A Multi-objective Optimal Trajectory Planning for Autonomous Vehicles Using Dragonfly Algorithm

R. Syama and C. Mala

Abstract Trajectory planning is considered as a major challenge in autonomous driving, which faces significant issues like safety and efficiency**.** This chapter proposes a novel swarm intelligence meta-heuristic optimization algorithm, called dragonfly algorithm, for the lane-change behaviour in the navigation of autonomous vehicles. A multi-objective lane-changing trajectory planning method has been proposed to optimize the trajectory and avoid collision, which mimics the dynamic and static swarm behaviours of the natural dragonflies. Whenever the autonomous vehicle senses an obstacle, automatically the lane-change manoeuvre should take place. The feasibility and the effectiveness of the algorithm are verified by simulation results using the lane-change data from the benchmark NGSIM dataset. Simulation results show that the proposed algorithm for trajectory planning gives an optimal path for lane-change scenario considering both static and dynamic obstacles.

Keywords Autonomous driving · Trajectory planning · Dragonfly algorithm · Lane-change manoeuvre

1 Introduction

Autonomous driving refers to self-driving vehicles that sense the environment and make decisions by their own without the involvement of a human driver. Autonomous driving has been one of the keen areas of research for the past few decades due to its ability to enhance the safety and efficiency of transportation system.

R. Syama (✉) · C. Mala
Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli, Tamil Nadu, India
e-mail: 406119007@nitt.edu; mala@nitt.edu

A. Haldorai et al. (eds.), *3rd EAI International Conference on Big Data Innovation for Sustainable Cognitive Computing*, EAI/Springer Innovations in Communication and Computing, https://doi.org/10.1007/978-3-030-78750-9_2

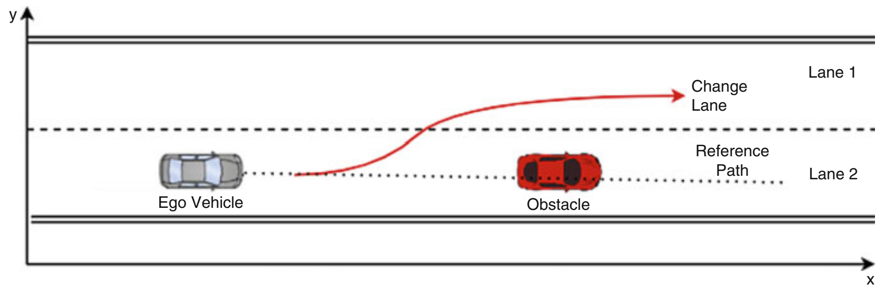


Fig. 1 Lane-change scenario

Trajectory planning has a critical role in autonomous driving technology, and it is one of the major challenges that must be addressed. In the real-world scenario, the autonomous vehicles must navigate autonomously and take intelligent decisions. The vehicle must determine a path for travel from the data it collected from the static and dynamic environments. Then, the trajectory planning is done to obtain an optimal path for the smooth movement from the source to destination achieving certain constraints. It is really important and hard to generate a path that is optimal at the same time meet the real-time requirements. Path planning involves searching and computation of optimal collision-free paths. The feasible path of the autonomous vehicle is generated considering the vehicle geometry, its surroundings, kinematic controls, etc.

The central idea of this chapter is to design a trajectory planning technique that can automatically generate an optimal path from the start to the goal position. Figure 1 shows the lane-changing process by autonomous vehicle, say ego vehicle, while avoiding obstacle. The obstacle avoidance trajectory must consider the safety criteria, mainly the longitudinal and lateral distances with the obstacles. To generate the optimal path, this chapter proposes a meta-heuristic Dragonfly Algorithm (DA). It is based on swarming behaviours of the natural dragonflies [1, 2]. Several researches have been done in trajectory planning for autonomous vehicles, and most of them are based on the searching techniques [3, 4]. But, these techniques have failed in considering obstacle avoidance behaviour.

2 Related Works

The lane-changing behaviour is essential to perform different driving activities such as road merging, entry and exit to a highway, vehicle overtaking, etc. [2]. Several studies have been done to meet the trajectory planning problem for the lane changing in different areas. The major challenge in generating a trajectory is to ensure the feasibility while considering the different constraints. Different optimization algorithms have been applied to trajectory planning problem in recent years. The commonly used algorithms are Ant Colony Optimization [5], Genetic Algorithms [6], Particle Swarm Optimization [7], etc.

In global planning approach, a path is generated from the initial position to the end position from the prior information about the environment. Several methods have been used for global planning such as A* algorithm [8], RRT algorithm [9] and Dijkstra's algorithm [10]. Major drawback of these algorithms is that the whole trajectory calculation process is very time consuming. Therefore, these algorithms are not suitable for trajectory planning of real-time applications such as autonomous vehicles where the obstacles are mainly dynamic.

Local trajectory planning techniques are widely used in most of the real-time applications. These approaches calculate the trajectories for a limited time window, which considers the environment conditions. Several geometric algorithms are used for the local trajectory planning such as splines [11], Bezier curves [12], Clothoid curves [13, 14], etc. These methods generate smooth trajectories by connecting the waypoints for the navigation of the vehicles. But, these methods also suffer from long computation time. Sigmoid curve–based approach is used in [15] for local trajectory planning for obstacle avoidance behaviour. The safe space between the autonomous vehicle and the obstacles is also taken into consideration.

