International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Supporting BIT*-Based Path Planning with MPC-Based Motion Planning of the Self-Driving Car

[1]Ahmed A. Al-Moadhen, [2]Haider G. Kamil, [3]Ali R. Khayeat

Submitted: 12/09/2022 **Accepted:** 24/12/2022

Abstract: This paper presents the enhanced operation of the path planner integrated with a predictive controller for a self-driving vehicle to accomplish trajectory planning and avoid obstacles. The path planner used the Batch Informed Trees (BIT*) planning algorithm approach and the tracking controller is designed based on the model predictive control (MPC). BIT* algorithm is used to find the best path between the start and the goal nodes. Then the MPC tracks the route and controls the vehicle's movement to its destination. Path planning control is vital point in avoiding autonomous car the obstacles during serious traffic scenarios. The MPC controls the main parameters of the vehicle: velocity, acceleration, and orientation. The traditional BIT* operation is enhanced by subjecting the generated trajectory to a basis spline (B-Spline) planner. This enhancement solves the hard angle and manoeuvre presented in the path, improves the trajectory points connections, and then swiftly obtains a collision-free trajectory. In addition, this paper tackles the issues related to avoiding local obstacles and the follow up of dynamic goal points in a complex and dynamic world. The model predictive controller is used to track the enhanced trajectory plan generated by the BIT*planner approach by using the kinematic model of the vehicle. A modal description of the approach for building the graph-search for these cases and displaying simulated and real-world examining data shows this method's practical application. In the simulation, the controller selects the best trajectories as references. Also, it enhances the performance of trajectory planning and ensures that the casual obstacle can be avoided in real-time and the robot can arrive at the final point smoothly. The results of the simulation show a reasonable accomplishment in navigation performance, the planned path is softer, and the efficiency of the search is higher in composite environments and different scenarios. Also, the test shows that the autonomous car can pursue the reference path accurately, even with sharp corners.

Keywords: *sampling-based planning, MPC, self-driving car, BIT*, motion planning*

1 Introduction

Recently, industrial and academic research centres have witnessed rapid evolutions in developing the technologies of self-driving vehicles. The traditional transportation roads will be reshaped in terms of safe, comfortable and intelligent mobility, controlling the motion planning of self-driving cars provided that they perform driving tasks. These cars face some challenges when moving in dynamic environments, such as moving obstacles, controlling the speed, braking, and steering during their journey. Path planning and motion control are important concepts for avoiding collision in autonomous vehicle scenarios. So, it is necessary to support system functionality with a strong real-time planning system.

In the recent decade, the autonomous vehicles have seen essential progress in terms of software and hardware on computing, perception, control and decision-making [1]. By swift expansion in technologies for both control and sensor, advanced driver assistance systems (ADAS) be more robust [2]. Driving autonomously contributes to improving the integrity of traffic, particularly by minimizing the mistakes that may make by drivers [3] [4]. In addition, driving autonomously can ease the fruitful utilisation of transport period for everyday passengers [5]. Path planning is among the different parts that advance Autonomous Vehicles (AVs) because it has the ability to compose one of the core methods.

The vehicle is considered as a non-holonomic and nonlinear system, so this system requires to be controlled in order to preserve the wished-for performance, e.g., driving at a wanted speed, maintaining riders relaxed while bypassing clashes with nearby cars and infrastructure [3]. The autonomous vehicle drives into and out of a sequence of separated points' list between

1Department of Electrical and Electronics Engineering,
College of Engineering - University of Kerbala, Karbala 56000, Iraq.
ahmedh1333@uokerbala.edu.iq
2Department of Computer Engineering Techniques, AlSafwa University College, Karbala 56000, Iraq.
Department of Electrical and Electronics Engineering, College of Engineering-University of Kerbala, Karbala 56000, Iraq.
3Department of Computer Science, College of Computer Science and Information Technology, University of Kerbala, Karbala 56000, Iraq.
Correspondence should be addressed to Ahmed A. Al-Moadhen;
ahmedh1333@uokerbala.edu.iq

the initial and the goal. These points represent a net area and make a whole driving scenery. Finally, vehicles reach the destination point by crossing the sequence of neighbouring networks, which are acquired by the heuristic algorithm.

In navigation, there are two types of algorithms for path planning: global and local. Global trajectory planning is in charge of creating a trajectory from an initial point to a destination point, while local trajectory planning is in charge of avoiding the dynamic obstacles as the robot progresses. The ant colony algorithm [6], Dijkstra algorithm [7], A∗ algorithm [8], and rapidly exploring random tree (RRT) algorithm [9]–[12] and its successor Batch Informed RRT* [13] are global path planning algorithms. There are three types of mobile robots: autonomy, semi-autonomy, and remote control. The autonomous mobile robots can sense, fit in their environment, and make decisions so that they can satisfy a broad range of task demands [14]. The research of making the mobile robot accomplish a pre-decided task autonomously in a composite environment is an active, focused and difficult mobile robot research.

