Collision Avoidance in Dynamic Environment: An ICS-Checker Using Extremal Trajectories

Khalil Muhammad Zuhaib[1,*], Aneela Pathan[1], Abid Ali Shah[2], Junaid Iqbal Bhatti[3], Arsalan Ahmed Soho[1],

[1]Department of Electronic Engineering, The University of Larkano, Larkana, Pakistan
[2]Department of Electrical Engineering, The University of Larkano, Larkana, Pakistan
[3]Department of Mechanical Engineering, The University of Larkano, Larkana, Pakistan
[*]Corresponding author: kmzuhaib@uolrk.edu.pk

Abstract

In this article, we have addressed the problem of navigating a vehicle with kinodynamic constraints in an environment where the trajectory of dynamic obstacles can be partially predictable. This study introduces an Inevitable state-checker named ICS^e-Test, which is an approach grounded in the concept of Inevitable Collision States (ICS). This primary contribution of the study, unlike the state of the art, is the selection of a fixed number of maneuvers for the ICS^e-Test, regardless of the number of dynamic obstacles present in the environment. Further, the paper proposed a collision avoidance scheme based on ICS^e-Test, and it is compared with the present approaches. The results show the superior performance of the proposed Inevitable state-checker using extremal trajectories.

Keywords—Mobile Robot, Collision Avoidance, Inevitable Collision State, Dynamic Environment, Robot Navigation

1 Introduction

In today's modern world, most robotics applications involve mobile robot navigation in a dynamic environment where there are other moving agents like humans, vehicles, and other robots. Such environments are uncertain due to the presence of multiple moving obstacles whose behavior is only partially predictable (if at all) [1-3]. The further application of autonomous navigation of robots is spatial exploration. In such applications, the robot does not have complete information about its environment due to the limitations of its onboard sensors. In these applications, partial motion plans are planned instead of a complete plan for the goal and or the sub-goal [4,5]. To successfully complete the given task, the robot has to safely navigate among obstacles by guaranteeing its safety and that of its surroundings.

In dynamic environments, collision-free decisions are to be made in a limited time while considering the future motion of the obstacles and appropriate look ahead [6-7]. In making such decisions, there are two issues that need to be addressed. The first issue is concerned with estimating the future behavior of moving obstacles over future time intervals. Various works can be found in the literature on the said problem [8-10]. After obtaining and examining the future model, the next challenging issue is to analyze it to develop a safe navigation strategy. This paper focuses on addressing this challenge by building on the idea of Inevitable Collision States (ICS). The concept of ICS was first presented in [11,12], and it defines ICS as a state where a collision is inevitable, regardless of the vehicle's future trajectory. In theory, finding whether a state qualifies as an ICS involves evaluating all possible infinite-length trajectories a robot could follow from that state [13]. This paper proposes a conservative ICS test approach that considers only four trajectories of finite length based on Velocity Obstacle (VO) [14,15] reasoning. Based on it, a collision avoidance approach is devised that ensures the robot will always be in a state where there is at least one future trajectory that allows the robot to escape the collision. This paper will also present the result of extensive simulation trials of the proposed ICS test and collision avoidance algorithm,

ISSN: 2523-0379 (Online), ISSN: 1605-8607 (Print)
DOI: https://doi.org/10.52584/QRJ.2202.01

and present the comparative analysis of these results with other approaches.

2 Related Work

The concept of ICS highlights the difficulty of ensuring collision avoidance, as it demands predicting the environment's future evolution while accounting for potentially infinite computational overhead. Since such conditions are nearly unattainable in real-world scenarios and the characterization of ICS is highly intricate, many researchers have explored relaxing the ICS requirements. Some of the early work for collision avoidance is based on τ-safety; approaches that compute future trajectories that are safe over the duration of τ seconds (hopefully significant to compute an updated safe trajectory) [16,17]. Planning a path that is collision-free during some time period of τ seconds does not guarantee safety. Because, let's say we are somehow able to compute a path that is collision-free and is τ seconds long, the question is, what will happen at the end of this trajectory? What if it leads the system toward the wall at high speed? A collision will eventually occur. So selecting a τ-safe path cannot guarantee our safety. Some other work is based on evasive trajectories: approaches that ensure that the robot will always be in a state where it can carry out an evasive maneuver, like a braking maneuver for a car-like robot or a circling maneuver for an aerial vehicle [18-20]. In [21], the author explores the viability theory for the safe motion of systems with multiple motion constraints. In [22], the authors proposed the Imitating Maneuvers (IM) approach that ensures that the robot remains in states where it is possible for the robot to attain and retain zero relative velocity with respect to moving obstacles. In this approach, the computation required to find out whether a state is ICS increases with the number of moving obstacles, which is a disadvantage. This is because the set of control sequences for the ICS test increases with the number of dynamic obstacles. In this article, we have proposed extreme trajectories/sequences for the ICS test. The advantage of the approach is that the control sequences for the ICS test are fixed and independent of the number of Dynamic obstacles.

