Complex & Intelligent Systems (2022) 8:5065–5086
https://doi.org/10.1007/s40747-022-00755-0

ORIGINAL ARTICLE

End-to-end human inspired learning based system for dynamic obstacle avoidance

S. M. Haider Jafri[1] · **Rahul Kala**[1]

Received: 9 June 2021 / Accepted: 15 April 2022 / Published online: 3 May 2022

Abstract
As a first, the paper proposes modelling and learning of specific behaviors for dynamic obstacle avoidance in end-to-end motion planning. In the literature many end-to-end methods have been used in simulators to drive a car and to apply the learnt strategies to avoid the obstacles using the lane changing, following the vehicle as per the traffic rules, driving in-between the lane boundaries, and many more behaviors. The proposed method is designed to avoid obstacles in the scenarios where a dynamic obstacle is headed directly towards the robot from different directions. To avoid the critical encounter of the dynamic obstacles, we trained a novel deep neural network (DNN) with two specific behavioral obstacle avoidance strategies, namely *"head-on collision avoidance"* and *"stop and move"*. These two strategies of obstacle avoidance come from the human behavior of obstacle avoidance. Looking at the current frame only, for a very similar visual display of the scenario, the two strategies have contrasting outputs and overall outcomes that makes learning very difficult. A random data recording over general simulations is unlikely to record the corner cases of both behaviors that rarely occur, and a behavior-specific training used in this paper intensifies the same cases for a better learning of the robot in such corner cases. We calculate the intention of the obstacle, whether it will move or not. This proposed method is compared with three state-of-the-art methods of motion planning, namely Timed-Elastic Band, Dynamic Window Approach and Nonlinear Probabilistic Velocity Obstacle. The proposed method beats all the state-of-the-art methods used for comparisons.

Keywords Motion planning · Deep learning · Dynamic obstacle

Introduction

While humans move in the environment or drive a car or bicycle, they observe the environment and identify the dynamic obstacles and static obstacles and infer the intent of the dynamic obstacles without having any communication. This information is extracted by humans with a glimpse of the environment using which the humans plan their behavior or control the car. This is a challenging problem in robotics. To imitate the human behavior, we closely observe the obstacle avoidance strategy of humans.

This paper addresses the problem of collision avoidance of dynamic and static obstacles closely inspired from the behavior of humans. The dynamic collision avoidance of the humans is a combination of two most common behaviors, *head-on collision avoidance* and *"stop and move"*. The head-on collision situation comes when a person moving towards a goal encounters an obstacle coming from the opposite direction directly towards the person. In this situation, the person both takes a left or right turn to avoid a collision and thereafter instantly aligns to the goal. In [1], the head-on collision avoidance behavior was solved by proposing a reciprocal velocity obstacle method, where the agents take right on coming closer to each other. Since the authors primarily aim to minimize the path length, the agents minimally deviate their paths while maintaining smaller clearances. In contrast, the proposed method is anticipatory where clarity of intentions is the main aim and therefore the agents balance between minimizing path length, maximizing clearance, and demonstrating a clear strategy to avoid each other without communication.

In this study, we believe that the person generally prefers taking a left turn instead of a right turn shown in Fig. 1 as

✉ S. M. Haider Jafri
sayedhaiderjafri6@gmail.com

Rahul Kala
rkala001@gmail.com

[1] Centre of Intelligent Robotics, Indian Institute of Information Technology, Allahabad, India

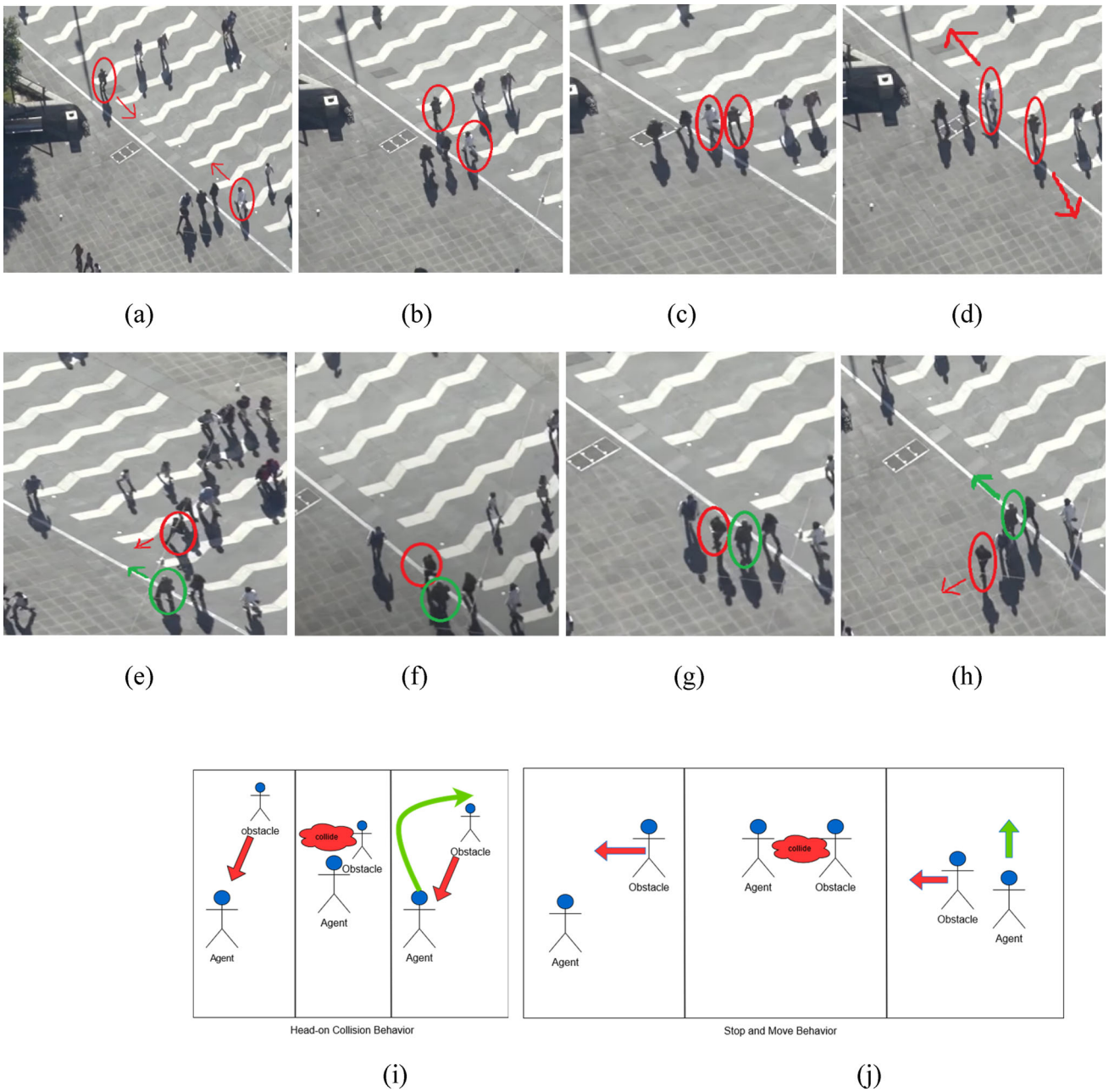


Fig. 1 Humans avoiding obstacles, Head-on collision [3]**:** (**a–d**) Person taking left to avoid the obstacle coming in the direction of motion marked by a red colored line, and again aligning in the direction to goal after avoiding, Stop and move: (**e**–**h**) Person (green) stops to let the obstacle (red) pass by, and starts moving after the obstacle has passed from the front. Schematic representation: (**i**) head-on collision (**j**) stop and move

per the social conventions in the authors' country. As per the social conventions in some other places, a right turn may be preferred. Figure 1a–d shows a head-on collision avoidance behavior, where both persons marked with the red circle try to avoid each other. The situation corresponding to the *"stop and move"* behavior comes when an obstacle crosses the path of a person, and the person slows (or stops in an extreme case) to give a pass to the obstacle as shown in Fig. 1e–h. The figures show the behavior where the person in green circle stops to let the person in red circle pass. The stop and move behavior is a natural behavior of humans while driving a car and while crossing the roads. The autonomous vehicles follow the same pattern of behavior while detecting traffic light or detecting intersections. In [2] the authors build a deep learning-based solution to drive a vehicle in different

scenarios, while trying to learn the affordance of the traffic state.

The dynamic obstacles are capable of decision making without communication. In a decentralized system of navigation, it is hard for the participating agents to get into a consensus on how to best avoid each other, as each agent's trajectory depends upon the others. How the humans can come to a consensus without any communication in a decentralized manner is still a mystery for the researchers. However, it is important to estimate the intent of the dynamic obstacle to devise a strategy to avoid it.

The imitation of obstacle avoidance strategy is not an easy task. The nature of a human's obstacle avoidance at a local level is reactive, while the deliberative aspects are largely limited to deciding the route around the static obstacles. The obstacle avoidance strategy is learnt from the experiences by the humans. The humans learn many tasks through their life experiences. First, they start walking, then they start running, and last but not the least they start driving. A similar concept applied in this paper. First, we make a navigation function to learn the static obstacle avoidance. Secondly, we learn the *head-on collision* behavior. At last, a *"stop and move"* behavior is learnt. The success of end-to-end learning-based [4] approaches in the literature suggest that obstacle avoidance behaviors can be learnt and successfully transferred to the robot. In this paper, the raw information of the environment, the temporal information of the nearest dynamic obstacle and the goal information makes the input to the end-to-end DNN to predict the angular and linear velocity and makes the agent capable to instantly avoid the obstacles.

The data set is collected by the deployment of Pioneer 3AT mobile robot in the Gazebo simulator. We have created a situation similar to the obstacle avoidance strategy of the humans in the simulator. We manually move the agent (Pioneer 3AT) using a joystick and avoid the dynamic obstacle using a head-on collision behavior and "stop and wait" behavior. The static obstacle is also avoided with a preference for the left turn. The data set is collected for 12 h and 0.25 million time-steps. The training of the end-to-end model from this dataset was difficult. We created a DNN model using a 1D CNN and LSTM with three inputs at the input layer. The model was successfully trained for the obstacle avoidance strategies. The major achievement of this model is that the behavior is a combination of three behaviors and while deployment of the agent in different scenarios (other than the trained), the agent behaves differently (other than the trained behavior). It can explain that the model can handle the uncertainty (situation other than training) by reacting differently.

In the area of learning-based motion planning, significant contributions are coming in multi-agent reinforcement learning (RL) frameworks that eliminate the hard work for the data collection. Our work contributes only towards the offline learning and not online learning like RL. In multi-agent RL, the different agents learn to mutually avoid each other in different settings while maximizing a common reward function. We argue that the humans do not follow a policy that maximizes a pre-known reward function. The robots learnt under multi-agent RL settings can often expect the humans to act in certain ways and a contradictory avoidance by the humans can come as an unforeseen situation for the robots. Further, the robot acting contradictory to what the humans anticipate can be uncomforting for the humans, significantly limiting the adoption of robots amidst human environments.

Similarly, the use of deliberative planning algorithms for the generation of a dataset for supervised learning is a commonly used technique. The trained algorithm in such a context carries the same limitations as that of the deliberative algorithm used for making a dataset. Again, the limitation of such approaches is that the humans may act contradictory to the simulated obstacles, a situation that the robots are not trained to handle. The robots are also not guaranteed to display the social conventions anticipated by the humans.

A typical implementation for learning the robot navigation in a supervised manner teleoperates the robots in general scenarios, sometimes using intelligent algorithms for the navigation of the obstacles. A significant portion of the human navigation consists of free-walking or following someone at front at a distant. The head-on collision avoidance and stop-and-wait are rare behaviors. A large dataset recording is likely to feature very small instances of head-on collision avoidance and stop-and-wait behaviors, that many models may even filter out as noise. The two behaviors are heuristically very important. Therefore, the proposed approach creates a dataset focusing on such rare behaviors in order to train the robot well on such corner cases that are often neglected in the literature.

The main contributions of this paper are as follows:

1. This paper proposes an end-to-end reactive motion planning algorithm to avoid the dynamic and static obstacles in the environment.
2. The end-to-end DNN architecture is trained using three different behavior-specific datasets to incorporate multiple behaviors in a single function of DNN.
3. We introduced intention detection in this method, which makes the agent aware of the intention of the obstacles close to the agent, including whether the obstacle is static or dynamic. The intension is from a raw sensor feed that removes the necessity of object level detection of the obstacles.
4. The method outperforms 3 state-of-the-art methods for the navigation of robots.