The navigation in a mobile robot can be defined as the operation of a mobile robot finishing pre-planned tasks autonomously. The mobile robot can significantly recognize the orientation of the target movement and finish the navigation task when precisely catches its pose and the pose of obstacles in its world [15]. Yet, the lack of required technologies directed to the immature field of autonomous vehicle. Many challenges need to be solved in order to achieve fully automated vehicles for people. The principles of perceiving the environment, making decisions, planning paths, controlling motion, networking cars, and human with vehicle interaction are the key technologies of self-driving vehicles. Planning the vehicle's path and controlling its motion are the two most essential key technologies. Both are considered core modules and play a vital role in driving safely and comfortably [2]. Many works of literature show the progress in controlling the longitudinal motion of a vehicle by exploiting the PID controller. While controlling the lateral motion of a vehicle is a relative composite problem that requires more solid methods to secure comfort in addition to safety.

The PID controller parameters and these parameters are difficult to determine because of the need for high precision to deal with systems with nonlinearities. The MPC method has been demonstrated as an encouraging scheme to resolve valuable control achievement over the cutting edge of autonomous driving machinery. The MPC controller exploits the vehicle kinematics to anticipate its planned motion states by linking current states and goal states developed by the planner [2].

The essential role of the MPC controller is to keep track of the generated states to enable the vehicle to reach the target with maintaining efficiency [16]. It is vital to boost the efficiency of the energy at the same time [17]. The use of MPC with an autonomous vehicle to avoid the obstacles was presented in [18]. In addition, the autonomous vehicle can be endowed with state-of-the-art functions, like the planning and control techniques both display the driver's behaviour [19]. The driver model based on MPC meditates distinct drivers' skilfulness by regarding the stochastic aspects of drivers' steering traits [20].

In this paper, we present a framework that permits to embody the motion elementary and exterior perceptual stimuli that based on MPC directly into the planning process. So, a standard illustration of the suggested approach of assembling the searching graph in these cases, in addition to showing real-world and simulated experiment data, shows the functional application of this method. The framework integrates a tracking controller and local path planner to support autonomous vehicles. The path planner based on BIT* generates a real-time reference path for the tracking controller based on model predictive control. The path planner is enhanced to be utilised on medium-speed and low-speed metropolitan routes and highways [2].

BIT* improves the trajectory constantly throughout iterations. This paper uses the sampling process of BIT*, and then a B-Spline planner [21], [22] is used to decrease the initial's path cost and further curb the area of sampling to achieve the randomness of sampling adequately, then quickening the convergence speed and smoothing the sharp angles in the path.

This paper tackles avoiding local obstacles and tracking the dynamic destination positions in environments. Therefore, an effective path planning approach that integrates MPC with BIT* algorithm is proposed. Therefore, a BIT* framework that allows the autonomous car to incorporate MPC-based motion actions and the generated trajectory directly into the planning process is presented.

The essential contributions in this paper are: (1) improving the control accuracy, the model inconsistency induced by irregular road circumstances, the sharp curve, and the zigzag path generated is used to simplify the model by adopting a smother approach (B-Spline). This method decreases the cost of the initial trajectory. The enhanced BIT* with the B-spline planner reduces the size of the elliptical shape region to accelerate the converging rate. (2) We use the B-Spline planner to match the global reference trajectory positions in the frame of the robot to get a steady local trajectory reference to enable a robot to track. (3) An implementation scheme of MPC is selected

for trajectory tracking controller and speed generation by regarding obstacles close to a robot.

The rest of this paper is structured as follows: the illustration of the path tracking issue and the design of the controller used is presented in Section 2. The MPC controller is explored in Section 3. The implementation of the MPC controller is discussed in Section 4. Section 5 presents the experimental platform. Section 6 explains the analysis of the results. Section 7 provides the conclusion and suggestions for future works.

Figure 1 shows the block diagram of the proposed framework in this paper [23]. The BIT* planner gets a trajectory according to the initial state, final state and the task. The B-Spline deals with tortuous paths and makes them smoother. Then the MPC tracker locates the best series of controlling v and ω and then sends motion commands to the robot's platform.