3 Problem Statement

When planning online for a dynamic environment, the model of dynamic obstacles is considered to be valid over the short time horizon. So, robot navigation requires running sensing, planning, and execution cycles. For planning, a time slot is allocated to compute

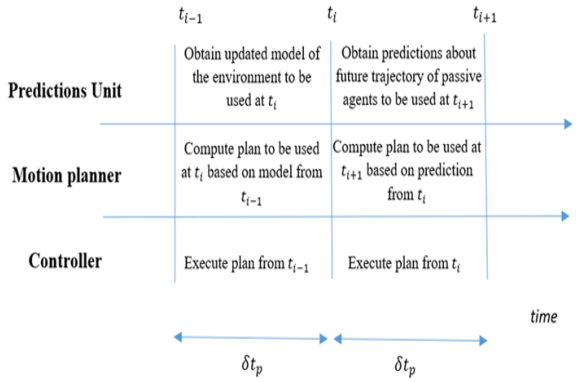

Fig. 1: Sense Plan Act Synchronization Scheme

the solution to the goal. However, in general, the computation of a complete trajectory of the robot to the goal point cannot be guaranteed. A sense plan, an act scheme for navigation in a dynamic environment, is explained below:

So we are concerned with a partial plan, planned over a specific time horizon $\tau > \delta t_p$ that leads the system closer to a given goal. A plan computed in every planning cycle, say (t_{i-1}, t_i), is executed in the next cycle (t_i, t_{i+1}) for a time interval $t_p < \tau$ seconds long, as shown in Figure 01. In order to safely navigate the robot, the plan computed in every planning cycle, say (t_{i-1}, t_i), for the execution in the next cycle (t_i, t_{i+1}) should not result in an Inevitable-Collision-State pass t_{i+1}. Now, we redefine the problem as follows: The problem is to compute a plan to be executed during (t_i, t_{i+1}) that is (a) collision-free, (b) does not result in an inevitable collision state past t_{i+1}, and (c) choosing an optimal partial trajectory in every planning cycle that leads towards the goal point.

4 Inevitable Collision States

We will begin by revisiting the velocity obstacle concept, as outlined in [23] and [24]. Based on this concept, the Inevitable Collision states are defined for a kinodynamically constrained system, which will eventually lead us to introduce the concept of ICS testing using extremal sequences.

4.1 Velocity Obstacle (Revisited)

Velocity Obstacle VO_{t_o} computed at time $t = t_0$ represents the set of absolute velocities that would lead the robot to collide with the neighboring obstacle within the time window $[t_o,]$ [23-25]. Selecting a single velocity outside VO_{t_o} at $t = t_o$ guarantees no collision at all times as long as the obstacle remains on its current trajectory. $VO_{t_0}$is constructed in terms of the

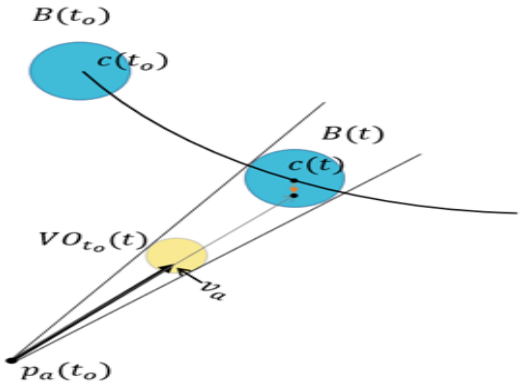

Fig. 2: The temporal component of velocity obstacle $VO_{t0}(t)$ at time t

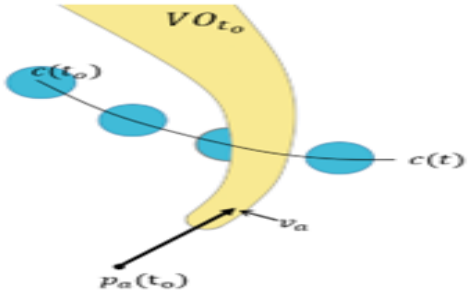

Fig. 3: Velocity Obstacle shown as the union of its temporal component

union of temporal elements $VO_{t_0}(t)$, which is the set of absolute velocities that would lead the robot to collide with an obstacle at a specific time t. Let the set of points held by a planner obstacle at time $t[t_o,]$ be given by $B(t)$, and the obstacle's general trajectory is $c(t)$. Then, a temporary component can now be expressed as follows:

$$VO_{t_0}(t) = \frac{c(t) \bigoplus B(t)}{t - t_0} \tag{1}$$

Where $c(t)B(t)$ represents the Minkowski sum of $c(t)$ and $B(t)$, as shown in Figure 2.