A collision-free trajectory planning method using B spline and RRT-based method is used to improve the robustness of motion planning in autonomous vehicles [16]. Still there exists a timeout possibility. A non-linear Model Predictive Control approach for vehicle navigation is suggested, which considers collision-free trajectory [13]. It generates a dynamic obstacle avoiding trajectory planning with certain constraints. It failed to study the random movement of moving obstacles. The real-time motion planner is proposed in [17, 18]. A number of vision-based approaches are also applied to path planning problems [19, 20].

In this chapter, Dragonfly algorithm is proposed for optimizing the vehicle trajectory. It is a new approach that has powerful capabilities to solve different optimization problems. Studies have shown that DA performs better compared to PSO and gains several multimodal test functions [21]. This approach has several advantages, which makes it suitable for real-world applications:

1. Convergence is guaranteed during optimization since the weights are changed adaptively.
2. It always converges to the global optimum.
3. Randomness can be added to the algorithm.

3 Proposed Method

This chapter proposes a trajectory planning method that generates an optimal collision avoidance trajectory for the autonomous vehicle using Dragonfly Algorithm (DA). Dragonfly algorithm is an optimization technique, which was proposed by Seyedali Mirjalili in 2015. It is based on the static and dynamic grouping behaviour

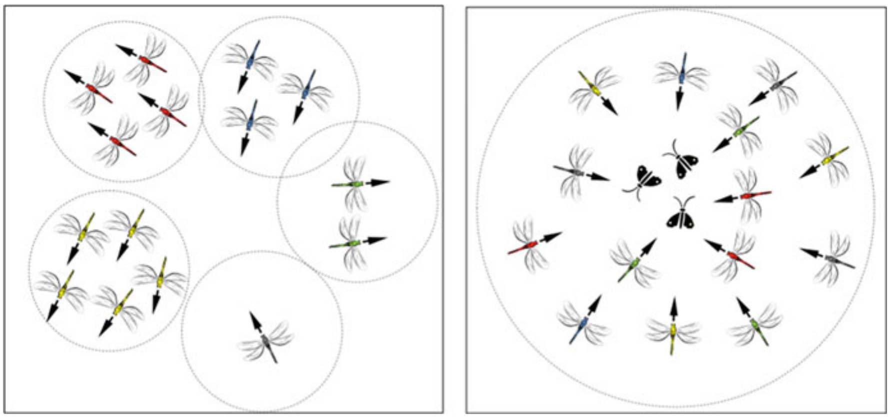


Fig. 2 Dynamic and static dragonfly swarms

of biological dragonflies, which mimics the different phases of optimization and can be employed to solve a wide range of optimization problems [21]. Dragonflies are small insects that hunt and eat other small creatures like butterflies, bees etc. [22]. Dragonflies have a unique swarming behaviour. They form groups only for two reasons: hunting and migration [23]. The swarm formed for hunting is known as static (feeding) swarm and the swarm formed for relocation is known as dynamic swarm. The static and dynamic swarms [21] are illustrated in Fig. 2.

In static swarm, small groups of dragonflies move forward and backward to hunt other flying insects. A swarm is dynamic where a huge number of dragonflies are grouped to move in a single direction over a long distance. These dynamic and static swarms compose the exploitation and exploration stages of DA. These two behaviours are in accordance with the meta-heuristic stages [21].

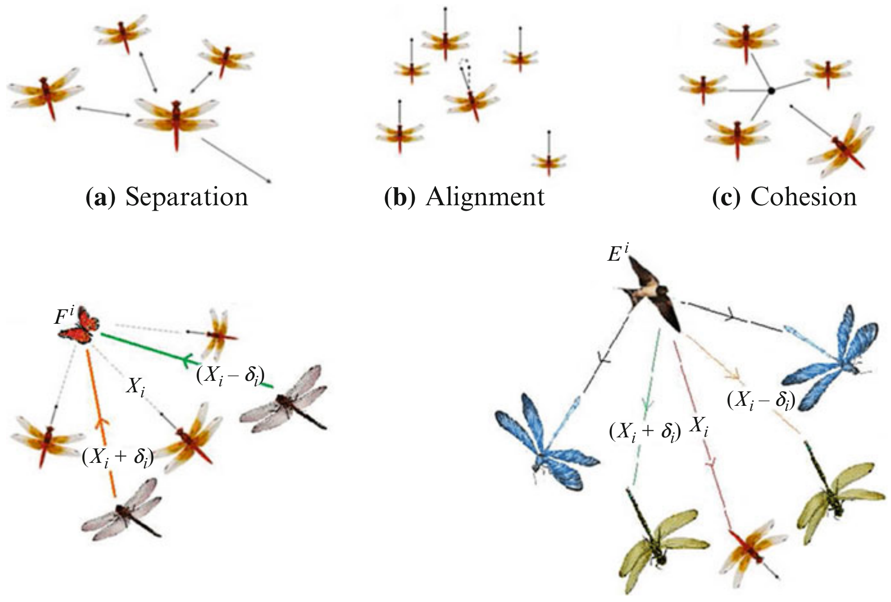
The dragonfly individuals exhibit five primitive behaviours which can be used to model the swarm behaviour. The dragonflies exhibit five properties, which are shown in Fig. 3 [21]. Each of the behaviour of the dragon flies are modelled as follows [21]:

1. *Separation (Si)* represents the operation to avoid collisions that the individuals follow with other individuals in the region. It is calculated as [21]:

$$Si = -\sum_{j=1}^{N} X - Xj \qquad (1)$$

where X is the location of the dragonfly, Xj is the location of jth nearest individual, and N is the number of nearby individuals.