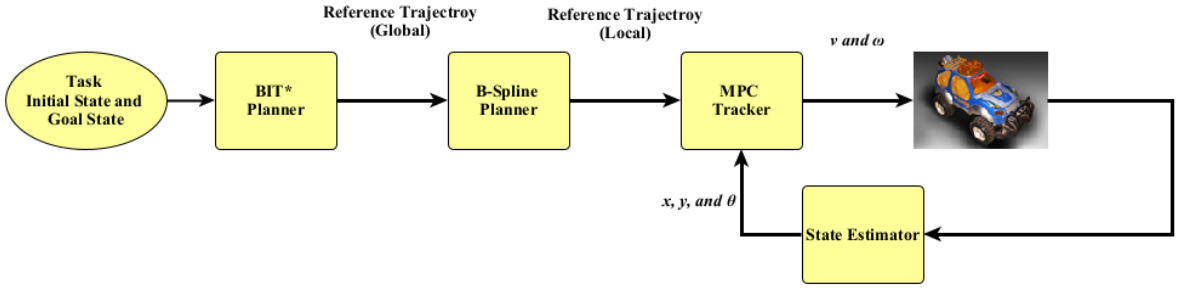

Figure 1: BIT* and MPC Path Planning and Tracking Framework.

2 Path and Motion Planning by the BIT* with MPC

Typically, a graph is constructed by connecting points from the state space that are sampled randomly by using sampling-based algorithms. The regular graph represents a set of collision-free paths, and then these algorithms answer queries by calculating the shortest (best) trajectory, which joins the starting state with a target state through the map. The BIT* is one of the methods of building a graph which is established by applying metric motion for going from the centre point of one cell to the centre point of another.

The methodology in this paper establishes constructing the BIT* by inserting edges that belong to the robot, which executes various controllers available. The resultant structure, a BIT* expanded with MPC, is called BIT*MPC.

2.1 Vehicle Kinematic Model

In this paper, we use the vehicle mathematical model, which is obtained to design a trajectory-tracking controller. We specify the vehicle model by considering kinematics principles in the stationary or robot coordinate system with basis vectors $(\hat{e}_x, \hat{e}_y, \hat{e}_z)$ coordinate system. Besides, the aim of this paper is to how to follow the referenced path quickly and regularly, which corresponds to vehicle handling solidity; hence a basic 'bicycle' model is used.

The vehicle kinematics model is given by Equation 1 [24], [25] [26]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{\tan\delta}{l} \end{bmatrix} \times v \; or \; \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \qquad (1)$$

Where $\mathbf{x} = [x \; y \; \theta]^T$ means the vehicle pose in the global coordinate system. $\mathbf{u} = [v \; \delta]$ means the variables of the control, which are obtained from the outputs of the MPC. The $[x \; y]^T$ vector represents the coordinate that is attached at the centre of the vehicle's back axle. θ is the vehicle heading angle. v represents the longitudinal velocity. δ means the steering angle of a front tire. Finally, l is the rear shaft length of the vehicle. In Figure 2, the following self-driving car equations model can be used [5]

.

$$\dot{x} = v \cos\theta \qquad (2)$$

$$\dot{y} = v \sin\theta \qquad (3)$$

$$\dot{\theta} = \frac{v}{l} \tan\delta \qquad (4)$$

$$\dot{x_f} = v_f \cos(\theta + \delta) \qquad (5)$$

$$\dot{y_f} = v_f \sin(\theta + \delta) \qquad (6)$$

$$\dot{\theta} = \frac{v_f}{l} \tan\delta \qquad (7)$$

$$\frac{v}{v_f} = \cos\delta \tag{8}$$

Where v is the reverse speed, v_f is the front speed.

The steering angle δ, within the limits of the vehicle mechanics, $\delta \in [\delta_{min}, \delta_{max}]$, and the front velocity v within an acceptable range $v \in [v_{min}, v_{max}]$ should be selected appropriately in order to solve the controlling and planning problems.

The heading rate ω is selected instead of steering angle δ in a simplified version of the model. So:

$$\delta = \tan^{-1}\left(\frac{l\omega}{v}\right) \tag{9}$$

Simplifying the heading angle rate to:

$$\dot{\theta} = \omega, \omega \in \left[\frac{v}{l}\tan\delta_{\min}, \frac{v}{l}\tan\delta_{\max}\right] \tag{10}$$