The structure of the paper is as follows: Sect. 1 gives the introduction of the proposed method and the collection of the dataset in the simulator. Section 2 discusses the three

aspects of motion planning (i) classical methods, (ii) hybrid methods, and iii) end-to-end supervised and reinforcement learning based approaches. In Sect. 3 we discuss the inputs and the architecture of the proposed method. In Sect. 4, the method is evaluated in static and dynamic environments and comparisons with the state-of-the-art methods are presented. The strengths and weaknesses of the method are discussed in Sect. 4.

Related works

Many algorithms have been proposed for dynamic and static environments. The deliberative algorithms work for static and partially dynamic environments. However, they consume a lot of energy and time to improvise the path for the static environment and the dynamic environment for situations where the trajectories of the other agents can be anticipated. The reactive approaches save the energy and time in decision making for moving the robot to the goal. The major drawback of this approach that there is no guarantee to reach the destination. The learning-based approaches are used for reactive decision making, enabling the agent to learn from the experiences. This section of the paper covers the different algorithms of motion planning from the classical approaches to the learning-based motion planning approaches to make the readers aware of the gaps of research that exist so far and how the present approach contributes to fill these gaps.

The most naïve algorithms of motion planning are Bug1, Bug2 [5] and Tangent bug [6] algorithms. These algorithms make the agent follow the boundary of the obstacle until they find an obstacle free orientation towards the goal. Another common class of algorithms is vector-based. The basic idea of these algorithms is negative and positive forces. The first assumption is that the obstacle in the environment generates a negative force, and the goal generates a positive force. Based on this assumption, the two algorithms implemented are VFF (*Vector Field Force*) [7] and PFs (*Potential Fields*) [8]. These algorithms are easy to implement and not much complex. They are highly parametric and complex to configure and make the robot stuck in a local minimum, not guaranteed to find the solution.

To navigate the robot in a structured environment, the Cartesian coordinate space is used. On the contrary, the histogram approaches represent the environment in the angular form. They find the angular orientation of the goal, obstacle and free space, and present in the form of a cost function. The VFH (*Vector Field Histogram*) [9] and its updated version VFH + [10] are the histogram-based approaches. These algorithms incorporate the uncertainty of the environment and also adhere to the kinematic constraints of the agent (non-holonomic robot). They also suffer from the local minima problem in concave (say, "U"-shaped) obstacles and do not allow the agent to move further towards the goal. In the velocity space approaches, the whole environment is represented in the form of velocity space instead of Cartesian space. The space includes the velocities which makes the agent move towards the destination without any collision, eliminating the velocities which are occluded by obstacles. The algorithm selects the best velocity at each iteration from the velocity space, which quickly moves the agent to destination. Some prevalent velocity-based methods in the motion planning of mobile robot are DWA (*Dynamic Window Approach*) [11] and CVM (*Curvature velocity Method*) [12]. The problem of local minima in these approaches cannot completely be eradicated and persists.

To overcome the problem of local minima, the *ND* [13] (*Nearness Diagram*) and *SND* [14] (*Smooth Nearness Diagram*) algorithms were proposed. These algorithms classify the environment for different scenarios and the strategy of obstacle avoidance and goal seeking used is different for different scenarios. The major issues of these kinds of algorithms are a highly complexity and a good perception of the environment is necessary to increase the accuracy of classification.

Another area of motion planning to tackle the problem of local minima is the combination of more than one technique. One technique finds the global goal of the agent and another technique finds the next step to reach the sub-goal. In [15], the authors used the optimal control theory to find the optimized path based on travelling time between every pair of targets. The authors used different clustering algorithms to solve the task assignment problem, assigning each vehicle to a cluster of targets to be optimally visited. Similarly, in [16], the optimal control theory was used to find the optimized cost between every target, while a co-evolutionary and multi-population Genetic algorithm was used for the task assignment problem. Both approaches used navigation in a drift field. The methods like LCM (*Lane Curvature Method*) [17] and BCM (*Beam Curvature Method*) [18] divide the environment to collect the goal and apply CVM to reach this goal. The other method DCVM, modified version of CVM is used to handle the dynamic obstacles. This method predicts the Cartesian coordinates of the dynamic obstacle and merges in the perception of the environment. This way the agent can plan their next movement and avoid the dynamic obstacle and anticipate the collision free manoeuvre. In this kind of algorithm, the risk of collision depends upon how accurately the position of the dynamic obstacles is predicted. The tracking of dynamic obstacles either uses the Kalman filter [19] or LSTM neural network [20]. NPVO (*Non-Linear Probabilistic Velocity Obstacle*) [21] uses the LSTM network to predict the motion of the dynamic obstacles and the probabilistic approach finds the velocity available to move the agent without any collision towards the goal. The DNN LSTM has the power to predict the temporal information

for a larger time period and it is only restricted to the availability of a large dataset. On the contrary, the Kalman filter calculates the time steps but does not need a large training dataset. These approaches assume that the complete information about the environment, including a knowledge of the positions and velocities of the dynamic agents is available. This can be time consuming, and the approaches are affected by the false positives and the false negatives of the detection and tracking algorithms, heavily limiting their use.

The reviewed methods are either deterministic or probabilistic approaches or with combination of deep learning techniques. There is another family of methods based on *DNN*. These methods are individually capable of navigating the agent intelligently in the presence of dynamic and static obstacles. Pomerleau [22] proposed an end-to-end model of shallow neural network named *ALVINN* (*Autonomous Land Vehicle in a Neural Network*) to steer the mobile robot in the public roads. In recent works, neural networks have gained a lot of attention with the introduction of deep learning by Goodfellow [23]. The resurgence of *DNN* makes the researchers come up with an end-to-end *DNN* model to steer the mobile robot (car) in public roads. *CNN* is the best pattern recognition technique developed so far. CNN extracts the best features from the images directly without any feature extraction technique used in the previous classifiers. Chen et al. [2] proposed a *CNN* to evaluate the safe distance from the preceding obstacles present in the environment and used the same as inputs to control the agent. Bojarski et al. [24] of *NVIDIA* also used CNN to steer the car by using a camera and no goal information was provided. Similar to our work, Xu et al. [25] used the CNN LSTM neural network to predict the discrete action to move the car with a single camera. However, in our work the predicted action is inspired from the human behaviors. Heckar et al. [26] used the 360-degree view of the camera as one input and planned routes as another input to the DNN and predicted the steering and speed, instead of giving a complete route. In our work we are only giving the coordinates of the goal. Müller et al. [27] model of learning was divided in three sections. The first received the segmented data. The second section generated the trajectory based on the driving policy. The last section was the PID controller to control the actuators. This method segmented the data of CARLA simulator to evaluate the model in real world. This method preprocessed the raw image before giving as an input to the model. The methods discussed drive a simulated robot around a facility while avoiding static and dynamic obstacles. When two humans avoid each other, they have some lesser understood conventions that must be honored. The humans already expect the other entity to act as per the conventions. While adhering to the conventions, there is a possibility of having many corner cases that may be barely observable in a general recording of the data. The aim of this paper is thus not to record a huge volume of a general navigation data, but to record behavior-specific data. A machine trained on a behavior specific data is more likely to handle the corner cases well. Furthermore, since a behavior has already been fixed, the data requirement and thus the complexity of the network is limited.

The major drawback of the supervised learning-based motion planning is the need of a large collection of datasets and training the difficult interactive task. Reinforcement learning provides relief of dataset and also trial and error makes the agent learn the difficult task itself. Pan et al. [28] used a simulator to learn the agent behaviors and the authors deployed the agent in the real-world scenarios. Similarly, Shalev-Shwartz et al. [29] applied transfer learning from virtual world to real world based on reinforcement learning. The learning in the reinforcement learning depends upon the policy and it is not necessary the selected policy will converge the training in the right direction. To overcome this problem, Fan et al. [30] deployed the agent in different scenarios with multiple policies that were hybridized with the traditional approach to handle the simple scenarios. On contrary the manual collection of data in supervised learning guarantees to converge the training. We further argue that reinforcement learning attempts to maximize a reward function, while the learnt system is only optimal for a chosen reward function. The humans in everyday life display conventions that are not formally known and thus designing the correct reward function is a very big challenge. Furthermore, the humans react differently in different situations, and thus replacing a human by a simulated agent is not possible, since the simulated agent is not guaranteed to behave like the humans. Behavior specific training can only be done by a supervised training on a data recorded with the help of the humans.

We have reviewed many methods of motion planning, from classical approaches to end-to-end learning. This literature helps the present study of motion planning. This study is inspired from the human's behavior of obstacle avoidance in real life scenarios, including the display of social etiquettes during navigation. The behavior transfers the human-navigation learning to the agent with the belief that the transfer represents the socially accepted strategy of obstacle avoidance in the dynamic environment.

Human inspired reactive motion planning

This section represents the complete flow of the proposed approach, from perception to the obstacle avoidance behaviors.

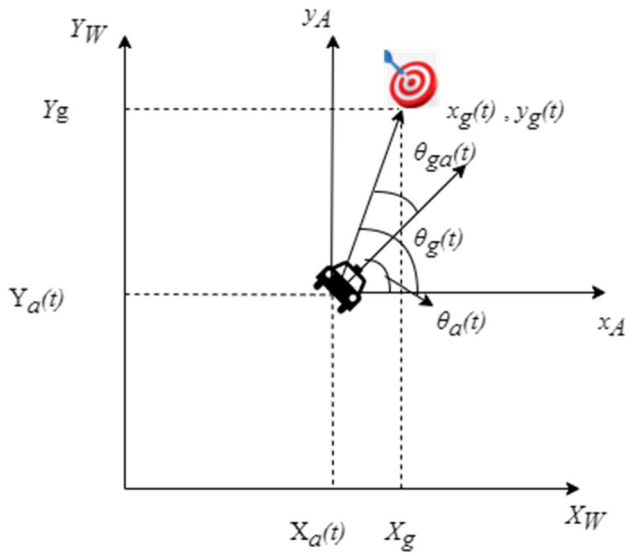


Fig. 2 Goal information with respect to the world frame (X_W, Y_W) and agent frame (X_A, y_A) at time instant t

Perception of the environment

In the area of motion planning an important part is how the agent perceives the environment. The first percept in this proposed method is the raw local information of the environment $R(t)$ at time t to make the agent independent of the environment, while dealing with the dynamic environment. It was observed in the preliminary experiments that the algorithm got confused between the static and dynamic obstacles and the agent seldom stopped in front of the static obstacle with the belief that the static obstacle will cross the agent path. This problem causes the agent to get stuck for an infinitesimal time period. To eradicate this confusion in the algorithm, we have added the intention detection $u(t)$ which identifies the intention of the obstacles, whether they intent to move or stay in the same position. The intention detection module calculates the position vector of the obstacles in the Cartesian space and the angular space frame in every step of time from the raw data of LiDAR.