Now VO_{t_0} can be defined in terms of $VO_{t_0}(t)$ as follows:

$$VO_{t_0} = \bigcup_{t \in [t_o, \infty]} VO_{t0}(t) \tag{2}$$

Geometrically, VO_{t_0} is of the shape of a warped cone in velocity space, as can be seen in Figure 3.

4.2 System with kinodynamic constraints

If the robot's state is such that its absolute velocity is in VO_{t_0}, then the collision will eventually occur if

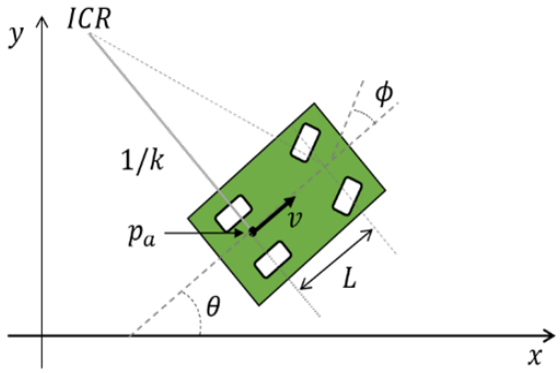

Fig. 4: The kinematic car-like model. The states are given by the position vector p_a, orientation angle θ, and steering angle ϕ. The wheelbase is represented by L. The instantaneous velocity v aligns with the direction of the rear wheels. Here, ICR stands for Instantaneous Center of Rotation.

it continues to move with its absolute velocity into the future. To avoid a collision, it is desirable for the robot to select its new absolute position outside the VO_{t_0} instantaneously (at $t = t_0$). A mobile robot with kinodynamic constraints can't do so. For example, for a car-like robotic system, as in Figure 04, the instantaneous velocity can only be in the direction of the rear wheels. However, it is possible to maneuver the kinodynamically constrained system to attain any absolute velocity over time. Now we will define an ICS for a kinodynamically constrained system in terms of a velocity obstacle.

4.3 Inevitable Collision State (ICS)

Let $s \in S$ be the system state at time t, and let U be a bounded control space. Let us denote a state transition equation of a robot in general form as:

$$\dot{s} = f(s, u) \tag{3}$$

where $u \in U$. Let $\tilde{u} : [0, t_f] \rightarrow U$ denote the control sequence in time for intervals of fixed duration δt. All combinations of control sequences possible over a time interval $[0, t_f]$ are represented by a set $\tilde{U}$. By integrating the state transition (3) from $t = 0$, for sequence $\tilde{u}$ we can obtain a series of states ordered over time, i.e., a curve, in $S \times T$ where T represents the time dimension. The state obtained by integrating (03) at time t for input $\tilde{u}$ is represented by $\tilde{s}(s(0), \tilde{u}, t)$. In order to formally define ICS in terms of velocity obstacle, we will represent the state by $s = (p, v, \tilde{z})$, where the position is represented by p, the cartesian velocity is represented by v and $\tilde{z}$ is the rest of the state parameters. Now we can define ICS as follows:

Definition 01(ICS): A state $s(0)$ is an inevitable collision state, if $\forall \tilde{u} \in \tilde{U}$ there does not exist a time $t \in [0, \infty]$ and state $\tilde{s}(s(0), \tilde{u}, t)$ such that $v \notin V0_{t0}$.

According to definition 01, if for a state of the robot, there exists no sequence in $\tilde{U}$ that can move the absolute velocity of the system outside the velocity obstacle, then this state will be an inevitable collision state. In the next section, we will discuss the proposed ICS checking approach named as ICS test.

5 ICT Test

In actual theory, checking whether or not a given state is an ICS requires us to examine state trajectories that a system can follow from a given state for all possible control sequences $\forall \tilde{u} \in \tilde{U}$ of infinite length in time ($t_f = \infty$). Examining all the trajectories for infinite-duration control sequences is not feasible. However, a conservative approximation of the ICS set can be found with a limited number of control sequences within $\tilde{U}$. Considering a conservative estimate of an actual ICS set will not lead us to compromise on safety [22]. Now we will discuss our approach to the ICS-test that considers only extreme trajectories of finite length for checking a state for ICS, in a dynamic environment.

5.1 Extremal Sequences

The conservative estimate ICS in terms of VO_{t_0} allows us to check whether a state s is an ICS state by considering a subset of U. We can consider only those sequences that can bring the velocity vector of the system out of VO_{t_0} in a minimum amount of time. Such time-optimal sequences will be the solution to the following minimization problem:

$$\min_{\tilde{u} \in U} \int_0^{t_1} 1dt \tag{4}$$

Subjected to robot dynamics in (3) and control constraint $u \in U$, with the initial conditions:

$$v_0 \in V_{t0}$$

and final conditions

$$v_1 \notin V_{t1}$$

Where v_o and v_1 are the Cartesian velocities at $t = t_0$ and $t = t_1$ respectively, $(t_o < t_1)$.