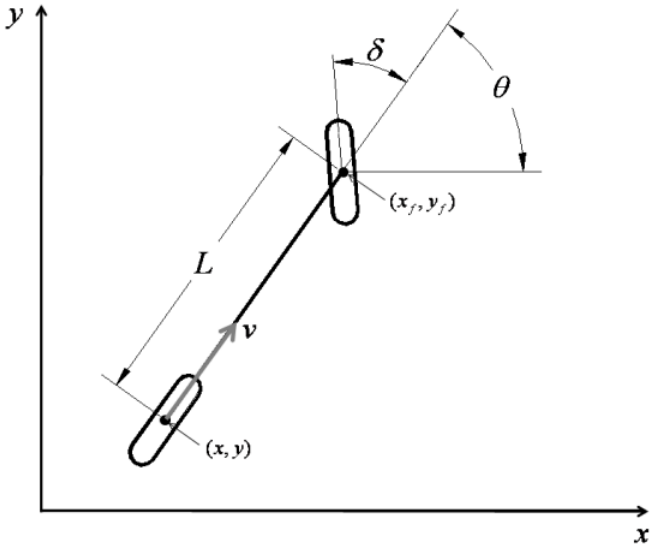

Figure 2: Kinematic bicycle model

2.2 Model Predictive Control (MPC) [27]

MPC is a combination between control and optimization. A control method finds the control input by optimizing a fitness function (J) dependent on restrictions. The fitness function computes the preferred control parameters by using the model of the system to predict future system outputs. Typically, MPC operates by solving an optimization issue at every sample time k to choose the control inputs for the next N sample times, known as the prediction horizon. A quadratic cost function commonly minimises control action and the error between the predicted and reference path, r. It is essential that the prediction and optimization work jointly to yield a series of the controller output u and the resultant system output y. The optimization problem is:

$$\begin{aligned} min J = & \sum_{i=1}^{N} ||r(k+i) - y(K+i)||_Q^2 \\ & + \sum_{i=0}^{M-1} ||\Delta u(k+i)||_S^2 \end{aligned} \tag{11}$$

the model restrictions,

$$\begin{aligned} x(k+i) &= f\big(x(k+i-1), u(k+i-1)\big) \\ &= Ax(k+i-1) + Bu(k+i-1) \\ y(k+i) &= g\big(x(k+i)\big) \\ &= Cx(k+i) + Du(k+i) \\ x(0) &= x_0 \end{aligned} \tag{12}$$

and,

$$\begin{aligned} Ax &\le b \\ Cx &\le 0 \\ u^{min} &\le u(k+i) \le u^{max} \end{aligned} \tag{13}$$

in which $A \in \mathbb{R}^{n\times n}$ means the state-transition matrix. The input matrix is $B \in \mathbb{R}^{n\times m}$. The output matrix is $C \in \mathbb{R}^{p\times n}$. $D \in \mathbb{R}^{p\times n}$ is used to allow a direct joining between *u* and *y*. In this paper $D = 0$.

During a prediction horizon, the state variables of the MPC are predicted at each sampling point. The MPC controller controls vehicle speed and steering based on linearized model.

The state of the vehicle at sample time *k* is:

$$z_{k+1} = [x_k, y_k, v_k, \theta_k] \quad (14)$$

x: longitudinal pose, y: lateral pose, v: velocity, θ: yaw angle

The control input is:

$$u_k = [\alpha_k, \delta_k] \quad (15)$$

α: linear acceleration ($\dot{v} = \alpha$), δ: steering angle.

The autonomous vehicle is modelled using a kinematic model:

$$\begin{aligned} z_{k+1} &= z_k + f(z_k, u_k).T \\ f(z_k, u_k) &= [v_k \cos\theta_k, v_k \sin\theta_k, \omega_k, \alpha_k]^T \end{aligned} \quad (16)$$

The MPC controller minimize this cost function for path tracking:

$$\begin{aligned} min\ & Q_f(z_{T,ref} - z_T)^2 \\ & + Q\sum_{k=2}^{T}(z_{k,ref} - z_k)^2 \\ & + R\sum_{k=1}^{T}{u_k}^2 \\ & + R_d\sum_{k=1}^{T}(u_{k+1} - u_k)^2 \end{aligned} \quad (17)$$

Where T is the horizon length, Q_f is the state-final matrix, Q is the state-cost matrix, R is the input cost matrix, and R_d is the input difference cost matrix. z_{ref} comes from target path and speed and subject to:

- Linearly vehicle model:

$$z_{k+1} = Az_k + Bu + C \quad (18)$$

- Maximum steering speed:

$$|u_{t+1} - u_t| < du_{max} \quad (19)$$

- Maximum steering angle:

$$|u_t| < u_{max} \quad (20)$$

- Initial state:

$$z_0 = z_0 \quad (21)$$

- Maximum and minimum speed:

$$v_{min} < v_k < v_{max} \quad (22)$$

- Maximum and minimum input:

$$u_{min} < u_k < u_{max} \quad (23)$$

You can get a discrete-time mode with Forward Euler Discretization with sampling time:

$$\boldsymbol{z_{k+1} = z_k + f(z_k, u_k)dt} \quad (24)$$

So:

$$z_{k+1} = Az_k + Bu_k + C \quad (25)$$

Where:

$$A = \begin{bmatrix} 1 & 0 & \cos(\theta)\,dt & -v\sin(\theta)\,dt \\ 0 & 1 & \sin(\theta)\,dt & v\cos(\theta)\,dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \dfrac{\tan(\delta)}{L}dt & 1 \end{bmatrix} \quad (26)$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ dt & 0 \\ 0 & \dfrac{v}{Lcos^2(\delta)}dt \end{bmatrix} \quad (27)$$

$$C = \begin{bmatrix} v\sin(\theta)\,\theta dt \\ -v\cos(\theta)\,\theta dt \\ 0 \\ -\dfrac{v\delta}{Lcos^2(\delta)}dt \end{bmatrix} \quad (28)$$

2.3 Batch Informed Trees (BIT*) [13] [28]

Commonly, the planners, based on graph search, are decomposing the SDC world into an m-dimensional net. Each cell in this net becomes a node located on the graph. BIT* method of creating a graph uses the RRT, which is structured by providing each node a metric motion planning.

BIT* utilises a specific heuristic suitable to search a sequence of growingly intensive implied RGGs while using prior information [29]. It is regarded as an expansion of incremental graph-based search methods, such as Lifelong Planning A* (LPA*) [30]. This anytime expression constructs sampling-based planners practical on many continued path planning in spite of dependence on sampling [28].

BIT* uses a heuristic sampling domain for the problem to minimise the path length. The search begins with start state x_{start} to reach a goal state x_{goal}. These points are called the focal points for the elliptical shape of the batch search, as shown in Figure 3. The style of the ellipse relies on both the x_{start} and x_{goal} to find the lower cost c_{min} and the best cost of the solution found c_{best}. The ellipse eccentricity is given by $\frac{c_{min}}{c_{best}}$.

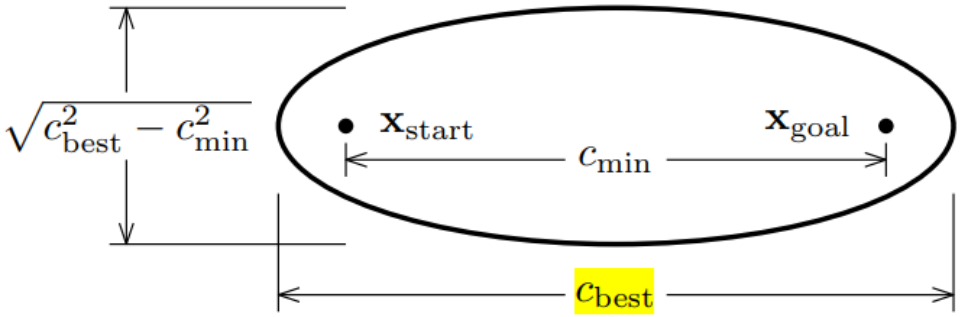

Figure 3: BIT* heuristic to discover the minimum length of the path.

BIT* includes a line ordering identical to a lazy TLPA*. In BIT*, a starting RGG with implied links is spread uniformly across the random samples. The RGGs edges have to a k-nearest graph [31] or r-disc graph [32]. So, the RGG r or k is picked to decrease graph complexity while preserving asymptotic prerequisites as a function of the samples' number. Then, a detailed tree is created from the begin, directed to the destination by a heuristic search (see Figure 4). This graph has edged with free collision, and its building finishes when a solution is discovered, or it can no longer be extended (Figure 5). This is the batch. If the solution is located, these samples are restricted to the subproblem that has a more suitable solution [33].