The last input in the algorithm is the goal information $D(t)$ and it makes the agent capable of aligning in the direction of the goal in the absence of obstacles. The goal information $D(t)$ contains the Euclidean distance of the goal w.r.t the agent frame, $d_g(t)$; and the orientation towards the goal w.r.t the agent frame, $\theta_{ga}(t)$. Suppose the agent is at position $(X_a(t),Y_a(t))$ with an orientation $\theta_a(t)$ as reported by the onboard localization system used by the robot. The goal information calculation is shown in Eqs. (1–5) and the notations are shown in Fig. 2. Equations (2–3) measure the distance to goal $d_g(t)$ with the known agent position $(X_a(t), Y_a(t))$ and goal position (X_g, Y_g). The goal location is assumed to be constant. Equation (2) changes the origin to the location of the agent giving the robot position in the new coordinate axis as $(x_g(t),y_g(t))$, while Eq. (3) calculates the distance to goal. Equation (5) calculates the orientation towards the goal w.r.t. the agent $\theta_{ga}(t)$ as the difference between the angle to goal $\theta_g(t)$ calculated in Eq. (4) and the agent's orientation $\theta_a(t)$. Equation (5) specifically uses the atan2 function to bound the angular difference within the range $(-\pi,\pi]$.

$$D(t) = \left[d_g(t),\, \theta_{ga}(t)\right], \tag{1}$$

$$x_g(t) = X_g - X_a(t),\; y_g(t) = Y_g - Y_a(t), \tag{2}$$

$$d_g(t) = \sqrt{x_g^2(t) + y_g^2(t)}, \tag{3}$$

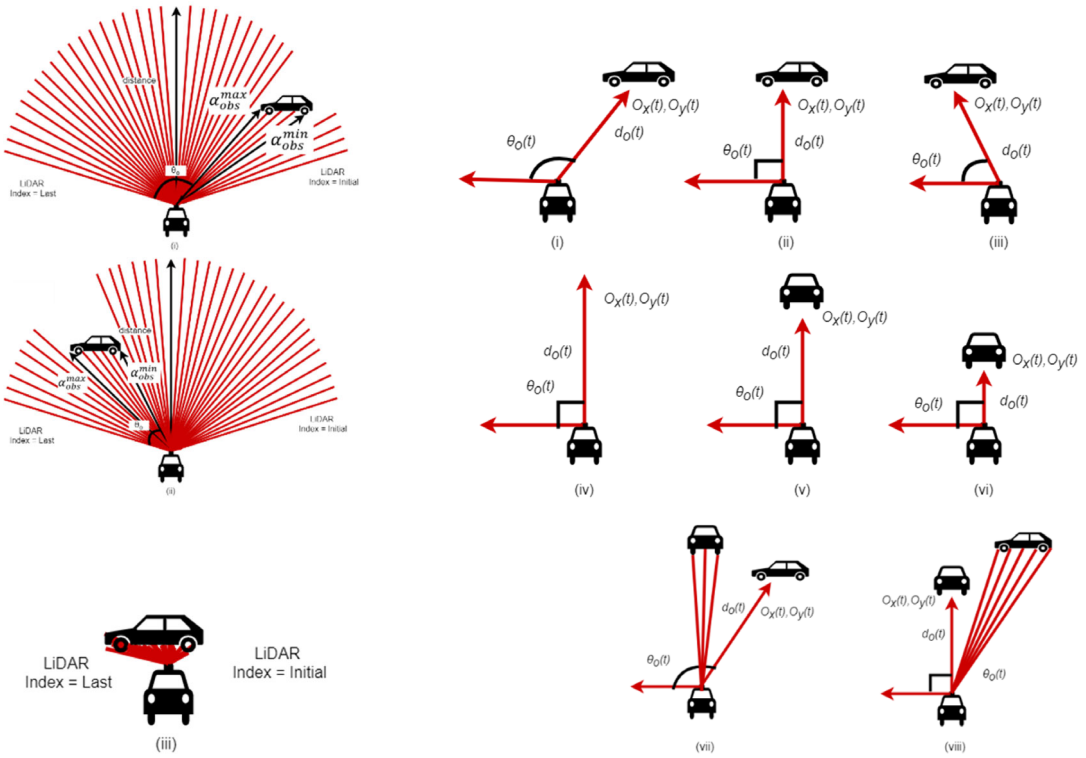
$$\theta_g(t) = atan2\big(y_g(t),\, x_g(t)\big), \tag{4}$$

$$\theta_{ga}(t) = atan2\big(sin\big(\theta_g(t) - \theta_a(t)\big),\, cos\big(\theta_g(t) - \theta_a(t)\big)\big). \tag{5}$$

Intention detection

The intention detection solves the major problem of motion planning algorithm associated with the difference between the dynamic and static obstacle avoidance. Typically, literatures use object level classifications to track and predict the motion of the moving object that can be problematic for false positives and false negatives that may arise for different reasons. Therefore, the proposed work uses raw LiDAR readings to locate a point of interest on the dynamic object that is tracked to guess the intent of the dynamic object. Unlike many other literatures, the dynamic objects are heuristically detected, and their dynamics information is extracted, rather than letting the machine learning approach learn the same computations. The training of the network with multiple objects without intention detection failed many times on scenarios that were not present in training and thus such a network restricted the agent in a similar type of environment as used in training. Intention detection gives additional inputs related to the obstacles that enable the agent to avoid all types of obstacles.

In this detection, the temporal information $u(t)$ of only the nearest dynamic or static obstacle is explicitly calculated while the other obstacles would still be given to the network as the LiDAR raw data of the environment. In Fig. 3a, the FOV (Field of view) of the LiDAR sensor is shown, where the obstacles come under the vicinity of the robot and all three cases of Fig. 3a are implemented in Algorithm 1. In Fig. 3b, where two obstacles come under the vicinity of the robot, but the temporal information is collected for the nearest obstacle



(a) FOV of LiDAR with different scenario of detection

(b) Temporal information calculation

Fig. 3 Intention detection

only and the other obstacle information is still available from the raw LiDAR data $R(t)$. The intention detection algorithm uses the raw LiDAR sensor readings to locate a characteristic point on the obstacle surface used for tracking. The process is shown in Algorithm 1. Line 1 checks if at least one obstacle is within a distance of $d_{\min}$, in which case it can affect the motion of the robot. Let $R_\alpha(t)$ be the laser reading at an angle of α at time t. α_{obs} is the angular range of the nearest obstacle detected by the LiDAR scan angles, shown in Fig. 3. A laser ray strikes at a general angle α and once it hits an obstacle, the laser scanner reports the distance to the obstacle at the angle α. The laser scanner scans at all angles between an initial and a final angle, in steps that are specific to the LiDAR sensor. The closest obstacle will correspond to the smallest laser ray reading with a value of $min(R(t))$, say that corresponds to the laser ray at an angle α. Since the robot is a smooth circular body, laser rays just before α (and similarly after α) will also hit the robot body at a distance of just greater than $min(R(t))$. Assume that the change in the distance reading at the exterior surface of the obstacle body facing the robot is ε, meaning that the obstacle exterior body facing the robot has a distance range of $[min(R(t)), min(R(t)) + \varepsilon]$. ε roughly corresponds to the robot radius which is the variation in distance that the laser ray would experience as it traverses the robot's body. All angles that give readings within this range are found from the LiDAR sensor. The continuous angular range that bounds these LiDAR readings is the angular occupancy of the nearest obstacle, or $\alpha_{\text{obs}} = [\alpha_{obs}^{min}, \alpha_{obs}^{max}]$. The calculation is shown in lines 2, 4 and 5. If the obstacle is to the left of the robot (with a negative angle), the rightmost corner corresponding to α_{obs}^{max} is taken in Lines 6–11; while if the obstacle is to the right of the robot (with a positive angle), the leftmost corner corresponding to α_{obs}^{min} is taken in Lines 13–18. Sometimes the obstacle may be directly ahead of the robot, in which a portion of the obstacle has a positive angular coverage while another portion of the obstacle has a negative angular coverage. In such cases, the leftmost corner corresponding to α_{obs}^{min} is chosen in Lines 20–24.

The network stores the obstacle distance ($d_{\text{o}}(t)$) and obstacle position information ($O_{\text{x}}(t)$, $O_{\text{y}}(t)$), along with the first derivative (or speed, $v_{\text{x}}(t)$, $v_{\text{y}}(t)$) and the second discrete derivative (or acceleration, $a_{\text{x}}(t)$, $a_{\text{y}}(t)$) that is given as an input to the network, shown in Eqs. (6–12). Here h is the time constant between two consecutive sensor readings. The robot is controlled at a constant control cycle (by sleeping for excess time) to maintain a constant time interval between every two readings.

$$u(t) = [O_x(t),\ \ O_y(t),\ \ \theta_o(t), d_o(t),\ \ a_x(t), \\ a_y(t),\ \ v_x(t), v_y(t)] \tag{6}$$

$$O_x(t) = d_o(t)cos(\theta_o) \tag{7}$$

$$O_y(t) = d_o(t)sin(\theta_o), \tag{8}$$

$$v_x(t+1) = \frac{O_x(t+1) - O_x(t)}{h}, \tag{9}$$

$$v_y(t+1) = \frac{O_y(t+1) - O_y(t)}{h}, \tag{10}$$

$$a_x(t+1) = \frac{v_x(t+1) - v_x(t)}{h}, \tag{11}$$

$$a_y(t+1) = \frac{v_y(t+1) - v_y(t)}{h}. \tag{12}$$

.

Algorithm 1: Intention_detection

Input: R(t)
Output: θ_o, d_o, O_x, O_y

If min(R(t)) ≤ d_{min} // (threshold LiDAR range for obstacle detection)
  $\alpha_{obs} = \{\alpha: R_\alpha(t) < \min(R(t)) + \epsilon\}$
else: return (0,0,0,0)
$\alpha_{obs}^{max} = \max(\alpha_{obs})$
$\alpha_{obs}^{min} = \min(\alpha_{obs})$
If $\alpha_{obs}^{max} < 0$
  θ_o(t)= α_{obs}^{max}
  d_o(t)= $R_{\alpha_{obs}^{max}}(t)$
  O_x(t)= d_o(t) cos(θ_o)
  O_y(t)= d_o(t)sin(θ_o)
  return(θ_o(t), d_o(t), O_x(t), O_y(t))
Else
  If $\alpha_{obs}^{min} > 0$
    θ_o(t)= α_{obs}^{min}
    d_o(t)= $R_{\alpha_{obs}^{min}}(t)$
    O_x(t)= d_o(t) cos(θ_o)
    O_y(t)= d_o(t) sin(θ_o)
    return(θ_o(t), d_o(t), O_x(t), O_y(t))
  Else
    θ_o(t)= α_{obs}^{min}
    d_o(t)= $R_{\alpha_{obs}^{min}}(t)$
    O_x(t)= d_o(t) cos(θ_o)
    O_y(t)= d_o(t) sin(θ_o)
    return (θ_o(t), d_o(t), O_x(t), O_y(t))

The novelty of this manuscript is the different reactive behaviors of obstacle avoidance incorporated in a single DNN (deep neural network) function. The single trained function collects the raw data of LiDAR and initiates the action accordingly. The percept of the environment is input to the trained function and the function predicts the desired linear and angular speed to make the agent reach the desired location without colliding with any obstacle. Figure 4 shows the two most common behaviors of humans, and we advocate that these behaviors are robust in nature for the avoidance of dynamic obstacles that can be transferred to mobile robots. We have used the same condition in training. When we deploy the agent in different situations other than those used for training, the agent demonstrated a mixed behavior. The agent did not show the two behaviors explicitly. The trained function was capable of handling the uncertainty. In the experiment section the agent was deployed in 80 novel conditions to check the robustness of the proposed method and the agent successfully navigated in all of them.

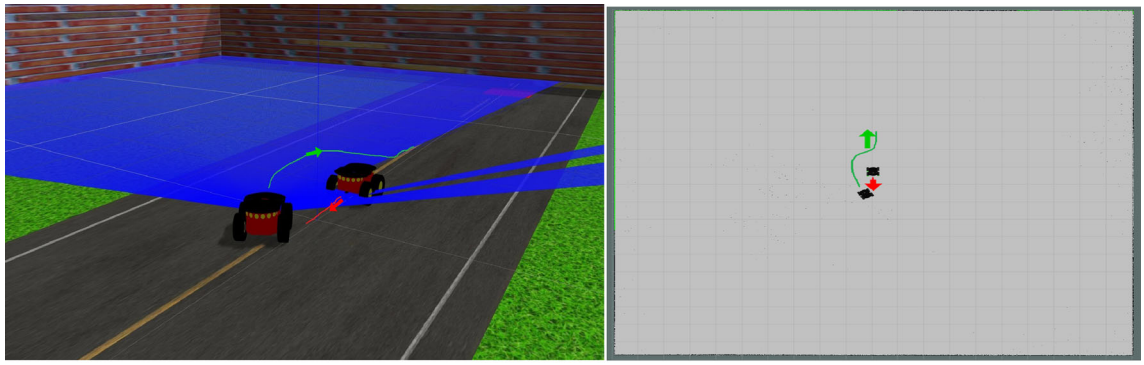
Proposed architecture:

The architecture of DNN based navigation function has three vital inputs. The first is the environment percept input represented by the raw LiDAR readings $R(t)$ that is passed through a 1D CNN to extract the relevant feature from the raw data. The robot sensor percept is sequential in nature. Therefore, the extracted features from the CNN are passed into a LSTM network. The second input $u(t)$ contains the running status of obstacle near to the agent and is directly given to the LSTM network. These data are the sequential information of obstacle motion. The second input plays a significant role in determining the behavior to exhibit. Based on this information, the network can enable or disable the specific behaviors. The extracted feature from both the inputs is concatenated and passed to the fully connected (FC) layer. The third input is the goal location relative to the current robot pose and is directly given to the FC layer. This two-dimension data is directly concatenated with the extracted raw data features and obstacle temporal information features. The FC layer of the model predicts the angular and linear velocity that is given to the actuators. The network architecture is shown in Fig. 5. The inputs and outputs are also shown as Eq. (13). The overall algorithm is given as Algorithm 2.