2. *Alignment* (*Ai*) represents the dragonfly's velocity which matches with the other nearby dragonflies of the same group. It is calculated as [21]:



(a) Separation **(b)** Alignment **(c)** Cohesion

(d) Modified steps for Attraction towards the food **(e)** Modified steps for Distraction from the enemy

Fig. 3 Behaviour of dragonflies

$$Ai = \frac{\sum_{j=1}^{N} Vj}{N} \tag{2}$$

where V_j represents the velocity of the jth dragonfly.

3. Cohesion (Ci) is the tendency of the individual to move towards the centre of the group. It can be calculated as [5]:

$$Ci = \frac{\sum_{j=1}^{N} Xj}{N} - X \tag{3}$$

4. Attraction (Fi) represents the attraction towards food source and can be represented as [5] :

$$Fi = X^{+} - X \tag{4}$$

where F_i represents the food source of the ith individual.

5. Distraction (Ei) is the distraction from enemies and is represented as [5]:

$$Ei = X^{-} - X \tag{5}$$

where E_i denotes the location of the enemy of the ith dragonfly

The step vector ΔX and the position X can be used to update the location of the dragonflies in the search domain. The update is done as follows [21]:

$$\Delta X_i^{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + \omega \Delta X_i^t \tag{6}$$

where s, a and c represent the weights for separation, alignment and cohesion, respectively. F is the food factor, e is the enemy factor, ω is the inertia weight and t is the iteration vector. The location of the ith dragonfly at $(t + 1)$ is revised as [21]:

$$X_i^{t+1} = X_i^t + \Delta X_i^{t+1} \tag{7}$$

The radius of the neighbouring space increases as the algorithm progresses. If the dragonfly has no neighbours, then the position and velocity are updated using Levy flight technique, which is a variation of the random walk process to apply randomness to the position and velocity of dragonflies. The update can be performed as [5]:

$$X_i^{t+1} = X_i^t + \text{levy}(d) \times X_i^t \tag{8}$$

3.1 Modelling of Objective Functions for Path Planning

The main objective considered in this chapter is to navigate the autonomous vehicle in a road with different obstacles. The vehicle should sense the surroundings for obstacles and avoid them while reaching the destination with optimal smooth trajectory.

The problem of trajectory planning is considered as a minimization problem and two objective functions are considered in this chapter. The first function helps the vehicle to traverse to the destination avoiding the obstacles and second enables it to obtain a short smooth trajectory. The algorithm chooses the best path from the several paths for the vehicle to travel.

When the vehicle reaches the obstacle, the sensors detect it and the range of the obstacle is calculated. Dragonfly algorithm is then initiated to avoid the collision. It will find the best next position avoiding the obstacle while reaching the goal. The food source gives the optimal path for the ego vehicle to move. The position of the ego vehicle is updated as Eq. (6). The best approximations are stored and retrieved by an archive. The food source of the dragonfly is chosen from the archive.

Therefore, the food source is always a good candidate solution. Therefore, the quality of the food source is proportional to the optimal path length. The next position of the ego vehicle always depends on the distance between food source and goal and the obstacle. The important factors considered here are obstacle avoidance behaviour and goal-finding behaviour.

The collision avoidance is modelled using obstacle avoidance behaviour. The position of the food source is selected as the global best path, which satisfies the safety criteria of the system by maintaining the maximum distance between the ego vehicle and the obstacle. The objective function can be calculated as the Euclidean distance between the best position and the obstacle in the environment [23].

$$\text{Distance}_{obj-F} = \sqrt{(X_{ob} - X_{Fi})^2 + (Y_{ob} - Y_{Fi})^2} \tag{9}$$

where (X_{ob}, Y_{ob}) gives the position of the obstacle and (X_{Fi}, Y_{Fi}) gives the location of the food source.

The nearest obstacle to the vehicle is calculated as [23]:

$$\text{Distance}_{Obj-V} = \sqrt{(X_{ob} - X_{V})^2 + (Y_{ob} - Y_{V})^2} \tag{10}$$

The food source must be kept at the minimum distance from the goal. The food source gives the global best position. The goal-finding behaviour can be calculated as the Euclidean distance between the food source and the goal. It can be represented as [23]:

$$\text{Distance}_{G-F} = \sqrt{(X_{G} - X_{Fi})^2 + (Y_{G} - Y_{Fi})^2} \tag{11}$$

These two behaviours can be combined to find the objective function of the system. It can be represented as [23] :

$$\text{Objective function} = C1.1/\,(\min\,(\text{Distance}_{OB-F})) + c2.\text{Distance}_{G-F} \tag{12}$$

where $C1$ and $C2$ are the fitting parameters, and they have a huge influence on the optimal trajectory.

3.2 Algorithm

The pseudo-code for the trajectory planning is given in Algorithm 1.