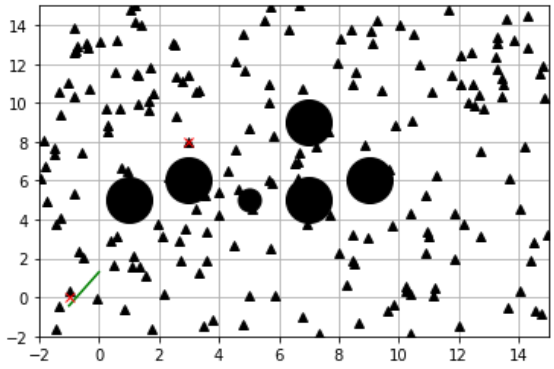

Figure 4: the batch of samples

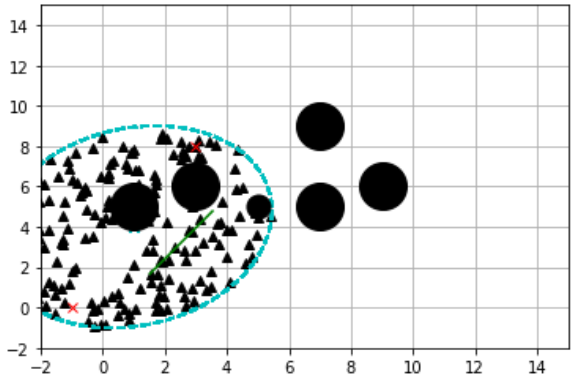

Figure 5: In every batch, the search expands around the min solution using a heuristic.

2.4 B-Spline Planner

B-spline [21], [22] curves contain all the benefits of Bezier curves and overpower the disadvantages that Bezier curves do not have on the local revision. The B-spline equation is:

$$p(u) = \sum_{i=0}^{p-1} d_i \, N_{i,k}\,(u) \qquad (29)$$

The d_i (i = 0, 1, …, p-1) means points in the collection S of samples, and the $N_{i,k}(u)$ represents the essential function and the declaration of this function is:

$$N_{i,0}(u) = \begin{cases} 1 & u_i \le u \le u_{i+1} \\ 0 & otherwise \end{cases}$$

$$\begin{aligned} & N_{i,k}(u) \\ & = \frac{u - u_i}{u_{i+k} - u_i} N_{i,k-1}(u) \\ & + \frac{u_{i+k+1} - u}{u_{i+k+1} - u} N_{i+1,k-1}(u) \end{aligned} \tag{30}$$

$$Define\ \frac{0}{0} = 0$$

And $u_i (i = 0, 1, \ldots, p\text{-}1)$ is the node parameter.

3 Results and Discussion

3.1 Environment setting

We implemented BIT* to display the efficiency of the proposed method using B-Spline and MPC under spyder (python 3.9) environment and used a global planner to develop reference trajectory. When an obstacle appears on the map, it will have a high "cost value". So, we can check collisions based on the cost. Both the environment and trajectories are shown using the plot function in python.

First, we use BIT* and the B-Spline to develop a global reference trajectory. The length of the path is taken as the path cost, and then we compare the path length, the loops and time in different algorithms to demonstrate the more acceptable achievement of the BIT* with B-Spline, as the fewer loops mean a quick converging rate. The B-Spline with BIT*is used to get the reference trajectory, then the MPC approach is applied for path tracking and controlling motion, and the achievement of our proposed work is explained in various scenarios.

3.2 BIT* and B-Spline Planner

The first scenario deals with a few obstacles with a narrow channel, as displayed in Figures (6-9). The positions of the beginning and the destination points are (-1, 0) and (3, 8), respectively. The red or blue solid lines are the trajectory created by using the global planner. In the map, we can see that there are three obstacles (circle shape) between the beginning and the destination, and the best trajectory for the mobile robot is through the selected channel among the obstacles.