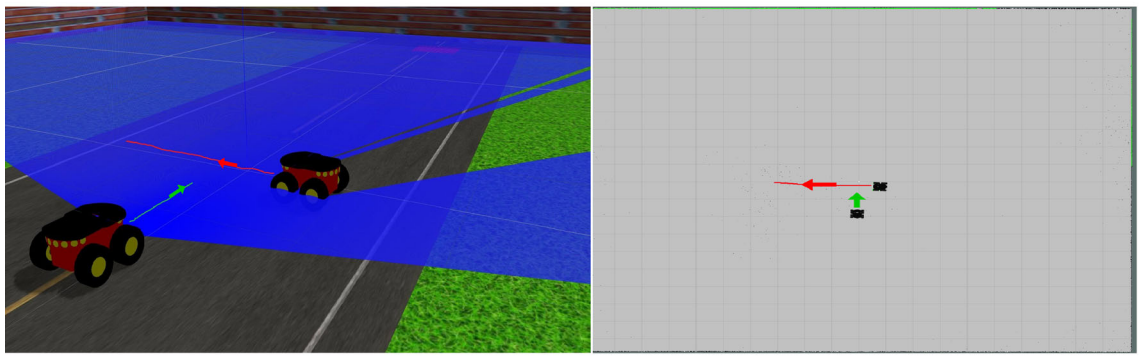
$$[v, \omega] = f(R(t), D(t), u(t)). \tag{13}$$

Algorithm 2: Controller Algorithm LMBRC

Step 1 : RList ← Ø
Step 2 : **while** True
Step 3 : Update R(t) (laser readings, obtained by callback of the laser topic)
Step 4 : Update (X_g, Y_g) (current goal, obtained by callback of the trajectory topic)
Step 5 : Update (X_a(t), Y_a(t)) (Agent pose, obtained by callback of the odometry topic)
Step 6 : [θ_o(t), d_o(t), O_x(t), O_y(t)] = intention_detection(R) //Algorithm 1
Step 7 : [v_x(t), v_y(t)] = velocity_calculation (θ_o, d_o, O_x, O_y) // from equation 9 to 10
Step 8 : [a_x(t), a_y(t)] = acceleration_calculation (v_x, v_y) // from equation 11 to 12
Step 9 : u(t) = [θ_o(t), d_o(t), O_x(t), O_y(t), a_x(t), a_y(t), v_x(t), v_y(t)]
Step 10 : $x_g(t) = X_g - X_a(t), y_g(t) = Y_g - Y_a(t)$
Step 11 : $d_g(t) = \sqrt{x_g^2(t) + y_g^2(t)}$
Step 12 : $\theta_g(t) = atan2(y_g(t), x_g(t))$
Step 13 : $\theta_g(t) = atan2(sin(\theta_g(t) - \theta_a(t)),\ cos(\theta_g(t) - \theta_a(t)))$
Step 14 : D = [d_g(t), θ_g(t)]
Step 15 : Add R(t) to RList
Step 16 : I(t) = [RList, D(t), u(t)]
Step 17 : Normalize I(t)
Step 18 : **If** size(RList) = N_p
Step 19 : [v(t), ω(t)] = f(I(t))
Step 20 : Publish v(t) and ω(t) command the robot
Step 21 : RList ← Ø
Step 22 : sleep to maintain constant rate (N_pr)
Step 23 : **end while**



(a)



(b)

Fig. 4 Reactive behavior transfer to the robot (**a**) Head-on collision avoidance (**b**) "Stop and move" behavior

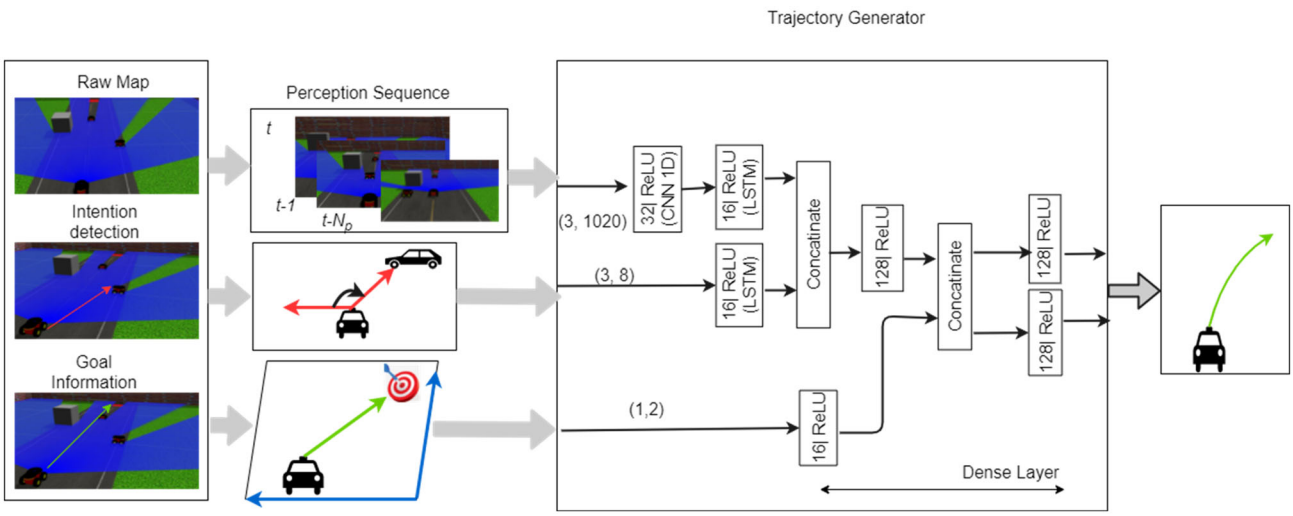


Fig. 5 Network architecture

Results

The algorithm does not have many parameters that significantly affect the algorithm's performance. There are a few parameters only to mention. The value of $d_{\min}$ was taken as 10 m in the paper. From the experimental point of view this parameter does not have any effect and can be set to an arbitrarily large value. It is believed that obstacles beyond a distance should have no effect on the decision making of the robot, a theoretical guarantee that enables simulators restrict proximity queries to within the specified threshold. Practically an obstacle too far off will have a negligible effect in

Table 1 Computation time of DWA, TEB, LMBRC and NPVO

Methods	Computation time (milliseconds)
DWA	126
TEB	113
LMBRC	**101**
NPVO	106

the motion of the robot. However, to ascertain those obstacles very far off do not affect the working of the robot, a hard limit of $d_{\min}$ (10 m) is set. ε roughly corresponds to the robot radius which is the variation in distance that the laser ray would experience as it traverses the robot's body. The algorithm appends N_p inputs as a sequence to extract latent temporal parameters. Since it significantly affects the dimensionality of the input, the value is taken to be a small value as $N_p = 3$. A higher value would significantly increase the input dimensions, while a smaller value would not get enough temporal samples to extract the latent temporal parameters. h, the time constant between two sensor readings, was used to denote the measurement of velocity and acceleration however the parameter is subsumed by the weight of the model.

The other parameters correspond to the architecture of the network. Like many deep learning approaches, we argue that the architectural parameters do not significantly affect the performance of the network, however, the quality of data does affect the performance. The architectural parameters shown in Fig. 5. The LiDAR data have a dimension of 1 × 1020, which after appending $N_p = 3$ data samples make the input of 3 × 1020 dimensions that is passed through the 1D CNN with filter size 32 and ReLU activation function to extract the relevant features. The LSTM has an output dimension of 16 and ReLU activation function. The second input of obstacle information has a dimension of 3 × 8 that is directly given to a LSTM network with 16 output dimension and ReLU activation function. A concatenation of the two LSTM outputs is given to a Fully Connected (FC) layer of 128 neurons and ReLU activation function. The third input of goal information with a dimension of 1 × 2 is directly given to a FC layer of 16 neurons and ReLU activation function. The outputs of the two FC layers are concatenated to represent the latent input features. These features are converted into linear velocity and angular velocity using two separate FC layers of size 128 each and a ReLU activation function.

The agent with approximately $rd = 0.25$ m radius is deployed in a lane of 4 m and the agent's initial position is in the 1-m radius with center (− 8.48, 0.08) and the goal position is taken as (8.48, 0.08). The agent must reach to the 0.5-m radius of the goal position. The agent starts tracking the obstacles when they come in the radius of $d_{min} = 10$ m. The agent is controlled at a frequency of 10 Hz.

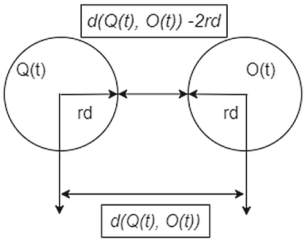


Fig. 6 Clearance calculation between agent $Q(t)$ and obstacle $O(t)$ and rd is the agent radius

Table 2 Quantitatively analysis of DWA, TEB, NPVO and LMBRC in the 20 static environment

Reactive controller	Clearance mean [m]	Travelled distance mean [m]	Travelling time mean [s]
DWA	1.31	16.92	11.2
TEB	1.16	16.77	**8.5**
NPVO	1.28	16.92	21.1
LMBRC	**1.65**	**16.43**	12.5

The robot was steered using a joystick from the third view to record the dataset. In general, with a third view, the human operator can look at the obstacles which may not be within the sensor's field of view and make maneuvers to avoid such obstacles in anticipation, something that cannot be learnt. However, the results reported here are behavior specific and only contain the obstacles that participate in the exhibition of the behavior. The robot's LiDAR sensor used in this approach can look at all the other dynamic obstacles and there is no occlusion. On the contrary, if the first view was used using a front mounted camera, the human operator would not have known about the dynamic obstacles at the side (or back) that are beyond the camera's field of view. A LiDAR sensor however scans 360 degrees leaving only a few laser rays that strike the robot's body and the LiDAR has a complete view of the map in this case. The complete dataset generated by the approach is available at [31].

The proposed algorithm's performance evaluation was done in static and dynamic environments with different scenarios. The proposed algorithm's performance is quantitatively compared with the following three state-of-the-art algorithms to evaluate the robustness. In the end statistical test was applied to find the relevance of quantitative analysis.

- The *Dynamic Window Approach* (DWA) [11]: The ROS DWA planner used. The agent was equipped with a DWA planner deployed in the environment with 80 scenarios.
- *The Timed elastic band* (Teb) [32]: The ROS Teb planner was used. It was also deployed in 80 different scenarios.