The solution to such a time minimization problem is bang-bang control sequences [26]. Such sequences can be integrated to a point in time into the future when the velocity of the system moves outside the velocity obstacle. To explain it in detail, let us now consider a case study of a robot with car-like dynamics.

5.2 Case Study Car-Like Robot

Consider a car-like robot A with nonholonomic constraints, whose steering angle ϕ and linear acceleration $\dot{\alpha}$ is directly controlled. Similar to [15], the curvature u_ϕ is taken as a control input from which the angle of the steering ϕ can be computed as $\phi = tan^{-1} u_\phi L$. In this equation, L denotes the wheelbase of the car-like robot. The state s is expressed as a tuple (p, α, θ), where the robot rear wheel axle position is $p = [xy]^T$, the robot speed is α, and its orientation is θ. The motion is governed by:

$$\begin{aligned} \dot{x} &= \alpha \cos\theta \\ \dot{y} &= \alpha sin\theta \\ \dot{\theta} &= \alpha u_\phi \\ \dot{\alpha} &= u_a \end{aligned}$$

where $|u_a| \leq u_a^{max}$ and $|u_\phi| \leq u_\phi^{max}$ The instantaneous velocity of the robot is $v = [\dot{x} \quad \dot{y}]$. Let us explain the procedure by considering a case shown in Figure 05 (a).

Figure 5(a) shows an obstacle B_j with center p_b at time $t = 0$ moving with velocity v_b towards A with state $s(0)$, for which the corresponding linear velocity is directed along the positive y-axis. Figure 5(b) shows that the relative velocity $v_{ab}(0) \in RVO_j^{s(0)}$ at $t = 0$, where relative velocity obstacle, $RVO = VO_0 - v_b$. If the linear relative velocity of A remains unchanged in the future, then the collision will eventually occur.

Let BO_j^+ be the set of points occupied by an obstacle B_j^+ with the center at the origin, as shown in Figure 5(c). Let $p_{ab} = p - p_b$ be the relative position of A w.r.t. obstacle B_i^+ at time t. The obstacle and robot will be in collision at the time t if $p_{ab}(t) \in BO_j^+$. We can find out whether a given state $s(0)$ at $t = 0$ is ICS_R by checking only for those sequences, that can change $v_ab(0) \in RVC_j^{(s(0)}$ to $v_ab(t) \in RVC_j^{s(t)}$ in minimum possible time t. This problem is solved using the bang-bang principle. Integrating states using extremal control sequences $(\tilde{u}^{+R}, \tilde{u}^{+L}, \tilde{u}^{-R}, \tilde{u}^{+L})$ will result in four extremal trajectories, denoted by $p_{ab}^{+R}, p_{ab}^{+L}, p_{ab}^{-R}, p_{ab}^{-L}$, where + and – denote $u_a = u_a^{max}$ and $u_a = -u_a^{max}$ are applied respectively, L and R denote $u_\phi = +u_\phi^{max}$ and $u_\phi = -u_\phi^{max}$ are applied respectively, see Figure 5(c). It can be seen that two of the four extremals penetrate the obstacle and two do not. For extremal p_{ab}^{+R}, the

time t at which $v_{ab}(t) \notin RCO_j^{s(t)}$ is smallest compared to all possible collision-free trajectories (all trajectories that pass between p_{ab}^{+R} and p_{ab}^{+L}).

Figure 6(a-b) shows a case in which all trajectories except one extremal penetrate the obstacle. Any motion forward without escaping the velocity obstacle will result in a state that will be ICS, as shown in Figure 6 (c) all four extremal trajectories penetrate the obstacle.

In order to know whether a given state $s(0)$ at $t = 0$ is $ICS_R(B_j)$, we integrate extremal up to a point in time t where extremal either penetrate the obstacle BO_j^+ or the relative velocity $v_{ab}(t) \notin RVO_j^{s(t)}$ at time t (detail are in Algorithm 01). If there exists at least one extremal such that $v_{ab}(t) \notin RVO_j^{s(t)}$ at time t, then the given state is not $ICS_R(B_j)$.

The time over which trajectories are required to be integrated will depend upon obstacle size, its velocity, and the given states of the system. However, we can set a maximum limit on the duration of integration.

Algorithm 01 named $ICS^e - test$, summarizes all the steps needed to test a given state for ICS. The input to algorithm 01 is the state of the system, set B_j for N moving obstacles, and their future trajectories $c_j(t)$. If the algorithm finds that the robot's current velocity is in the computed velocity obstacle, then the set of extremal control sequences E is selected from $\tilde{U}$ for checking whether or not the given state s is an ICS.