Algorithm 1 TP_Dragonfly

```
Input : Dragonfly Population P, Number of Dragonflies N
Output : Optimal Path Coordinates

1 : Initialize the starting and goal positions of the autonomous vehicle.
2 : Follow the reference trajectory until an obstacle is sensed
3 : if vehicle senses obstacles, start dragonfly algorithm
4 : Initialize the population of dragonflies randomly $P_i$ (1,2,…N)
5 : Set the step vectors $\Delta X_i$ (1,2,..m)
6 : **while** (t < Maximum number of iterations) **do**
7       Calculate the fitness of each dragonfly $f(X_i)$
8       Update the location of the food source and enemy
9       Update the weights for inertia $\omega$ ,separation s, alignment a, cohesion c, food factor f and enemy factor e
10      Compute S, A, C, F, and E using (1) – (5)
11      Update neighbor radius, velocity vector, position vector using (6) – (8)
12      Move the vehicle to the global best position
13  **end while**
```

4 Simulation Results

The proposed TP_Dragonfly algorithm for trajectory planning is simulated using Matlab. The lane-changing scenario is evaluated using the real data from the NGSIM (Next Generation Simulation) dataset. Next Generation Simulation (NGSIM) program is the project by US Federal Highway Administration and is collected by detailed vehicle trajectory data on southbound US 101 and Lankershim Boulevard in Los Angeles. The data from the NGSIM dataset provides real traffic information such as vehicle velocity, position, acceleration, lane, etc. This data is used for studying the features of lane-changing process and for validating the lane-changing models.

In the simulation, the positions, velocities and accelerations of ego vehicle and other vehicles including the obstacles are obtained from NGSIM dataset, and the optimal trajectory is generated by the proposed model. For each time step, the proposed method plans the trajectory of the ego vehicle dynamically. Figure 4 shows the trajectory planning using the proposed TP_Dragonfly algorithm.

This proposed method can be applied for both static and dynamic obstacles. When the ego vehicle senses a static obstacle, it starts changing the lane considering the safety criteria. But for the dynamic obstacles, when the speed of the ego vehicle is greater than the speed of the dynamic obstacle in front, then the ego vehicle starts the lane change. The ego vehicle follows the reference path until it senses the obstacle. When the vehicle detects the obstacle, then the TP_Dragonfly algorithm is

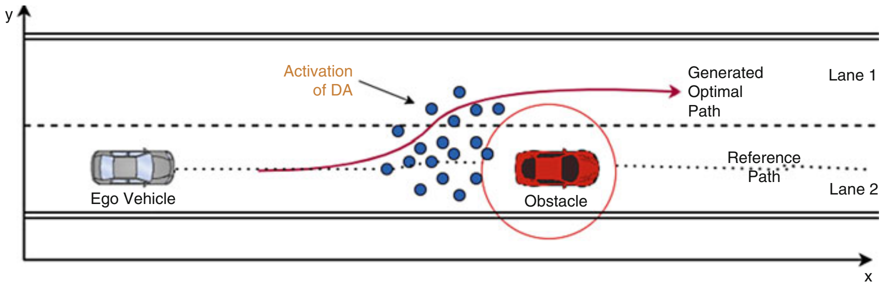


Fig. 4 Schematic diagram showing activation of TP_Dragonfly Algorithm

Table 1 Parameter values selected for TP_Dragonfly

Sl No	Symbol	Value
T	Max. Number of Iterations	430
N	Population Size	100
S	Separation	0.1
A	Alignment	0.1
C	Cohesion	0.7
F	Food	1
E	Enemy	1
C1	Fitting Parameter 1	1
C2	Fitting Parameter 2	1×10^{-4}
r1, r2	Random Numbers	[0,1]

activated to find an optimal path that avoids the obstacle. TP_Dragonfly algorithm calculates the next best position based on the objective functions and move forward avoiding the obstacle and reaches the destination.

The parameters used for simulation are shown in Table 1.

The efficiency of the proposed TP_Dragonfly algorithm relies on the accuracy of approximations of the parameters.

The proposed method can be validated using different lane-changing behaviours existing in the real world. The overtaking behaviour can be shown from the data given in the NGSIM dataset. This NGSIM data can be used to verify the efficiency of the proposed approach.

When the ego vehicle senses a static obstacle, it changes the lane to avoid collision. It initiates TP_Dragonfly algorithm to obtain the optimal path. Figure 5a and b shows the lane-changing behaviour of ego vehicle when a static obstacle is encountered.

When the speed of the ego vehicle is higher than that of the vehicle in front, the ego vehicle slowly changes the lane just before it approaches the vehicle in front. This vehicle is identified as a dynamic obstacle by the ego vehicle sensors, and it starts choosing a lane-change process by initiating TP_Dragonfly algorithm