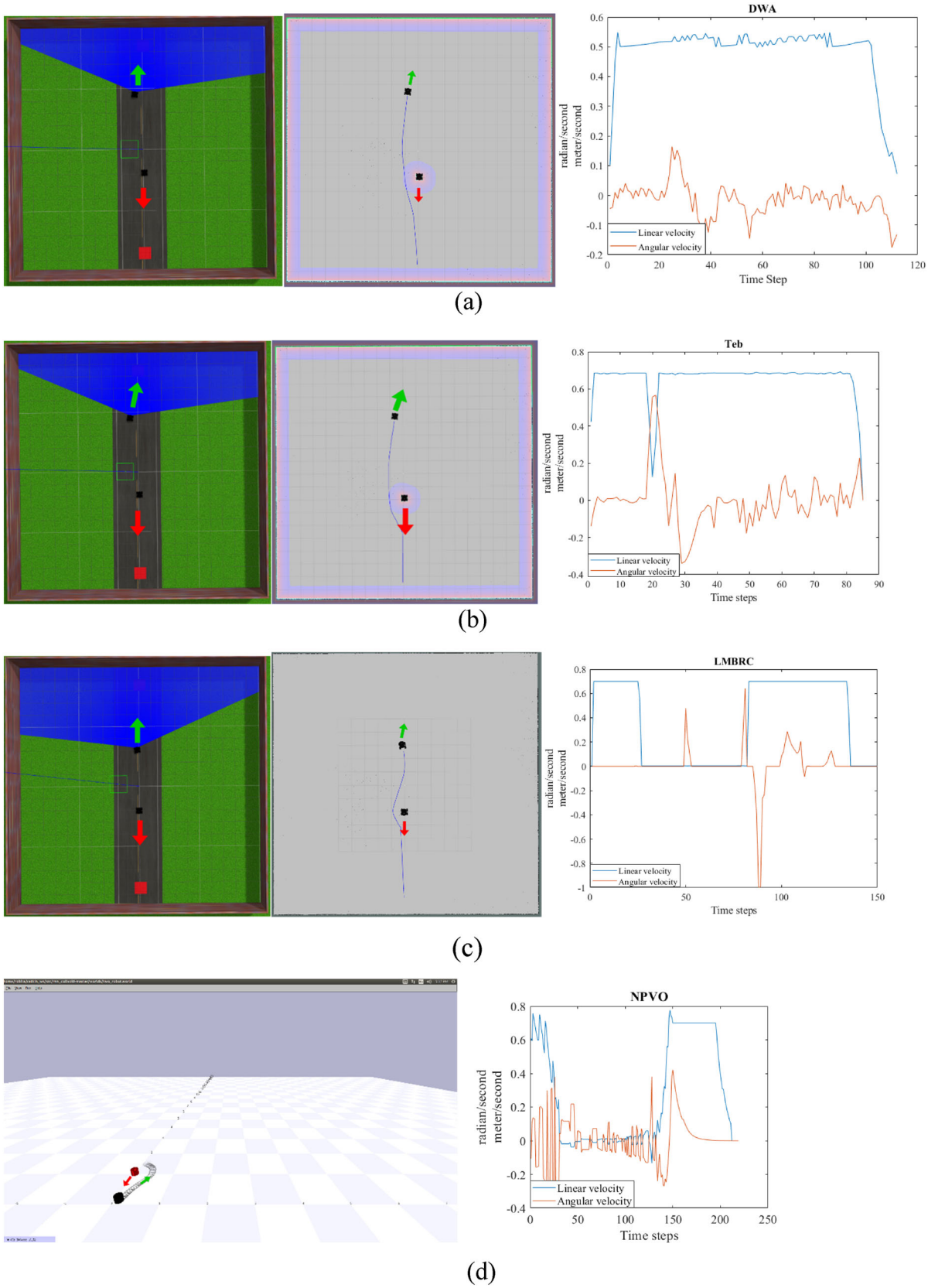


Fig. 7 Static obstacle avoidance representation in one instance and velocities: (**a**) DWA (**b**) TEB (**c**) LMBRC (**d**) NPVO

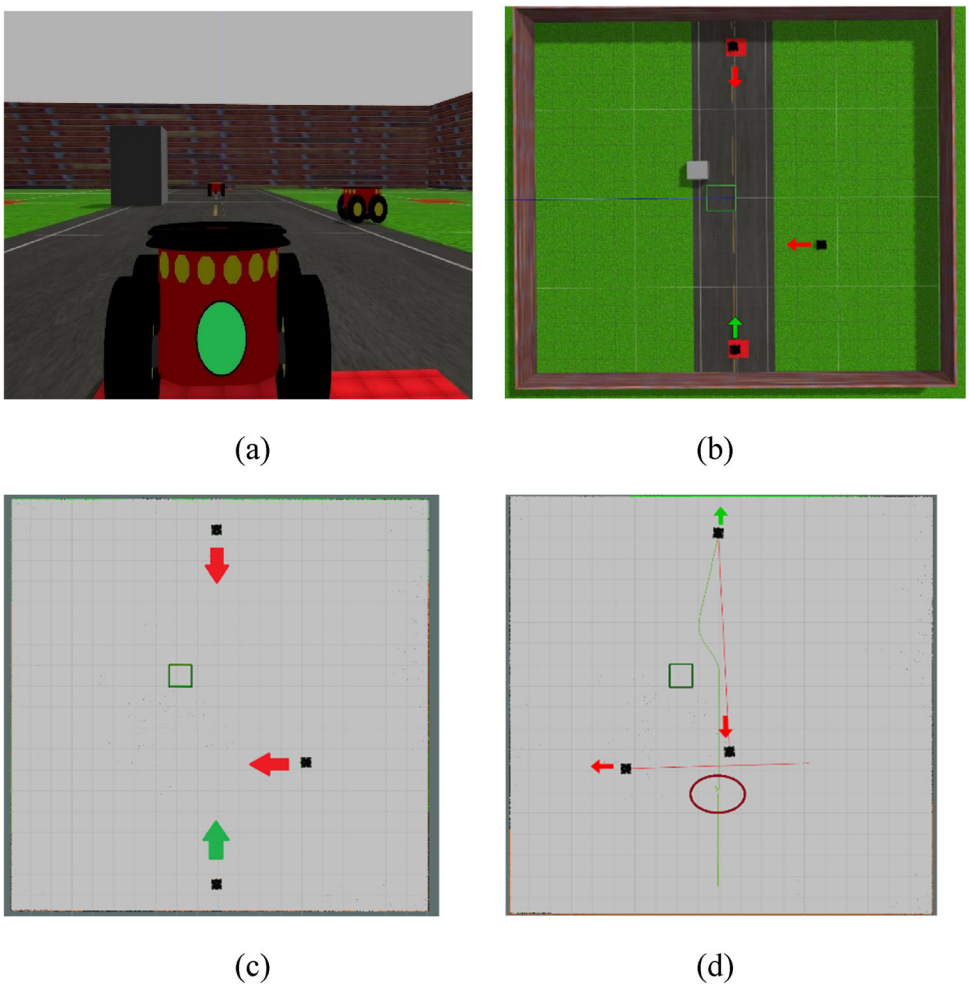


Fig. 8 One Scenario of Dynamic obstacle environment: (**a**) Front View; green dot representing the mobile robot is the agent the other two are obstacles (**b**) Top View (**c**) Rviz representation; green arrow is the agent direction of motion, red arrow is the obstacle direction of motion and the box is the static obstacle (**d**) Agent reaches the destination; green is the agent trajectory, red is the obstacle trajectory and the notch of trajectory inside the red circle represents the agent waiting to let the obstacle cross

- *Nonlinear Probabilistic Velocity Obstacle* (NPVO) [21]: This method used the power of LSTM to predict the motion of dynamic obstacle and the NPVO selected the collision free velocity.

Evaluation metric

We first evaluate all the methods based on computational time. In Table 1, it is shown that while all the approaches take competing computation time, *LSTM Based Reactive Controller* (LMBRC) takes slightly less time compared to the other state-of-art methods. The table shows that a single-function based proposed algorithm takes slightly less time as compared to map-based methods and the other methods like NPVO. The computation time is the time elapsed between two consecutive publishing velocity by the methods.

To further investigate the robustness of the proposed method, we calculate four metrics named clearance, travelled distance, success (to reach the destination) and travelling time. We also considered the velocity pattern generated by the controller in all three methods. Clearance calculates the minimum distance between the agent Q and the nearest dynamic or static obstacle O shown in Eq. (14). It is assumed that the agent and the obstacle are both circular in nature. With this assumption, the clearance at any point of time is given by the distance between the centres minus the sum of radii of the agent and the obstacle. The calculation of clearance is shown in Fig. 6, where $d()$ is the Euclidean distance function and rd is the radius of agent and the obstacle. Clearance is a spatial metric that will change as the robot and obstacles move along with time. The minimum clearance in the entire run is computed as the clearance value of the scenario that measures the least distance between the agent and the obstacle when the two entities are closest to each other, shown in Eq. (15). The travelled distance is the distance travelled by the agent while moving towards the destination shown in Eq. (16). The travelling time is the time taken by the agent to travel from the source to the destination, which is an additional evaluation metric.

$$Clearance(CL(t)) = \min_{O} d(Q(t), \mathrm{O}(t)) - 2rd. \tag{14}$$

Table 3 Quantitative analysis of DWA, TEB, NPVO and LMBRC with obstacle velocity of 0.25 m/s

S.no	Obstacle velocity [0.25 m/s]															
	DWA				TEB				NPVO				LMBRC			
	CL	TD	S/F	TT	CL	TD	S/F	TT	CL	TD	S/F	TT	CL	TD	S/F	TT
1	0.35	16.79	1	11.7	0.55	16.57	1	9.5	0.59	17.70	1	12.6	0.11	16.50	1	10.5
2	1.46	16.83	1	11.7	0.78	16.62	1	9.3	0.89	16.45	1	10.6	1.48	16.28	1	8.4
3	0.21	16.73	1	12.0	0.68	16.74	1	10.6	0.72	16.68	1	12.1	0.20	16.36	1	10.3
4	0.42	16.85	1	11.5	0.55	16.83	1	10.2	0.64	17.40	1	13.3	1.41	16.53	1	9.1
5	NA	NA	0	NA	0.42	16.87	1	13.3	0.32	17.78	1	12.4	0.91	16.48	1	9.7
6	NA	NA	0	NA	0.39	16.94	1	14.6	0.53	16.54	1	12.1	0.47	16.39	1	10.3
7	NA	NA	0	NA	0.38	16.94	1	12.9	0.41	16.65	1	12.2	0.87	16.45	1	9.7
8	NA	NA	0	NA	0.31	16.92	1	13.4	0.25	17.35	1	12.5	0.68	16.42	1	10.0
9	0.33	16.86	1	12.1	0.38	16.89	1	12.8	0.51	16.90	1	11.9	0.29	16.41	1	11.2
10	0.43	16.91	1	11.6	0.34	16.76	1	12.7	0.47	16.60	1	10.7	0.91	16.38	1	8.3
11	NA	NA	0	NA	0.24	16.76	1	13.1	0.52	17.40	1	11.1	1.05	16.38	1	8.6
12	NA	NA	0	NA	0.23	16.90	1	10.9	NA	NA	0	NA	1.97	16.39	1	9.4
13	NA	NA	0	NA	0.21	16.91	1	11.5	0.32	18.44	1	10.9	1.88	16.45	1	12.1
14	0.19	16.82	1	12.1	0.23	16.92	1	11.2	0.39	18.37	1	12.6	1.70	16.45	1	10.1
15	NA	NA	0	NA	0.17	16.88	1	12.2	0.68	16.70	1	12.2	1.94	16.38	1	13.6
16	0.09	16.94	1	14.6	0.33	16.84	1	10.7	0.56	16.66	1	12.9	1.82	16.40	1	10.9
17	NA	NA	0	NA	0.21	16.83	1	10.9	0.33	16.69	1	11.6	1.82	16.49	1	13.2
18	NA	NA	0	NA	0.32	17.09	1	11.2	0.61	16.79	1	13.0	1.79	16.46	1	13.9
19	NA	NA	0	NA	0.18	17.11	1	13.7	NA	NA	0	NA	0.69	16.41	1	8.3
20	NA	NA	0	NA	0.23	16.98	1	14.6	0.19	16.68	1	13.1	1.53	16.23	1	10.5
Avg	0.44	16.84	40%	12.2	0.35	16.86	**100%**	12.0	0.49	17.09	90%	12.1	**1.18**	**16.41**	**100%**	**10.4**

$$Clearance(CL) = \min_{t} CL(t) = \min_{t,O} d(Q(t), \mathrm{O}(t)) - 2rd, \tag{15}$$

$$Travelleddistance(TD) = \sum_{t} d(Q(t+1), R(t)). \tag{16}$$

Static environment

In this experiment, the environment was set up with a static obstacle in 20 different places and the algorithm asks the agent to avoid the obstacle to reach its destination. In the proposed approach avoiding static becomes challenging because *"stop and move"* behavior confuses between the dynamic and static obstacle. In [33], it was argued that to know the intention of an obstacle either the motion of the obstacle or static object level classification is necessary. On the contrary, intention detection of obstacles removes this necessity. In Table 2, the clearance mean of the proposed method *LSTM Based Reactive Controller* (LMBRC) is better than all the three methods, the robot travelled lesser distance while moving towards the destination and however, it takes more time than DWA and TEB. It was observed LMBRC immediately aligns towards the destination while the rest of the methods take time to change their alignment and waste time, also increasing the travelled distance. That is why even having smaller clearance mean, they travel more distance as compared to LMBRC, shown in Fig. 6. The Figures show the travelling time using the number of time steps. A time step is 0.1 s in all the results. The only drawback of LMBRC while avoiding static obstacles is that sometimes it takes time to decide either to avoid the obstacle or to wait to let the obstacle pass that increases the travel time in a completely static environment. The same problem is faced by NVPO. It takes more time than LMBRC shown in Fig. 7c, d, where NPVO takes double time steps to reach the destination all three methods. The specific scenario used for testing is however deceptive in nature. Static obstacles generally comprise of walls, furnishing, etc. that are typically avoided by a deliberative planner that breaks goal-seeking as a sequence of sub-goal seeking behaviours. The reactive navigation thus does not need to account for the static obstacles and such modelling is absent in reactive approaches like the Velocity Obstacle method [34] and all their recent variants. To create a static obstacle, the