Algorithm 1 ICS^e-test

Input: s, $c_j(t)$, B_j
Output: Boolean output

```
ICS(B) ← True
if v ∉ ⋃_{j∈N} VO^{s(t_0)} then
    return ICS(B_j) = False
else
    Select E ⊂ Ũ
    for all u ∈ E do
        for i = 0 to m do
            Compute s_{i+1} = s_i + ∫_{t_i}^{t_{i+1}} f(s_i, u_i) dt
            if v_{i+1} ∉ ⋃_{j∈N} VO^{s(t_0)} then
                return ICS(B_j) = False
            end if
        end for
    end for
    return ICS(B)
end if=0
```

6 Safe Navigation

Algorithm 02 provides a summary of the steps involved in calculating the set of controls U_{safe}, which will lead to states at time δt that are not ICS. To do so, the sampling of the system's control space is done in such a way that it always includes the extremals. In this way, it is guaranteed that there will exist at least one $u \in U$ that will lead the system to states that are not ICS. The states obtained $s()$ by integrating the state equation of the system for all control inputs u are tested for ICS. Set U_{safe} will consist of all control inputs that will not lead the system to ICS.

Algorithm 2 Computing U_{safe}

Input: $s(0)$, $c_j(t)$, B_j
Output: U_{safe}

```
U_safe ← ∅
Sample U = {u_1, u_2, ..., u_n}
for all u_i ∈ U do
    s(δt) = s̃(s(0), u_i, δt)
    if s̃(s(0), u_i, [0, δt]) is collision-free then
        if ICS-Check(s(δt)) = False then
            U_safe ← U_safe ∪ u
        end if
    end if
end for=0
```

To navigate the robot toward the goal, a control input in $U_{safe}$is to be selected that leads the robot closer to the goal, p_g. A weighted function of cost built on the distance matrix is computed for the convergence to the goal. Algorithm 03 mentions the step-by-step procedure for the selection of such a control sequence.

Algorithm 3 Find the Best Safe Control Sequence

Input: U_{safe}
Output: u_i

```
min ← ∞
for all u_i ∈ U_safe do
    if ‖f(h(s, u_i, δt)) − p_g‖ < min then
        min ← ‖f(h(s, u_i, δt)) − p_g‖
        arg min ← u_i
    end if
end for
return arg min =0
```

7 Implementation and Results

In this section, the implementation of our proposed approach is presented in a simulated environment. To validate the effectiveness of our approach, this section will compare its performance relative to other existing approaches.

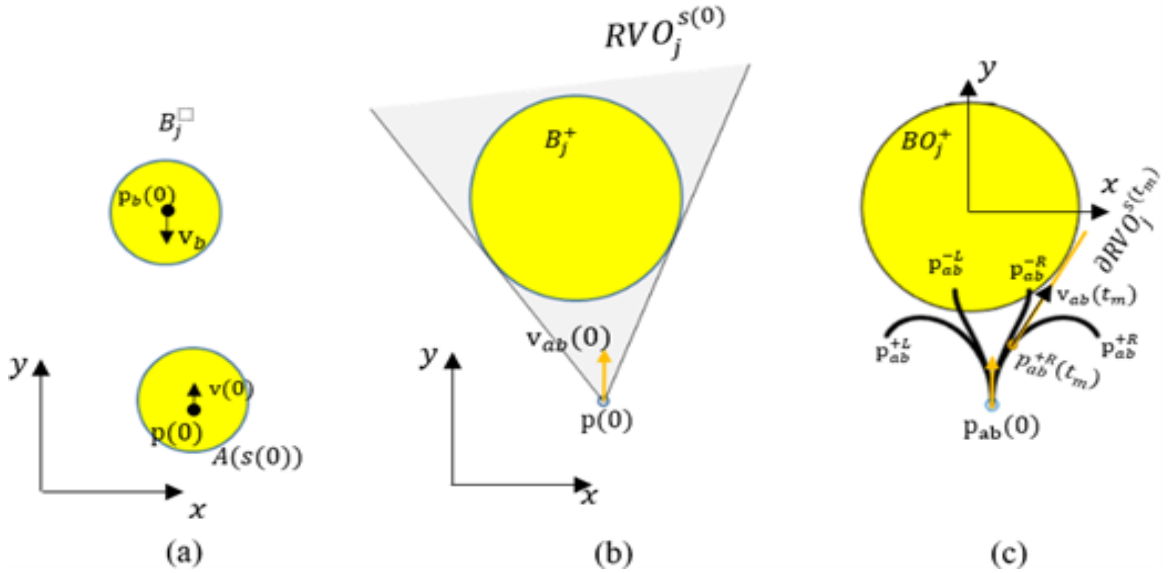

Fig. 5: In (a), robot A is headed toward a collision with an obstacle B_j, (b) shows the relative velocity obstacle RVO, and (c) shows the four extremal trajectories.