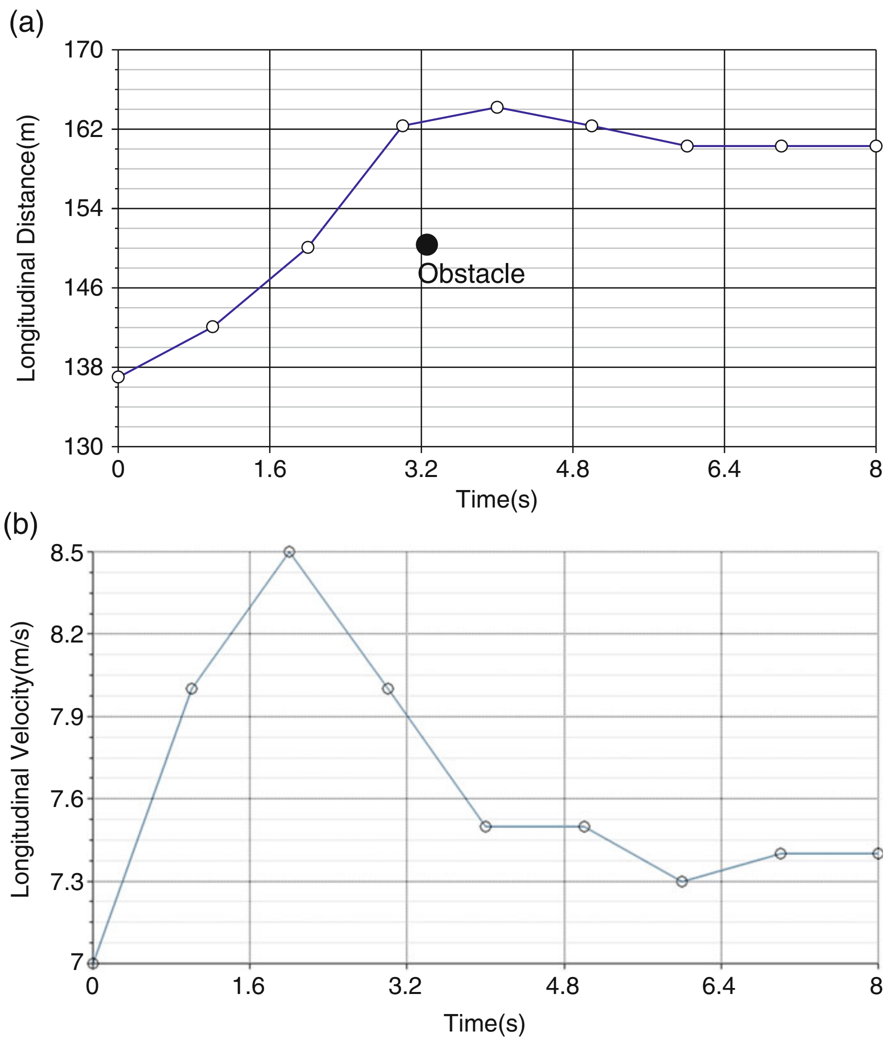


Fig. 5 Lane-change scenario for static obstacle

to generate an optimal path. Figure 6a shows the lane-changing behaviour of ego vehicle when it senses a dynamic obstacle.

Figure 6b shows the overtake scenario for the vehicle from the NGSIM dataset. The change of the speed of ego vehicle shows that the lane change adjusts the speed dynamically to adapt the change in the velocity of other vehicles.

The lateral positions and lateral velocity of the ego vehicle while changing the lane when sensing a dynamic obstacle are shown in Figs. 6c and d.

Figure 7 shows the lane-change trajectories generated by the proposed method and by the NGSIM data for the vehicle id 456 in NGSIM data. The vehicle detects a