Table 4 Quantitative analysis of DWA, TEB, NPVO and LMBRC with obstacle velocity of 0.5 m/s

S.No	Obstacle velocity [0.5 m/s]															
	DWA				TEB				NPVO				LMBRC			
	CL	TD	S/F	TT	CL	TD	S/F	TT	CL	TD	S/F	TT	CL	TD	S/F	TT
1	0.52	16.69	1	11.8	0.49	16.81	1	11.6	0.48	17.83	1	12.5	0.06	16.37	1	10.1
2	1.48	16.67	1	11.8	0.71	16.55	1	12.3	0.83	16.60	1	11.7	0.54	16.22	1	8.7
3	0.12	16.73	1	11.6	0.64	16.69	1	12.2	1.04	16.71	1	12.7	0.07	16.34	1	12.2
4	NA	NA	0	NA	NA	NA	0	NA	0.47	19.34	1	14.3	0.12	16.27	1	12.7
5	0.86	17.01	1	11.9	0.63	16.90	1	12.2	0.18	21.14	1	15.3	1.72	16.38	1	10.7
6	0.14	16.96	1	11.9	NA	NA	0	NA	0.60	16.69	1	12.1	1.0	16.32	1	11.2
7	NA	NA	0	NA	NA	NA	0	NA	0.46	16.64	1	11.8	0.96	16.31	1	14.1
8	NA	NA	0	NA	NA	NA	0	NA	0.49	16.74	1	11.7	1.66	16.38	1	8.9
9	0.54	16.75	1	11.1	0.28	17.36	1	9.9	NA	NA	0	NA	1.55	16.68	1	10.2
10	NA	NA	0	NA	NA	NA	0	NA	0.26	19.66	1	13.4	0.09	16.82	1	11.8
11	0.56	16.88	1	13.0	0.30	17.97	1	12.7	0.23	16.62	1	11.9	1.60	16.33	1	10.0
12	0.52	16.98	1	12.6	NA	NA	0	NA	0.27	16.71	1	10.8	2.88	16.50	1	11.5
13	0.29	16.92	1	12.7	0.46	17.95	1	12.1	0.25	16.67	1	11.7	0.41	16.45	1	9.7
14	0.39	16.97	1	13.2	NA	NA	0	NA	0.26	19.56	1	12.4	1.88	16.35	1	8.3
15	NA	NA	0	NA	0.05	17.55	1	13.9	0.60	16.71	1	12.0	2.13	16.37	1	9.3
16	NA	NA	0	NA	NA	NA	0	NA	0.09	16.70	1	11.9	1.28	16.44	1	11.1
17	NA	NA	0	NA	NA	NA	0	NA	0.14	16.71	1	10.4	1.12	16.50	1	12.8
18	NA	NA	0	NA	NA	NA	0	NA	0.25	17.27	1	12.8	1.66	16.40	1	11.3
19	NA	NA	0	NA	NA	NA	0	NA	0.44	16.71	1	11.7	1.69	16.39	1	10.7
20	NA	NA	0	NA	0.24	16.93	1	9.2	0.41	17.96	1	12.3	1.59	16.40	1	9.9
Avg	0.54	16.85	50%	12.2	0.40	17.19	45%	11.8	0.41	17.52	95%	12.3	**1.19**	**16.41**	**100%**	**10.8**

approach therefore uses another robot that is typically supposed to move, however the robot is not allowed to move creating a static environment. The robot looks at another robot and from the training data assumes that the robot would move and adjusts its speed accordingly. The robot may prefer to wait for the other robot due to similarity with the training cases. The evaluation of LMBRC in static environment shows that the intention detection eventually eradicates the confusion state of the agent. This increases the travel time. However, such cases are practically hard to come. Driving a vehicle following another vehicle is a common behavior and correspondingly slowing down once the vehicle ahead slows down is a common behavior. It is a rarity to have the vehicle ahead broken-down requiring surpassing the vehicle. Even though the time taken is larger, the algorithm can overcome the static obstacle. The NPVO method was evaluated in the *Stage simulator* due to the dependency of the method, however, we scaled the environment in terms of performance like the *Gazebo simulator*.

Dynamic environment

We quantitatively analyze the behavior of agents with dynamic obstacles with the same evaluation metrics. Due to the similarity of velocity of the agent and the obstacles, we do not specifically analyze the agent velocity with the dynamic obstacles. The only one significant change came in linear velocity of NPVO and LMBRC, the waiting time to decide the intention of the obstacle significantly decreased. LMBRC easily decided that the obstacle is dynamic using intention detection and NPVO anticipated the behavior of the dynamic obstacle easily. The agent was equipped with all four methods of study one by one and deployed in the environment shown in Fig. 8. Figure 8a, b show the front view and top view of the environment in the Gazebo simulator. In Fig. 8c, d, the obstacle and the agent are shown in the Rviz simulator. The static obstacle will not be detected when it goes outside the FOV (Field of view) of the agent. In Fig. 8c, the dynamic and static obstacle both come in the FOV of the agent. The green rectangle shows the static obstacle, while the dynamic obstacles are shown by black boxes with arrows showing the

Table 5 Quantitative analysis of DWA, TEB, NPVO and LMBRC with obstacle velocity of 0.75 m/s

S.No	Obstacle velocity [0.75 m/s]															
	DWA				TEB				NPVO				LMBRC			
	CL	TD	S/F	TT	CL	TD	S/F	TT	CL	TD	S/F	TT	CL	TD	S/F	TT
1	0.10	16.77	1	11.9	NA	NA	0	NA	0.32	16.71	1	12.0	0.83	16.41	1	8.5
2	1.12	16.80	1	12.0	0.73	16.75	1	8.7	0.69	16.47	1	10.4	1.40	16.40	1	8.2
3	NA	NA	0	NA	0.64	16.71	1	8.4	0.93	16.43	1	11.1	NA	NA	0	NA
4	NA	NA	0	NA	NA	NA	0	NA	0.45	17.63	1	11.8	0.42	16.43	1	9.9
5	0.41	16.91	1	11.6	NA	NA	0	NA	0.46	17.49	1	12.3	1.23	16.38	1	8.8
6	NA	NA	0	NA	NA	NA	0	NA	0.32	23.06	1	14.6	1.84	16.50	1	9.3
7	1.05	16.77	1	12.1	0.50	16.83	1	9.3	0.47	19.15	1	12.8	0.48	16.41	1	8.8
8	0.23	16.90	1	12.5	NA	NA	0	NA	0.40	17.93	1	12.3	0.66	16.36	1	9.3
9	0.15	16.95	1	12.2	0.32	17.01	1	9.9	0.33	23.63	1	16.1	1.08	16.29	1	8.8
10	0.38	16.71	1	12.0	NA	NA	0	NA	0.15	16.89	1	12.4	0.85	16.47	1	10.1
11	0.15	16.84	1	13.4	0.32	17.19	1	9.8	0.45	19.31	1	14.9	1.93	16.26	1	8.4
12	NA	NA	0	NA	0.66	16.79	1	10.0	0.17	20.18	1	13.9	1.64	16.39	1	9.8
13	0.66	16.87	1	13.2	0.48	16.94	1	9.4	NA	NA	0	NA	1.72	16.46	1	9.9
14	NA	NA	0	NA	0.14	17.21	1	9.7	0.31	19.26	1	14.6	1.34	16.44	1	10.3
15	0.55	16.69	1	12.1	0.42	17.14	1	11.9	0.46	16.66	1	12.5	1.78	16.38	1	9.7
16	0.30	16.96	1	12.5	0.42	17.55	1	11.8	1.0	16.70	1	13.2	0.05	16.45	1	10.7
17	0.57	17.06	1	12.5	0.65	17.20	1	11.7	0.48	17.13	1	13.2	0.65	16.39	1	11.1
18	0.55	17.13	1	13.6	0.23	16.86	1	10.9	0.39	16.67	1	13.6	2.16	16.46	1	8.4
19	NA	NA	0	NA	0.19	16.93	1	8.6	0.47	17.11	1	13.5	NA	NA	0	NA
20	NA	NA	0	NA	0.24	16.90	1	9.1	NA	NA	0	NA	1.06	16.50	1	10.4
Average	0.48	16.87	65%	12.4	0.49	18.30	65%	9.9	0.48	18.24	**90%**	13.1	**1.12**	**16.41**	**90%**	**9.4**

direction of motion. In Fig. 8d, the agent reached the destination, and both the static and dynamic obstacles are out of the FOV of the agent. The spot where the agent waits is marked with a red circle.

The main challenge of the proposed work is how to avoid the dynamic obstacle. We deployed the agent in 20 different scenarios of obstacles. The algorithm was tested on not more than 2 dynamic obstacles and in few cases combination of dynamic obstacles and static obstacles. Figure 8 shows only one scenario. The rest of the scenarios contains agent and obstacles in different positions. It was observed in the experiment that the agent shows a mix behavior of both learned behaviors. The other challenge in this approach is the maximum linear velocity of obstacles that the agent can avoid. We have performed 20 scenarios with 4 different velocities of obstacles. The performance of all four methods showed significant changes in their performance. In Table 3, the first experiment performed with the maximum linear velocity of obstacle as 0.25 m/s. DWA performed worst and got the only 40% success, even the maximum velocity of the agent was 0.7 m/s. The other methods performed to an acceptable level. It was observed in the DWA, it shows some sign of "stop and move" but immediately starts looking for the free space to avoid the obstacle that makes it the worst performer even with very low velocity of obstacle. The TEB and NPVO seems good in avoiding the dynamic obstacle and with low velocity both got 100% success in test cases. Still LMBRC looks efficient because of maintaining a higher clearance, lesser travelled distance and a lesser travelled time.

In Table 4, the metric values corresponding to obstacle velocity of 0.5 m/s shown. In this case, both map-based approaches DWA and Teb performed very poorly and only achieved 50% success in test cases. The increased velocity of obstacle does not show a high impact on the learning-based approaches LMBRC and NPVO.

In Tables 5 and 6, the DWA and TEB performance improves slightly, but still maintains a lesser success than NPVO and LMBRC. From Tables 3, 4, 5 and 6, it is clearly observable that the method involving a combination of different algorithms increases the latency and does not make the agent react immediately. In the map-based approach the latency is high due to local map update and local planner and global planner update. In the NPVO, the trajectory of the obstacle is predicted using DNN to update the local map

Table 6 Quantitative analysis of DWA, TEB, NPVO and LMBRC with obstacle velocity of 1 m/s

S.No	Obstacle velocity [1 m/s]															
	DWA				TEB				NPVO				LMBRC			
	CL	TD	S/F	TT	CL	TD	S/F	TT	CL	TD	S/F	TT	CL	TD	S/F	TT
1	NA	NA	0	NA	NA	NA	0	NA	0.37	18.08	1	9.9	0.87	16.43	1	8.8
2	0.78	16.84	1	12.5	0.72	16.99	1	8.7	0.72	16.71	1	11.6	1.03	16.42	1	8.0
3	0.33	16.81	1	11.7	0.77	16.54	1	7.4	0.66	16.68	1	11.7	0.37	16.48	1	8.7
4	0.07	16.89	1	12.1	0.08	17.49	1	9.8	0.30	19.29	1	10.4	0.45	16.28	1	8.3
5	NA	NA	0	NA	0.45	16.76	1	9.3	0.59	17.70	1	12.0	0.84	16.37	1	8.9
6	0.08	16.74	1	11.7	NA	NA	0	NA	0.52	17.21	1	11.8	0.72	16.38	1	8.9
7	0.14	16.75	1	12.3	0.59	16.74	1	9.1	0.47	17.13	1	11.6	0.69	16.42	1	8.5
8	0.04	16.80	1	11.8	0.72	16.91	1	9.3	0.47	18.18	1	12.2	1.02	16.38	1	8.9
9	NA	NA	0	NA	0.21	17.16	1	9.6	0.11	18.98	1	12.3	1.15	16.39	1	8.7
10	NA	NA	0	NA	NA	NA	0	NA	NA	NA	0	NA	1.51	16.42	1	8.8
11	0.18	16.77	1	12.1	0.22	17.12	1	10.3	NA	NA	0	NA	1.83	16.35	1	8.6
12	NA	NA	0	NA	0.24	17.16	1	10.2	0.64	17.88	1	12.4	1.86	16.39	1	8.7
13	NA	NA	0	NA	0.59	16.82	1	9.2	NA	NA	0	NA	1.89	16.51	1	10.4
14	NA	NA	0	NA	0.82	16.60	1	9.4	NA	NA	0	NA	1.58	16.29	1	9.7
15	0.52	16.84	1	11.7	0.57	16.85	1	10.6	NA	NA	0	NA	1.56	16.57	1	10.0
16	0.86	16.90	1	11.9	0.25	16.75	1	10.8	0.48	16.87	1	12.0	1.84	16.36	1	9.1
17	0.51	16.87	1	11.8	0.19	17.27	1	10.7	0.47	16.72	1	13.0	0.56	16.35	1	9.2
18	0.49	16.94	1	12.9	0.32	17.01	1	9.6	0.43	16.69	1	11.7	0.29	16.46	1	9.5
19	0.34	16.94	1	12.2	NA	NA	0	NA	NA	NA	0	NA	0.81	16.40	1	9.3
20	NA	NA	0	NA	NA	NA	0	NA	NA	NA	0	NA	1.20	16.40	1	9.0
Avg	0.35	16.84	60	12.0	0.44	16.94	75	9.6	0.47	17.54	65	11.7	**1.09**	**16.40**	**100**	**9.0**

also has a higher latency of reaction than LMBRC. NPVO failed to avoid the obstacle with velocity of 1 m/s. In LMBRC the latency is very less due to dependency on a single function. The tables also show that LMBRC takes less travel time in all the scenarios as compared to all other algorithms. However, collision in the higher velocity not only depends upon the latency but also on how the agent tries to avoid the obstacle. It is shown in Tables 5 and 6 that the DWA and TEB performance improves as compared to the previous cases.