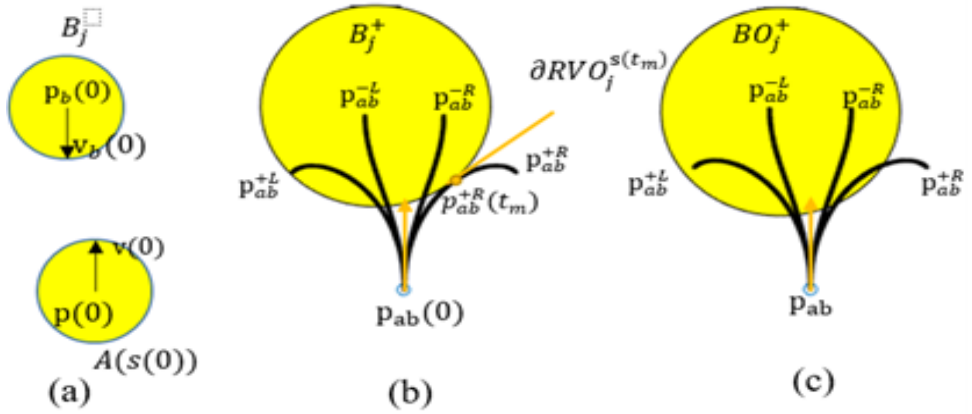

Fig. 6: In (a) robot A is headed toward a collision with an obstacle B_j, (b) shows a case in which the robot is in a state where all extremal trajectories except one will penetrate the obstacle, (c) shows a case in which robot is in a state where all extremal trajectories penetrate the obstacle.

7.1 Implementation Details and Simulation Setup

The simulated experiment was conducted by considering a car-like robot kinodynamic model as was considered in section 5.2. The maximum linear acceleration u_a^{max} and curvature u_ϕ^{max} is are set to be 1 unit/s^2 and 1.5 unit^{-1} respectively. The maximum system speed was set to be 2 units. The robot was of disc shape with its radius set equal to 1 unit. The environment is constructed for simulation, which is a square region 22 units in length and 22 units in width. The environment had several disc-shaped moving obstacles of radius set to 1. In this open environment, the obstacles were set to move along specified paths, i.e., straight lines and arcs, with a maximum speed limit of $\pm$ 1 unit/s. The obstacles are capable of changing their path, and the probability that they can change their trajectory within one second was set equal to 0.1. A scan cycle of 0.05 seconds and a duration τ of 3.5s was used. Figure 07 shows several simulation snapshots captured at different moments, displaying a green robot navigating through multiple moving red obstacles to reach the yellow-marked goal position. The technique was built in C++, and the simulations were run on a Core i5-3550 machine with 4GB of RAM.

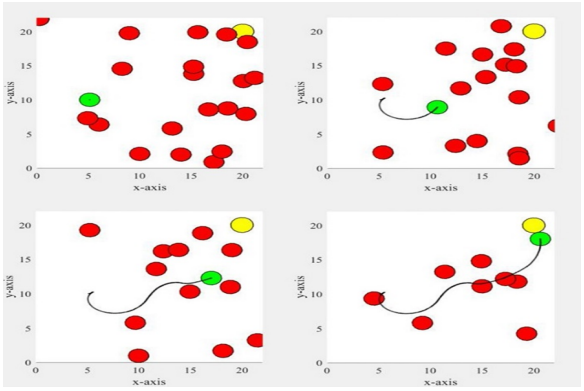

Fig. 7: Collision avoidance in a dynamic environment where obstacle has free will. Several simulation snapshots captured at different moments display a green robot navigating through multiple moving red obstacles to reach the yellow-marked goal position. The probability that the obstacle will change its current trajectory within one second is 0.1.

7.2 Comparative analysis

This section will compare two other cutting-edge collision avoidance techniques with the navigation strategy based on our suggested ICS test approach, i) Time-varying dynamic window [27] and ii) ICS-Avoid [19]. Both strategies are popular and frequently applied in practical settings. The performance of each approach is tested and compared in terms of four sets of parameters, namely Collision on average, Average number of maneuvers used to test a state, computation time for ICS-check, and success rate at which the robot accomplishes the task of reaching the goal point without running into any obstacle.

7.2.1 Experiment 01 – ICS^e-Test Comparison

For ICS^e-Test comparison, the probability that the object would change its current path in the future was first set to 0. The information about the future trajectory was made available to the three schemes for 1, 3, and 5 seconds. Each simulation run lasted two minutes. Table 1 displays the average number of collisions that occur between robot A and obstacle B_j during the run for 20 trials for each time horizon.

