "Kalman filter for mobile-robot attitude estimation: Novel optimized and adaptive

solutions," Mechanical systems and signal processing, vol. 110, pp. 569–589, 2018.
[23] M. B. Alatise and G. P. Hancke, "Pose estimation of a mobile robot based on fusion of imu data and vision data using an extended kalman filter," Sensors, vol. 17, no. 10, p. 2164, 2017.
[24] T. N. N. Hossein, S. Mita, and H. Long, "Multi-sensor data fusion for autonomous vehicle navigation through adaptive particle filter," in 2010 IEEE Intelligent Vehicles Symposium, pp. 752–759, IEEE, 2010.
[25] B. Wang, S. A. R. Florez, and V. Frémont, "Multiple obstacle detection and tracking using stereo vision: application and analysis," in 2014 13th international conference on control automation robotics & vision (ICARCV), pp. 1074–1079, IEEE, 2014.
[26] N. Long, K. Wang, R. Cheng, W. Hu, and K. Yang, "Unifying obstacle detection, recognition, and fusion based on millimeter wave radar and rgb-depth sensors for the visually impaired," Review of scientific instruments., vol. 90, no. 4, 2019-04.
[27] W. Hu, K. Wang, and H. Chen, "A robust localization approach using multi-sensor fusion," in Emerging Imaging and Sensing Technologies for Security and Defence III; and Unmanned Sensors, Systems, and Countermeasures, vol. 10799, pp. 195–201, SPIE, 2018.
[28] J. K. Suhr, J. Jang, D. Min, and H. G. Jung, "Sensor fusion-based low-cost vehicle localization system for complex urban environments," IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 5, pp. 1078–1086, 2017.
[29] D. Kang and D. Kum, "Camera and radar sensor fusion for robust vehicle localization via vehicle part localization," IEEE Access, vol. 8, pp. 75223–75236, 2020.
[30] T. Grami and A. S. Tlili, "Indoor mobile robot localization based on a particle filter approach," in 2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), pp. 47–52, IEEE, 2019.
[31] R. P. Guan, B. Ristic, L. Wang, and R. Evans, "Monte carlo localisation of a mobile robot using a doppler–azimuth radar," Automatica, vol. 97, pp. 161–166, 2018.
[32] Q. bin Zhang, P. Wang, and Z. hai Chen, "An improved particle filter for mobile robot localization based on particle swarm optimization," Expert Systems with Applications, vol. 135, pp. 181–193, 2019.
[33] L. K. et al., "Arena-bench: A benchmarking suite for obstacle avoidance approaches in highly dynamic environments," IEEE ROBOTICS AND AUTOMATION LETTERS, vol. 7, no. 4, pp. 9477–9484, 2022.