Statistical analysis

We have performed statistical analysis of the behavior of the agent in the dynamic environment. It is observed in Table 7, that LMBRC maintains a larger clearance mean than the other methods. It was expected from the proposed work, while dealing with dynamic obstacles the extra precaution is necessary to assure the collision free navigation. This proposed work maintains a smaller travelled distance even with larger clearance mean. Path length and clearance are opposing metrics. A high clearance (preferred) typically increases the path length (not-preferred) and vice versa. The algorithms therefore try to maintain a tradeoff between the two opposing factors. An unusual observation from the experiments is that the LMBRC algorithm maintains a larger clearance which however does not affect the path length. This is because the LMBRC algorithm can wait for the obstacles to clear rather than being driven by the obstacles in the case of contradictory navigation goals. Figure 9 shows the specific conditions where the algorithms exhibit the worst performance except the LMBRC algorithm. In Fig. 9a, c, e the robot starts following an obstacle for the sake of open space and this makes the robot maintain a smaller clearance while taking a large path length and time. On contrary, in Fig. 9g, the robot stops, does not move and waits for the obstacle to pass. This behavior makes the robot maintain a large clearance with a small path length. Another reason behind this observation is that LMBRC starts avoiding obstacles earlier and aligns back to the goal slowly in contrast to the other algorithms that saves on the path length metric. In Fig. 9b, d and f the algorithms start avoiding the obstacle only when they are considerably near and aim to steer back quickly, thus forming a steep curve around the obstacle. However, LMBRC

Table 7 (**a**–**d**) Statistical analysis in 80 random scenarios of DWA, TEB, NPVO and LMBRC

Obstacle velocity [m/s] m/s]	Clearance mean (Std.) [m]			
	DWA	TEB	NPVO	LMBRC
(a)				
0.25	0.44 (0.54)	0.35 (0.16)	0.49 (0.35)	**1.18 (0.60)**
0.50	0.54 (0.39)	0.40 (0.21)	0.41 (0.23)	**0.67 (0.78)**
0.75	0.48 (0.34)	0.49 (0.19)	0.48 (0.22)	**1.11 (0.58)**
1	0.38 (0.28)	0.44 (0.24)	0.47 (0.16)	**1.09 (0.52)**
Obstacle velocity [m/s]	Travelled distance mean (Std.) [m]			
	DWA	TEB	NPVO	LMBRC
(b)				
0.25	16.84 (0.06)	16.86 (0.13)	17.09 (0.62)	**16.41 (0.07)**
0.50	16.80 (0.1)	17.18 (0.58)	17.52 (1.37)	**16.36 (0.13)**
0.75	16.87 (0.13)	17.00 (4.39)	18.24 (2.17)	**16.40 (0.06)**
1	16.82 (0.06)	16.93 (0.26)	17.54 (0.89)	**16.40 (0.06)**
Obstacle velocity [m/s]	No. of success/failure			
	DWA	TEB	NPVO	LMBRC
(c)				
0.25	8/12	**20/0**	**20/0**	**20/0**
0.50	10/10	9/11	19/1	**20/0**
0.75	13/7	14/6	**18/2**	**18/2**
1	12/8	15/5	13/7	**20/0**
Obstacle velocity [m/s]	Travelled time mean (Std.) [Sec.]			
	DWA	TEB	NPVO	LMBRC
(d)				
0.25	12.2 (1.01)	12.0 (1.58)	12.1 (0.82)	**10.4 (1.68)**
0.50	12.2 (0.67)	11.8 (1.42)	12.3 (1.12)	**10.8 (1.49)**
0.75	12.4 (0.61)	9.9 (1.19)	13.1 (1.41)	**9.4 (0.87)**
1	12.0 (0.37)	9.6 (0.87)	11.7 (0.81)	**9.0 (0.56)**

starts proactively avoiding the obstacle, and after avoidance slowly drifts towards the goal that makes the traced path close to the geometrically shortest path possible shown in Fig. 9h.

We have performed the One-Tailed test shown in Table 8 to find the difference between LMBRC with the other methods and the calculated p-value of clearance, travelled distance and travelled time between LMBRC and other methods obtained lower than 3%. The lower p-value implies that the difference between both algorithms is significantly different.

Experiments under noisy settings

In this section, we demonstrate our experiment in an environment like real time. The real time environments are ubiquitous of noises in the sensor data. These noises make the robot inconsistent in its behavior. To understand the robustness of the proposed algorithm in a real time environment, we have introduced a Gaussian Noise in the LiDAR sensor and deployed the agent. Figure 10 shows four different levels of Gaussian noises in the LiDAR sensor.

From Table 9a it is seen that the noise with mean 0 and standard deviation 0.5 does not much affect the functionality of the algorithms and the agent manages to reach to the destination. In Table 9b–d, the map-based navigation approaches

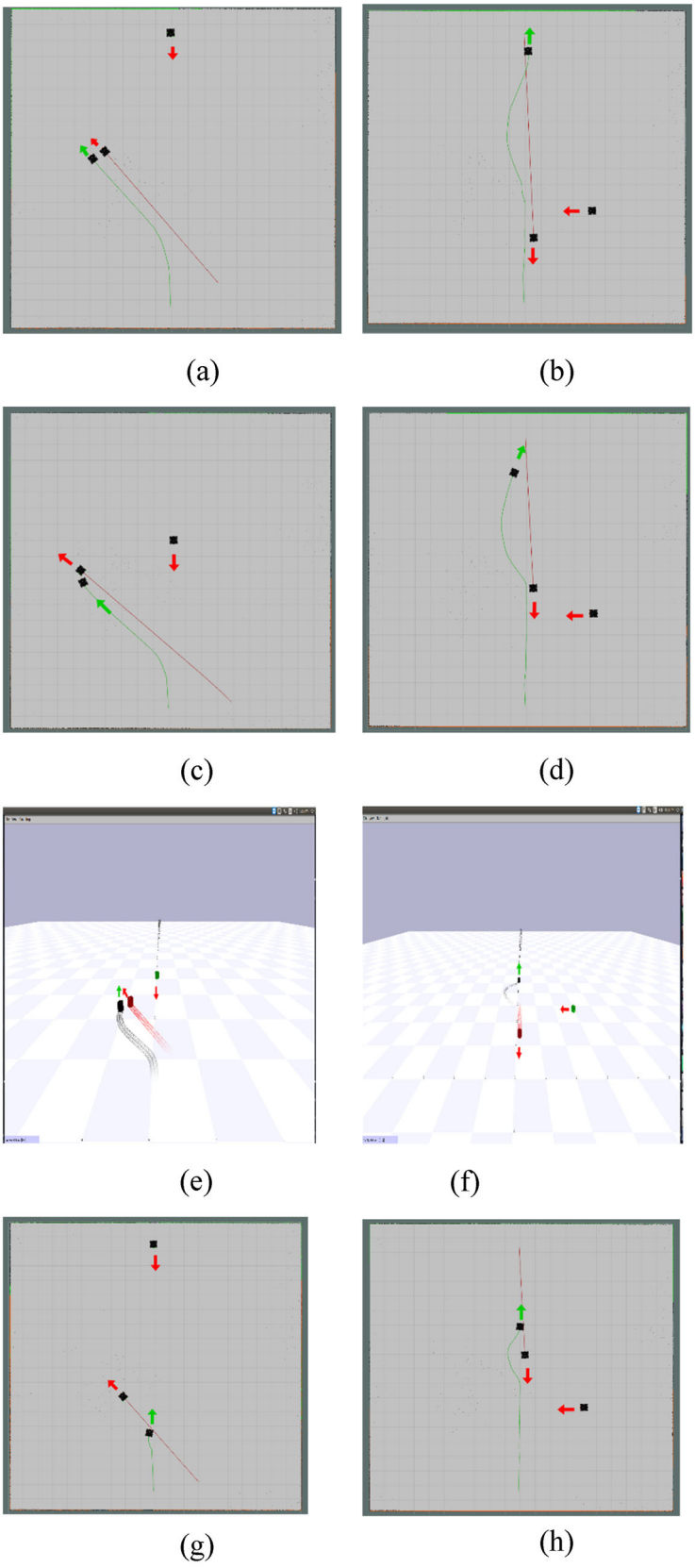


Fig. 9 Behavior pattern: (**a**–**b**) DWA (**c**–**d**) TEB (**e**–**f**) NPVO (**g**–**h**) LMBRC; obstacle path is red in color and agent path is green in color

Table 8 One-Tailed Test between LMBRC with DWA, TEB and NPVO

Obstacle velocity [m/s]	Clearance*P*-value			Travelled distance*P*-value			Travelled time*P*-value		
	DWA	TEB	NPVO	DWA	TEB	NPVO	DWA	TEB	NPVO
0.25	0.001088	9.81E-05	0.000262	1.39E-09	6.35E-13	1.65E-04	1.3E-03	0.0022	2.13E-04
0.5	3.17E-06	2.90E-07	9.82E-05	3.20E-08	1.152E-3	1.19E-03	2.9E-04	0.0477	4.69E-04
0.75	0.000179	0.000115	0.001596	1.10E-09	8.83E-08	1.16E-03	2.4E-12	0.1109	2.87E-10
1	2.31E-08	1.04E-06	3.46E-07	4.40E-15	4.56E-07	2.87E-04	4.6E-18	0.0153	7.44E-10

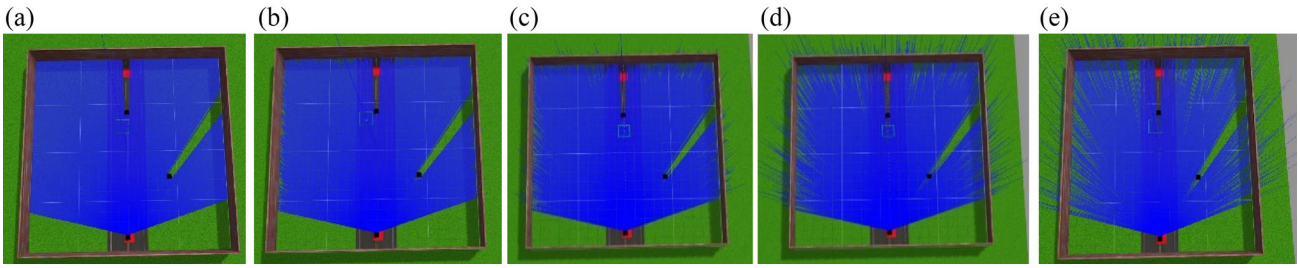


Fig. 10 Different level of Gaussian Noise (mean, standard deviation): (**a**) 0, 0 (**b**) 0, 0.5 (**c**) 0, 1 (**d**) 0, 2 (**e**) 0,3

and NPVO do not make the agent to reach and collide with the obstacle and make the robot stuck. On the contrary LMBRC managed to reach the destination but the increase of noise decreases its efficiency.