In Table 1, TVDW shows the weakest performance of the three. This is due to the fact that it only makes limited use of the obstacle trajectory's future information, limiting itself to a tiny portion of the available time. The ICS-Avoid and ICS^e-Test showed similar performance. However, the calculation time needed to check the state for ICS is the primary benefit of ICS-Test over ICS-Avoid. As the number of obstacles increases, the number of trajectories required to test the state for ICS-Avoid increases, as shown in Figure 8. Which in turn increases the computation time of ICS-Avoid to test the state for ICS.

TABLE 1: Average Collision Rate Comparison for different values of τ

Schemes	Collisions on Average		
	τ=1s	τ=3s	τ=5s
TVDW	9	5	3
ICS-Avoid	3.2	0	0
ICS^eTest	3	0	0

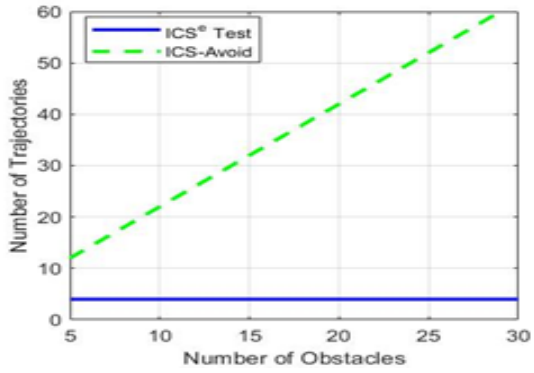

Fig. 8: Number of maneuvers for ICS check vs number of obstacles

In Figure 9, the time taken by ICS^e-Test and ICS-Avoid to compute where the state s is ICS or not as the number of obstacles increases is shown. It can be seen that ICS^e-Test computation time is less than ICS-Avoid.

In summary, ICS-Avoid and ICS^e-Test have better performance relative to TVDW when compared in terms of the number of collisions on average. However,ICS^e-Test has an edge over ICS-Avoid as it

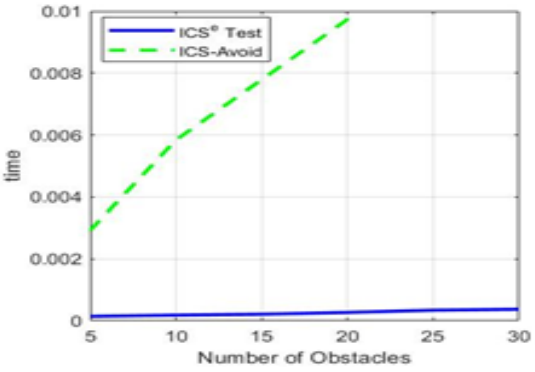

Fig. 9: Average time taken to compute ICS for a given state as the number of obstacles varies

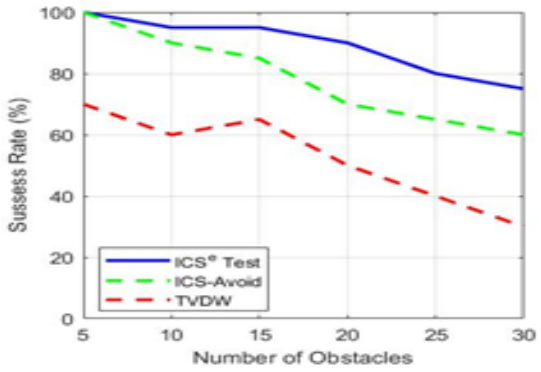

Fig. 10: Success rate with varying numbers of moving obstacles

takes less time to compute whether a given state is an ICS or not. It is a big advantage when used in real-time collision avoidance schemes, as shown in the next simulated experiment's result.

7.2.2 *Experiment 02 – Navigation towards the goal*

The chance that the obstacle may alter its trajectory in less than a second was now set at 0.05. For all three schemes, the future trajectory was made available for 3.5 sec. Each scheme is employed in a cycle plan act cycle to navigate the robot towards the goal. 20 Trials have been done. Figure 10 shows the success rate, which is the proportion of trials in which the robot successfully navigates without colliding from the start point to the target point.

It can be seen in Figure 10 that TWDW has the lowest success rate. The success rate of the navigation approach using ICS-Avoid is less than that which uses ICS^e-Test. This is because when an obstacle changes its path suddenly, the ICS-Avoid-based approach takes a long time to respond due to the large computation time required for computing ICS.

The above experiments validate that the proposed Inevitable state checker (ICS^e-Test), utilizing extremal trajectories, outperforms the state-of-the-art schemes for preventing collisions in real-time in dynamic situations, where the obstacles change their future trajectories at will, and quick replanning is required.