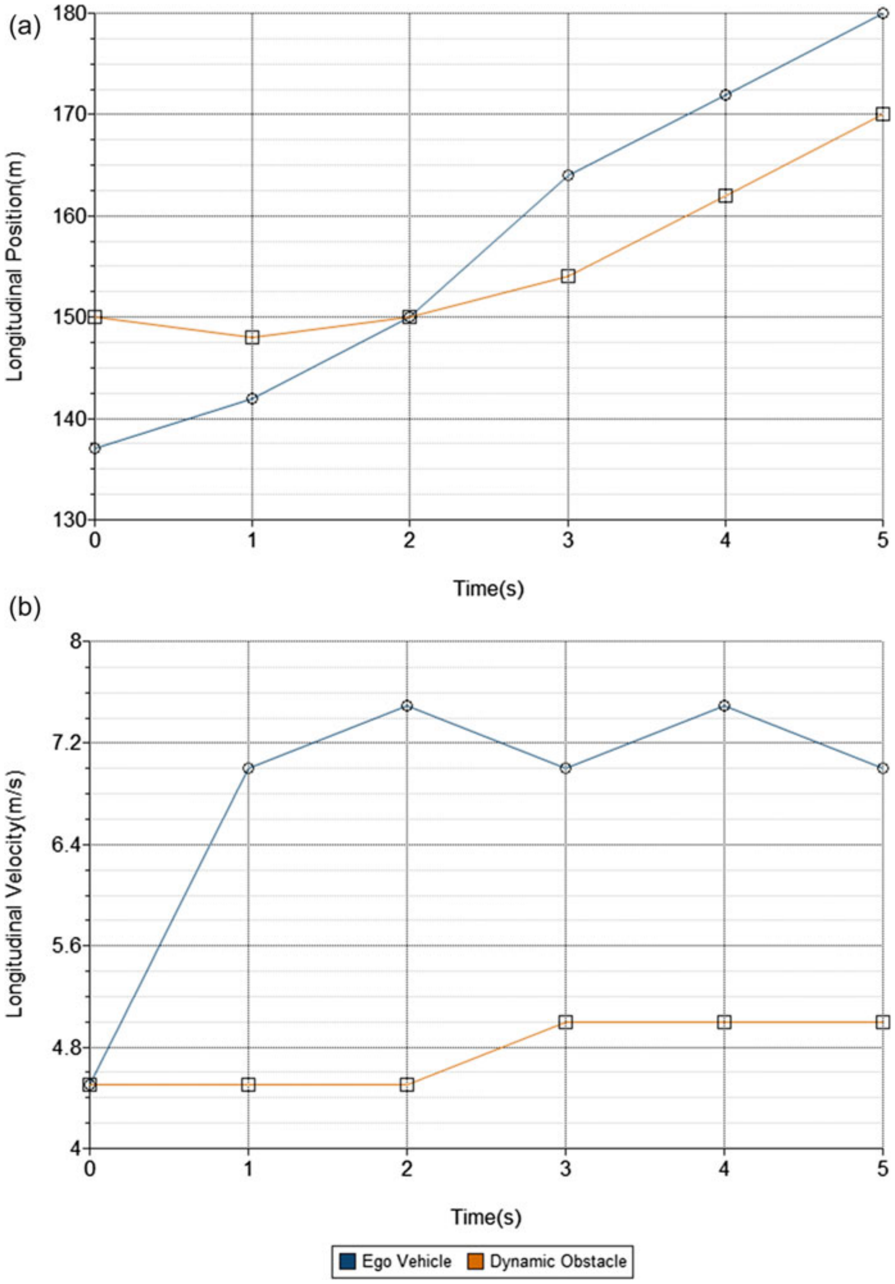
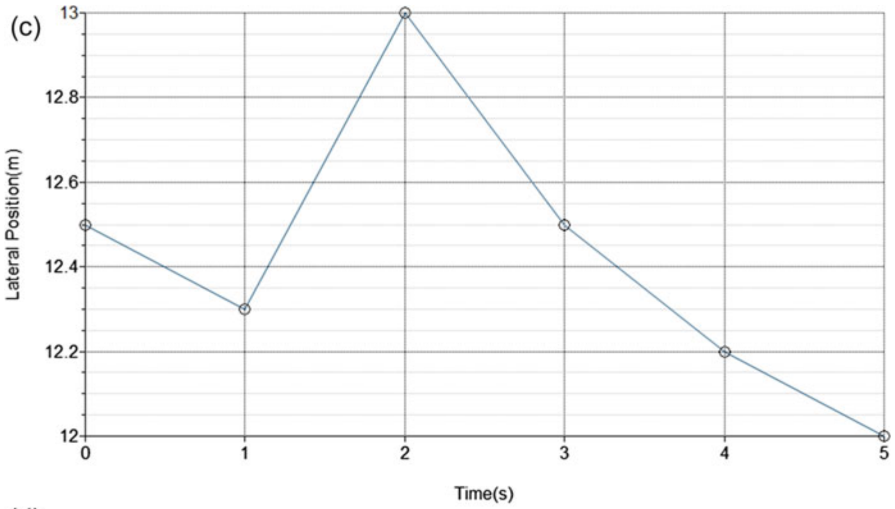


Fig. 6 (**a**) Lane-change scenario for dynamic obstacle. (**b**) Lane-changing scenario for dynamic obstacle. (**c**) Lane-change scenario for dynamic obstacle. (**d**) Lane-change scenario for dynamic obstacle



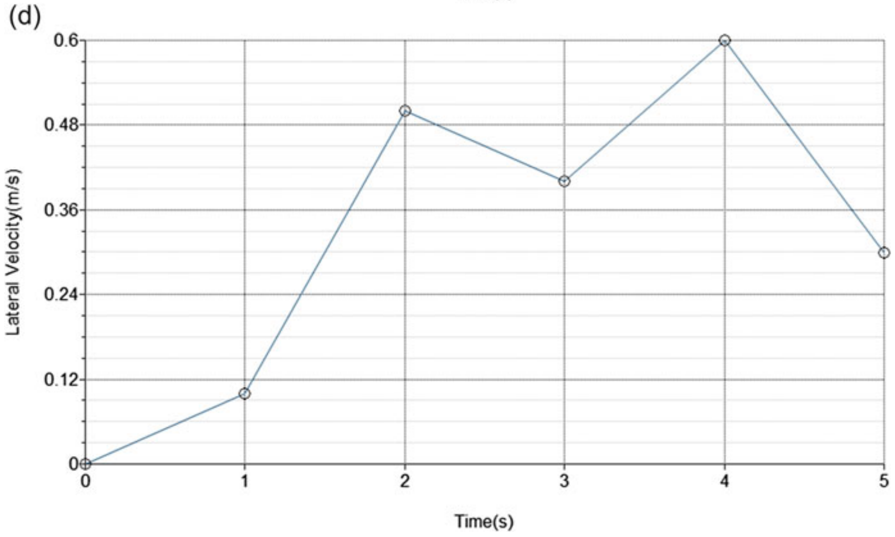


Fig. 6 (continued)

static obstacle, and it decides to change the lane. It is clear that the proposed method generates the optimal path.

The Fig. 8 compares the lane-change trajectories for vehicle taken from the NGSIM data for the vehicle id 466 in NGSIM data, which changes the lane as a result of the presence of moving obstacle. The ego vehicle moves at a speed of 60 km/h, and the obstacle vehicle moves at a speed of 45 km/h. The trajectory generated by the proposed method shows that it outperforms the real trajectory.