Conclusions

Reactive navigation or the direct mapping of input to output solves a lot of challenges of map-based navigation enduring since decades like software complexity and hard to reconfigure for the desired behavior. In general, for the reactive navigation algorithm the transient time between input to output action should be less, then only the collision with the dynamic obstacles can be avoided. Simultaneously the algorithm should also be concerned about the goal. For example, the human avoids the obstacle while he/she is also concerned that the deviation should not move him/her significantly away from the goal.

In the proposed network the raw data of the environment and the intention detection input of dynamic obstacles near to the agent makes the algorithm take extra precautions than the prevalent end-to-end algorithms. It was observed in the behavior of the agent that the intention detection input made the agent aware of the motion information of the obstacle, whether the obstacle is coming from the left or right without requiring an object-level classification. We have shown that the training with two behaviors does not only handle the two situations but also avoids the obstacle in 80 novel situations. The training with three different datasets to a single network without intention detection was difficult and confused the agent between static and dynamic obstacles. The intention detection input helped the agent to identify in between them.

The inspiration for the architecture was to provide safety in end-to-end learning, which is the major work in this area of research and a single function leads towards the destination. This neural network has three inputs. The first input passes through CNN and extracts the relevant information from the raw data of LiDAR. The second input extracts motion features of the nearest obstacle and directly passes to the LSTM. The third input is the goal information sent to the FC layer of the network. The concatenation of different layers was a difficult problem in the architecture, and it was possible only with multiple trial and errors. This is a novel architecture of the neural network and experiments show that it can handle the dynamic and static obstacles simultaneously and also lead to the destination.

The experiment procedure is the essence of this prototype developed for reactive navigation. In most literatures, the dataset is collected from sophisticated roads and also tested on roads used for training, where the agents move in different lanes. Outdoor scenarios have limited well-established behaviors, while the behaviors are less formally studied and categorized for the indoor scenarios. The approach thus focusses on such behaviors that have neither got a lot of interest in the literature, nor are likely to be recorded from a continuous stream of generic data. Even if such behaviors are recorded, they are likely to be filtered out as noise by a generic machine learning system. In the experiment, we intentionally make the obstacle try to hit the agent, also crossing the agent path from a different angle and with a different linear velocity to robustly train the agent.

Table 9 (**a**–**d**) Quantitative Analysis of DWA, TEB, LMBRC and NPVO in Noisy LiDAR sensor data

Methods	Gaussian noise (mean = 0.0, Stddev = 0.5)			
	CL (m)	TD (m)	S/F	TT (seconds)
(a)				
DWA	0.38	21.00	1	14
TEB	0.44	20.30	1	11
LMBRC	**1.2**	**16.56**	**1**	**10**
NPVO	0.50	17.67	1	11.5
Methods	Gaussian noise (mean = 0.0, Stddev = 1)			
	CL (m)	TD (m)	S/F	TT (seconds)
(b)				
DWA	NA	NA	0 (Collision)	NA
TEB	NA	NA	0 (Collision)	NA
LMBRC	**1**	**16.96**	**1**	**15.5**
NPVO	NA	NA	0 (Collision)	NA
Methods	Gaussian noise (mean = 0.0, Stddev = 2)			
	CL (m)	TD (m)	S/F	TT (seconds)
(c)				
DWA	NA	NA	0 (Collision)	NA
TEB	NA	NA	0 (Collision)	NA
LMBRC	**0.5**	**18**	**1**	**17**
NPVO	NA	NA	0 (Collision)	NA
Methods	Gaussian noise (mean = 0.0, Stddev = 3)			
	CL (m)	TD (m)	S/F	TT (seconds)
(d)				
DWA	NA	NA	0 (Collision)	NA
TEB	NA	NA	0 (stuck)	NA
LMBRC	**0.3**	**20.5**	**1 (away from goal)**	**19**
NPVO	NA	NA	0 (stuck)	NA

We have reviewed many literatures and no literature explicitly discusses about the maximum obstacle velocity. In the preliminary experiments, we observed that it is an important parameter to be discussed and also to evaluate the performance of the proposed algorithm. The experiments concern the indoor environments like hotels, hospitals, and airports and outdoor environments like pedestrians. The scenarios witness different obstacles with different preferred speeds.

The proposed DNN based architecture of reactive navigation inherits serval the strong points. (i) It can avoid dynamic obstacles with a velocity twice of the agent velocity. (ii) Tracking of the nearest dynamic obstacle without object-level classification provides extra safeguard from the collision. (iii) A single trained function makes the agent capable of choosing from multiple obstacle avoidance behaviors while leading the agent to the destination. (iv) First time in end-to-end motion planning, the obstacle avoidance behavior is inspired from specific obstacle avoidance behaviors of humans that are unlikely to be captured in a generic data.

The proposed work restricted the agent deployment in the environment of two dynamic obstacles only. This work is a testbed to incorporate the human intention in motion planning. We found that using a joystick for controlling the agent is the immediate way of involving the human intention. The future plan for this research is to collect the data of human trajectory in different scenarios like pedestrians, mall and

railway station where the dataset can incorporate human etiquettes and to train a DNN to find the efficient policy of obstacle avoidance.

In [30], the authors argue that the supervised learning requires a lot of data and reinforcement learning (RL) reduces the dependency of manual collection of a dataset, but at the same time the policy of RL does not make the agent to learn the social etiquettes of motion planning of the human and instead optimizes against a parametrized reward function. On the contrary, the problem in hand is to learn primitive behaviors that neither require a lot of data, nor are the scenarios too complex. While the number of behaviors may be large, navigation is largely a function of the obstacles in the immediate vicinity, against which the motion is predominantly a single behavior. In the future, we intent to incorporate the inverse reinforcement learning (IRL) to derive a reward function for Reinforcement Learning using the trajectory data. In the future we also intent to fuse the ability of Reinforcement Learning to handle complex obstacles with the ability of supervised learning to better capture the human social etiquette into one navigation network. Further, the testing needs to be done for real robots rather than relying on simulators.

Author contributions SMHJ carried out the complete experiment. SMHJ and Rk contributed to the design and implementation of the research, to the analysis of the results and to the writing of the manuscript.

Funding The work is sponsored by the Indian Institute of Information Technology, Allahabad.

Data availability Dataset available in the https://doi.org/10.17632/n8c423y4h6.1.

Declarations

Conflict of interest The authors declare that they have no conflict of interest. The authors declare that the work described has not been published previously, that is not under the consideration for publication elsewhere, that its publication is approved by all the authors and tacitly or explicitly by the responsible authorities where the work has been carried out, and that, if accepted, it will not be published elsewhere in the same form, in English or in any other language, including electronically without written consent of copyright- holder. The authors don't have any competing interest.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

1. Van den Berg J, Lin M, Manocha D (2008) Reciprocal velocity obstacles for real-time multi-agent navigation. In: 2008 IEEE international conference on robotics and automation (pp 1928–1935).
2. Chen C, Seff A, Kornhauser A, Xiao J (2015) Deepdriving: learning affordance for direct perception in autonomous driving. In: Proceedings of the IEEE international conference on computer vision (pp 2722–2730).
3. "The City-People walking on the street overhead view" *YouTube*, uploaded by Jonathan Lang, 10 March, 2014, https://youtu.be/26F_EcacVPU?list=PL-SckuqSBraz5GNTl2zE5t9vWxy_wxsIm.
4. Pfeiffer M, Schaeuble M, Nieto J, Siegwart R, Cadena C (2017). From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In: 2017 IEEE international conference on robotics and automation (icra) (pp. 1527–1533). IEEE.
5. Lumelsky VJ, Stepanov AA (1987) Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. Algorithmica 2(1):403–430
6. Kamon I, Rivlin E, Rimon E (1996) A new range-sensor based globally convergent navigation algorithm for mobile robots. In: Proceedings of IEEE International Conference on Robotics and Automation (Vol. 1, pp. 429–435). IEEE.
7. Borenstein J, Koren Y (1989) Real-time obstacle avoidance for fast mobile robots. IEEE Trans Syst Man Cybern 19(5):1179–1187
8. Hwang YK, Ahuja N (1992) A potential field approach to path planning. IEEE Trans Robot Autom 8(1):23–32
9. Borenstein J, Koren Y (1991) The vector field histogram-fast obstacle avoidance for mobile robots. IEEE Trans Robot Autom 7(3):278–288
10. Ulrich I, Borenstein J (1998) May. VFH+: Reliable obstacle avoidance for fast mobile robots. In: Proceedings. 1998 IEEE international conference on robotics and automation (Cat. No. 98CH36146) 2: 1572–1577. IEEE.
11. Fox D, Burgard W, Thrun S (1997) The dynamic window approach to collision avoidance. IEEE Robot Autom Mag 4(1):23–33
12. Simmons, R., 1996, April. The curvature-velocity method for local obstacle avoidance. In: Proceedings of IEEE international conference on robotics and automation 4: 3375–3382. IEEE.
13. Minguez J, Montano L (2004) Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios. IEEE Trans Robot Autom 20(1):45–59
14. Durham JW, Bullo F (2008) September. Smooth nearness-diagram navigation. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp 690–695). IEEE.
15. Bai X, Yan W, Cao M (2017) Clustering-based algorithms for multivehicle task assignment in a time-invariant drift field. IEEE Robotics and Automation Letters 2(4):2166–2173
16. Bai X, Yan W, Ge SS, Cao M (2018) An integrated multi-population genetic algorithm for multi-vehicle task assignment in a drift field. Inf Sci 453:227–238
17. Ko NY, Simmons RG (1998) October. The lane-curvature method for local obstacle avoidance. In: Proceedings. 1998 IEEE/RSJ

International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190) (Vol. 3, pp 1615–1621). IEEE.
18. Fernández JL, Sanz R, Benayas JA, Diéguez AR (2004) Improving collision avoidance for mobile robots in partially known environments: the beam curvature method. Robot Auton Syst 46(4):205–219
19. Welch, G. and Bishop, G., 1995. An introduction to the Kalman filter: 127–132.
20. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
21. Kurtz V, Lin H (2019) Toward verifiable real-time obstacle motion prediction for dynamic collision avoidance. In: 2019 American Control Conference (ACC) pp 2633–2638. IEEE.
22. Pomerleau DA (1988) Alvinn: an autonomous land vehicle in a neural network. Advances in neural information processing systems, 1: 305–313.
23. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT press.
24. Bojarski M, Yeres P, Choromanska A, Choromanski K, Firner B, Jackel L, Muller U (2017) Explaining how a deep neural network trained with end-to-end learning steers a car. arXiv preprint arXiv:1704.07911.
25. Xu H, Gao Y, Yu F, Darrell T (2017) End-to-end learning of driving models from large-scale video datasets. In: Proceedings of the IEEE conference on computer vision and pattern recognition (pp 2174–2182).
26. Hecker S, Dai D, Van Gool L (2018) Learning driving models with a surround-view camera system and a route planner. arXiv preprint arXiv:1803.10158.
27. Müller M, Dosovitskiy A, Ghanem B, Koltun V (2018) Driving policy transfer via modularity and abstraction. In: Conference on Robot Learning (pp 1–15). PMLR.
28. Pan X, You Y, Wang Z, Lu C (2017) Virtual to real reinforcement learning for autonomous driving. arXiv preprint arXiv:1704.03952.
29. Shalev-Shwartz S, Shammah S, Shashua A, (2016) Safe, multi-agent, reinforcement learning for autonomous driving. arXiv preprint arXiv:1610.03295.
30. Fan T, Long P, Liu W, Pan J (2020) Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. Int J Robot Res 39(7):856–892
31. Jafri SMH (2022) “Head-on collision, "stop and move" and static”, Mendeley Data, V1, https://doi.org/10.17632/n8c423y4h6.1, https://data.mendeley.com/datasets/n8c423y4h6/1
32. Rösmann C, Feiten W, Wösch T, Hoffmann F, Bertram T (2013) September. Efficient trajectory optimization using a sparse model. In: 2013 European Conference on Mobile Robots (pp 138–143). IEEE.
33. Everett M, Chen YF, How JP (2018) October. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS) (pp 3052–3059). IEEE.
34. Fiorini P, Shiller Z (1998) Motion planning in dynamic environments using velocity obstacles. Int J Robot Res 17(7):760–772