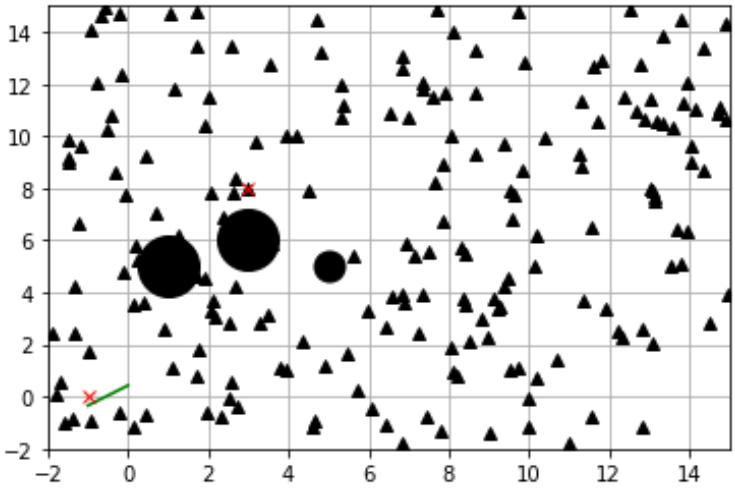

Figure 6: Initial state with few Obstacles

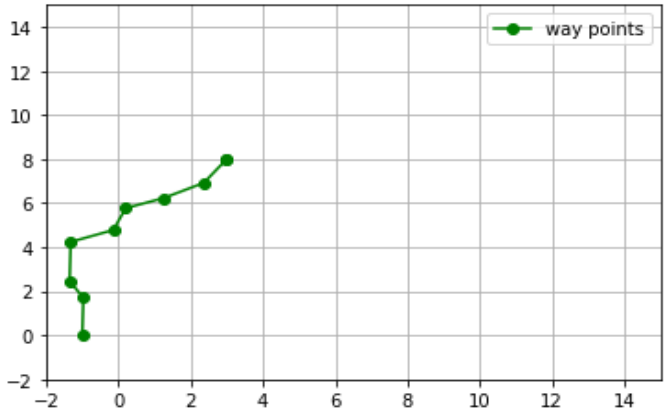

Figure 7: Path generated by BIT* only

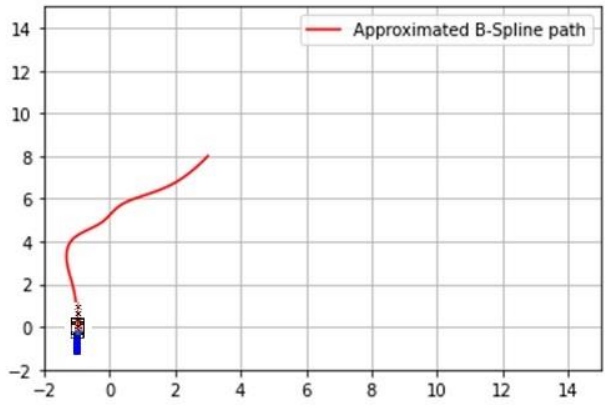

Figure 8: BIT* path after modified by Approximated B-Spline

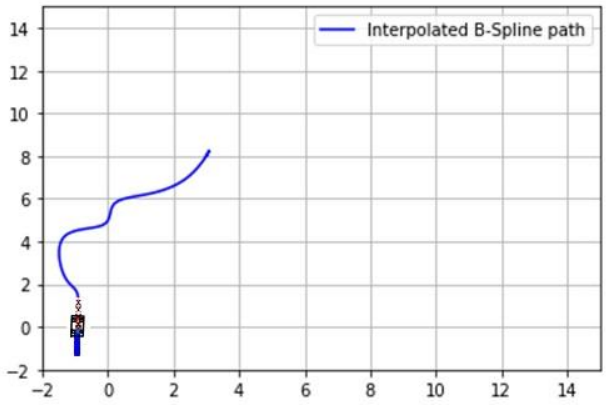

Figure 9: BIT* path after modified by Interpolated B-Spline

Table 1 shows the statistics of the planner with few obstacles.

Table 1: The results of the proposed work with few obstacles.

Path generated by	Path length	Computational Time (s)	Iteration
BIT*	10.38	73.63	80
BIT* with Approximated B-Spline	9.92	73.94	80
BIT* with Interpolated B-Spline	11.05	74.1	80

The second scenario deals with multiple obstacles. Mobile robots require to plan a free-from-obstacles trajectory in an environment has multiple obstacles (Figures 10-13) with many obstacles are used to test the achievement of the framework. The coordinates of the beginning and the destination points are (-1, -1) and (4, 12), respectively. There is a good path from the start to the goal. The path crosses the top left of the goal, so this is the best solution for the planner to see according to the initial state of the environment.