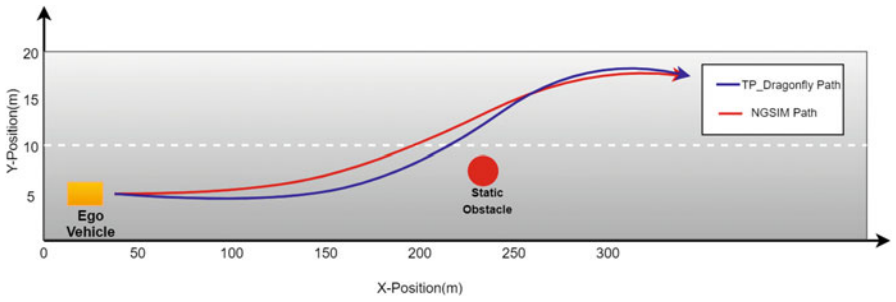


Fig. 7 Comparison of the trajectories generated for static obstacles

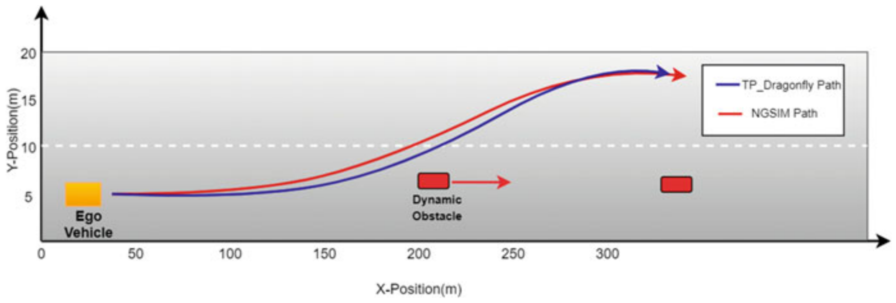


Fig. 8 Comparison of the trajectories generated for dynamic obstacles

The results show that the lane-change behaviour while sensing an obstacle in the current lane works well for the proposed TP_Dragonfly algorithm, which generates the optimal trajectory for lane change.

4.1 Influence of the Fitting Parameters on the Trajectory

Two fitting parameters C1 and C2 are used for the objective functions, and they have direct influence on the trajectory of the ego vehicle. The larger C1 value makes the vehicle move away from the obstacle and a smaller value makes it collide with the obstacle. The fitting parameter C2 decides the probability of the vehicle to reach the goal in optimal path. If C2 is large, there is high probability that the vehicle reaches the goal through an optimal path else it generates larger paths. The fitting parameters decide the convergence of the objective function, and the optimal values of these parameters eliminate the local minima. In this chapter, the fitting parameters are selected by trial and error. Figures 9 and 10 show the effect of C1 and C2 on the generated trajectory.

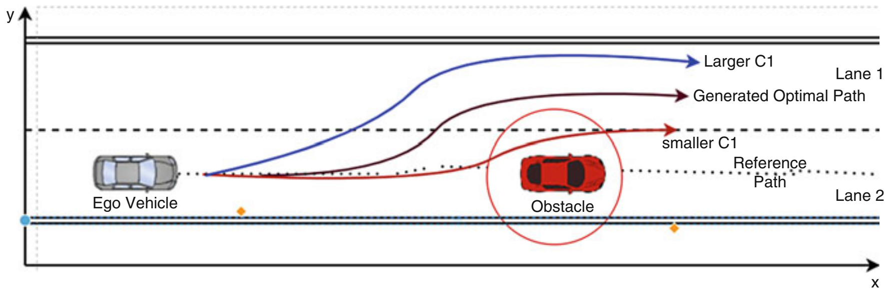


Fig. 9 Effect on fitting parameter C1 on trajectory

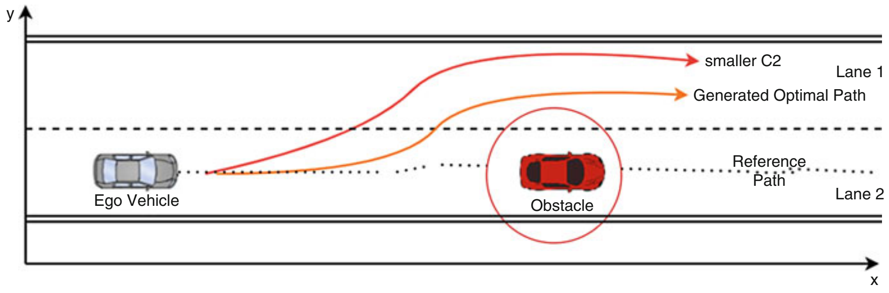


Fig. 10 Effect on fitting parameter C2 on trajectory

5 Conclusion

This chapter proposes an optimal trajectory planning technique for autonomous vehicles using a bio-inspired Dragonfly algorithm. The lane-change behaviour of the autonomous vehicles on approaching an obstacle is studied, and an optimal trajectory is generated for navigation to the goal for both static and dynamic obstacles. The simulation results show that this approach is feasible in the generation of optimal path for real-time data. The influence of fitting parameters on the trajectory is also investigated.