8 Conclusion

This paper has presented a Collision Avoidance scheme based on the proposed Inevitable Collision State-Checker (ICS^e-test) using Extremal Trajectories. The concept of Velocity is extended to compute ICS. It is shown that the four extremal trajectories can be used to check the state for ICS. Due to the fixed number of maneuvers, the computation required for this check was low. The comparative studies validate that the proposed collision avoidance approach for dynamic environments has a high success rate when compared to the state of the art.

References

[1] E. Prassler, J. Scholz, and P. Fiorini, "A robotics wheelchair for crowded public environment," IEEE Robotics & Automation Magazine, vol. 8, no. 1, pp. 38–45, 2001.

[2] R. C. Simpson, "Smart wheelchairs: A literature review," J. Rehabil. Res. Dev., vol. 42, no. 4, 2005.

[3] J. Lee et al., "ODS-Bot: Mobile robot navigation for outdoor delivery services," IEEE Access, vol. 10, pp. 107250–107258, 2022.

[4] S. Bouraine, T. Fraichard, and H. Salhi, "Provably safe navigation for mobile robots with limited field-of-views in dynamic environments," Auton. Robots, vol. 32, pp. 267–283, 2012.

[5] M. Otte and E. Frazzoli, "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning," Int. J. Robot. Res., vol. 35, no. 7, pp. 797–822, 2016.

[6] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2005, pp. 2210–2215.

[7] G. Droge and M. Egerstedt, "Adaptive time horizon optimization in model predictive control," in Proc. Amer. Control Conf., 2011, pp. 1843–1848.

[8] S. Kim et al., "BRVO: Predicting pedestrian trajectories using velocity-space reasoning," Int. J. Robot. Res., vol. 34, no. 2, pp. 201–217, 2015.

[9] A. Bera et al., "GLMP: Realtime pedestrian path prediction using global and local movement patterns," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2016, pp. 5528–5535.

[10] S. Zamboni et al., "Pedestrian trajectory prediction with convolutional neural networks," Pattern Recognit., vol. 121, p. 108252, 2022.
[

[11] T. Fraichard and H. Asama, "Inevitable collision states—A step towards safer robots?," Adv. Robot., vol. 18, no. 10, pp. 1001–1024, 2004.

[12] N. Chan, J. Kuffner, and M. Zucker, "Improved motion planning speed and safety using regions of inevitable collision," in Proc. 17th CISM-IFToMM Symp. Robot Design, Dyn., Control, 2008, pp. 103–114.

[13] R. Parthasarathi and T. Fraichard, "An inevitable collision state-checker for a car-like vehicle," in Proc. IEEE Int. Conf. Robot. Autom., 2007, pp. 3068–3073.

[14] D. Wilkie, J. Van Den Berg, and D. Manocha, "Generalized velocity obstacles," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2009, pp. 5573–5578.

[15] K. M. Zuhaib et al., "Collision avoidance from multiple passive agents with partially predictable behavior," Appl. Sci., vol. 7, no. 9, p. 903, 2017.

[16] D. Wilkie, J. Van Den Berg, and D. Manocha, "Generalized velocity obstacles," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2009, pp. 5573–5578.

[17] K. M. Zuhaib et al., "Collision avoidance of a kinodynamically constrained system from passive agents," Eng. Technol. Appl. Sci. Res., vol. 11, no. 1, pp. 6760–6765, 2021.

[18] L. Martinez-Gomez and T. Fraichard, "An efficient and generic 2D inevitable collision state-checker," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2008, pp. 234–241.

[19] L. Martinez-Gomez and T. Fraichard, "Collision avoidance in dynamic environments: An ICS-based solution and its comparative evaluation," in Proc. IEEE Int. Conf. Robot. Autom., 2009, pp. 100–105.
[20] S. Bouraine, T. Fraichard, O. Azouaoui, and H. Salhi, "Passively safe partial motion planning for mobile robots with limited field-of-views in unknown dynamic environments," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2014, pp. 3576–3582.
[21] M. A. Bouguerra, T. Fraichard, and M. Fezari, "Safe motion using viability kernels," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2015, pp. 3259–3264.
[22] R. Parthasarathi and T. Fraichard, "An inevitable collision state-checker for a car-like vehicle," in Proc. IEEE Int. Conf. Robot. Autom., 2007, pp. 3068–3073.
[23] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," Int. J. Robot. Res., vol. 17, no. 7, pp. 760–772, 1998.
[24] Z. Shiller, F. Large, and S. Sekhavat, "Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories," in Proc. IEEE Int. Conf. Robot. Autom., 2001, vol. 4, pp. 3716–3721.
[25] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in Robotics Research: The 14th Int. Symp. ISRR, Berlin, Heidelberg: Springer, 2011, pp. 3–19.
[26] A. E. Bryson, Applied Optimal Control: Optimization, Estimation and Control, Routledge, 2018.
[27] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," in Proc. IEEE Int. Conf. Robot. Autom., 2007, pp. 1986–1991.