References

1. Meraihi, Y., Ramdane-Cherif, A., Acheli, D., & Mahseur, M. (2020). Dragonfly algorithm: a comprehensive review and applications. *Neural Computing and Applications, 32*, 16625.
2. Laghmara, H., et al. (2019). Obstacle Avoidance, Path Planning and Control for Autonomous Vehicles. In *IEEE Intelligent Vehicles Symposium (IV)* (pp. 529–534). Paris, France: IEEE.
3. Manipriya, S., Mala, C., & Mathew, S. (2020). *Significance of Real Time Systems in Intelligent Transportation Systems, Handling Priority Inversion in Time-Constrained Distributed Databases* (pp. 61–85). IGI Global Publications.

4. Manipriya, S., Mala, C., & Mathew, S. (2020). A Collaborative Framework for Traffic Information in Vehicular Adhoc Network Applications. *Journal of Internet Services and Information Security, 10*(3), 93.
5. Wang, Y., Lu, X., & Zuo, Z. (2019). Autonomous Vehicles Path Planning With Enhanced Ant Colony optimization. In *2019 Chinese Control Conference (CCC), Guangzhou* (pp. 6633–6638). Technical Committee on Control Theory, Chinese Association of Automation.
6. Nazarahari, M., Khanmirza, E., & Doostie, S. (2019). Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Systems with Applications, 115*, 106–120.
7. Mandava, R. K., Bondada, S., & Vundavilli, P. R. (2019). An Optimized Path Planning for the Mobile Robot Using Potential Field Method and PSO Algorithm. In *Advances in Intelligent Systems and Computing* (Vol. 817). Singapore: Springer.
8. Panda, M., Das, B., Subudhi, B., et al. (2020). A Comprehensive Review of Path Planning Algorithms for Autonomous Underwater Vehicles. *International Journal of Automation and Computing, 17*, 321–352.
9. Xue, J., Kawabata, K., Zhu, J., Ma, C., & Zheng, N. (2014). A Fast RRT Algorithm for Motion Planning of Autonomous Road Vehicles. In *17th IEEE International Conference on Intelligent Transportation Systems*. ITSC.
10. Liu, Q., Xu, H., Wang, L., Chen, J., Li, Y., & Xu, L. (2020). Application of Dijkstra Algorithm in Path Planning for Geomagnetic Navigation. In *IEEE 11th Sensor Array and Multichannel Signal Processing Workshop (SAM)* (pp. 1–4). Hangzhou, China: IEEE.
11. González, D., Pérez, J., Lattarulo, R., Milanés, V., & Nashashibi, F. (2014). Continuous curvature planning with obstacle avoidance capabilities in urban scenarios. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (pp. 1430–1435). Qingdao: IEEE.
12. Daniel, J., Birouche, A., Lauffenburger, J.-P., & Basset, M. (2011). Navigation-based Constrained Trajectory Generation for Advanced Driver Assistance Systems. *International Journal of Vehicle Autonomous Systems (IJVAS), 9*, 269–296.
13. Hundelshausen, F., Himmelsbach, M., Hecker, F., Müller, A., & Wuensche, H.-J. (2009). Driving with Tentacles - Integral Structures for Sensing and Motion. In *The DARPA Urban Challenge* (pp. 393–440). Springer.
14. Mouhagir, H., Cherfaoui, V., Talj, R., Aioun, F., & Guillemard, F. (2017). Using evidential occupancy grid for vehicle trajectory planning under uncertainty with tentacles. In *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)* (pp. 1–7). Yokohama: IEEE.
15. Glaser, S., Vanholme, B., Mammar, S., Gruyer, D., & Nouvelière, L. (2020). Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction. *IEEE Transactions on Intelligent Transportation Systems, 11*(3), 589–606.
16. Zeng, D., et al. (2019). Novel Robust Lane Change Trajectory Planning Method for Autonomous Vehicle. In *IEEE Intelligent Vehicles Symposium (IV)* (pp. 486–493). Paris, France: IEEE.
17. Zhang, C., Chu, D., Liu, S., Deng, Z., Wu, C., & Su, X. (2019). Trajectory Planning and Tracking for Autonomous Vehicle Based on State Lattice and Model Predictive Control. *IEEE Intelligent Transportation Systems Magazine, 11*(2), 29–40.
18. Dolgov, D., Thrun, S., Montemerlo, M., & Diebel, J. (2008). Practical search techniques in path planning for autonomous driving. In *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08), (Chicago, USA)*. AAAI.
19. Zucker, M., Bagnell, J., Atkeson, C., & Kuffner, J. (2010). An optimization approach to rough terrain locomotion, Robotics and Automation (ICRA). In *IEEE International Conference* (pp. 3589–3595). IEEE.
20. Ben-Messaoud, W., Basset, M., Lauffenburger, J., & Orjuela, R. (2018). Smooth Obstacle Avoidance Path Planning for Autonomous Vehicles. In *IEEE International Conference on Vehicular Electronics and Safety (ICVES)* (pp. 1–6). Madrid: IEEE.

21. Mirjalili, S. (2016). Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications,**27***, 1053–1073.
22. Mashadi, B., & Majidi, M. (2014). Global optimal path planning of an autonomous vehicle for overtaking a moving obstacle. *Latin American Journal of Solids and Structures, 11*, 2555.
23. Rahman, C. M., & Rashid, T. A. (2020). A survey on dragonfly algorithm and its applications in engineering. *ArXiv abs/2002.12126*.