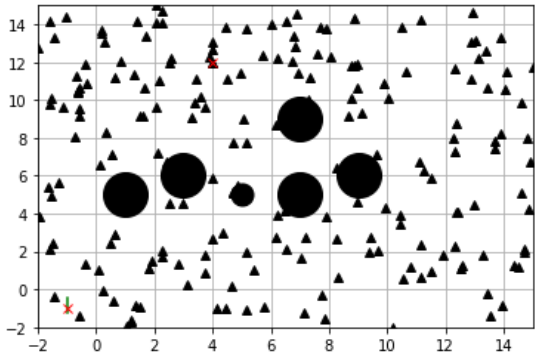

Figure 10: Initial state with many Obstacles

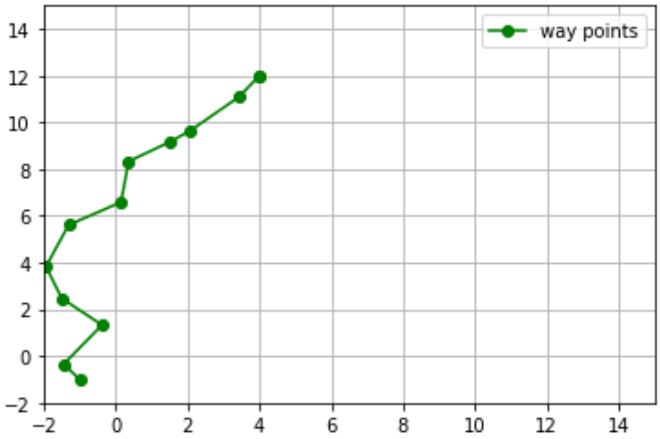

Figure 11: Path generated by BIT* only in many obstacles' environment

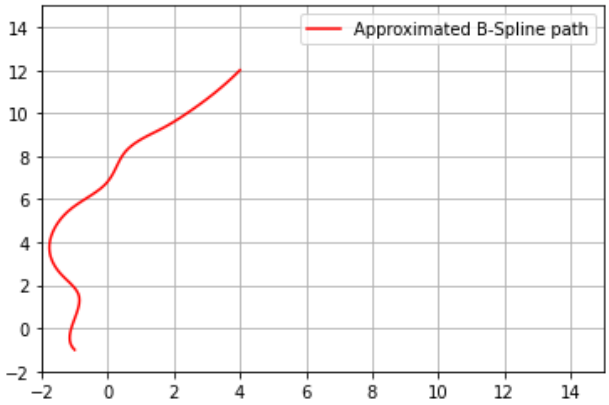

Figure 12: BIT* path after modified by Approximated B-Spline in many obstacles' environment.

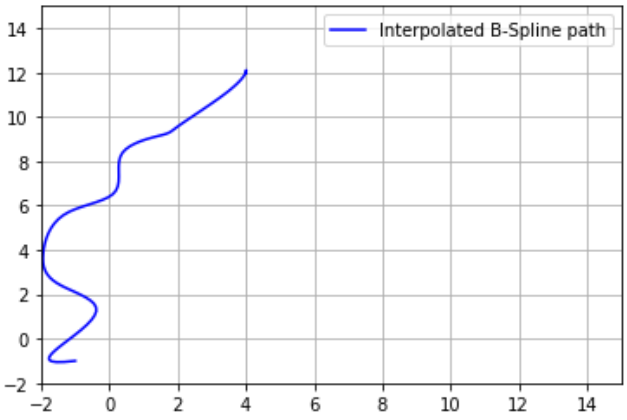

Figure 13: BIT* path after modified by Interpolated B-Spline in many obstacles' environment.

Table 2: The results of the proposed work with many obstacles.

Path generated by	Path length	Computational Time (s)	Iteration	Sample Size
BIT*	16.43	77.63	100	
BIT* with Approximated B-Spline	15.32	77.84	100	9296
BIT* with Interpolated B-Spline	17.62	78.31	100	

3.3 The Overall Framework

This section describes the whole framework to demonstrate the efficacy of our suggested method. The BIT* supported by the b-spline planner investigates the reference trajectory based on the current start and destination points. The MPC technique aws suggested to calculate the optimal linear speed and angular speed (control inputs) and passed them to the robot. The settings of some essential parameters are shown in Table 3.

Table 3. MPC Parameters in the framework.

Parameters	Value	Parameters	Values
Sampling Time	0.2	input cost matrix	diag([0.01, 0.01])
prediction horizon	5	input difference cost matrix	diag([0.01, 1.0])
execute horizon	2	state cost matrix	diag([1.0, 1.0, 0.5, 0.5])

Figures (14-17) show the yaw angle changes and path curvature in a map with few and many obstacles and the reference trajectory created by BIT* with a B-Spline planner.

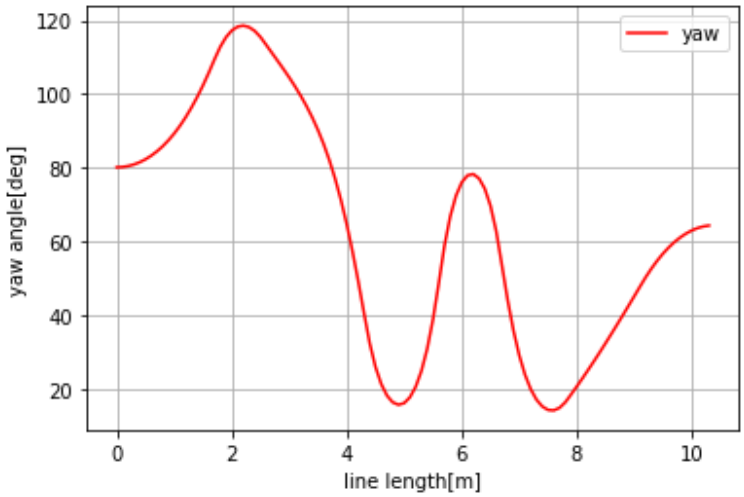

Figure 14: Yaw angle changes in few obstacles.

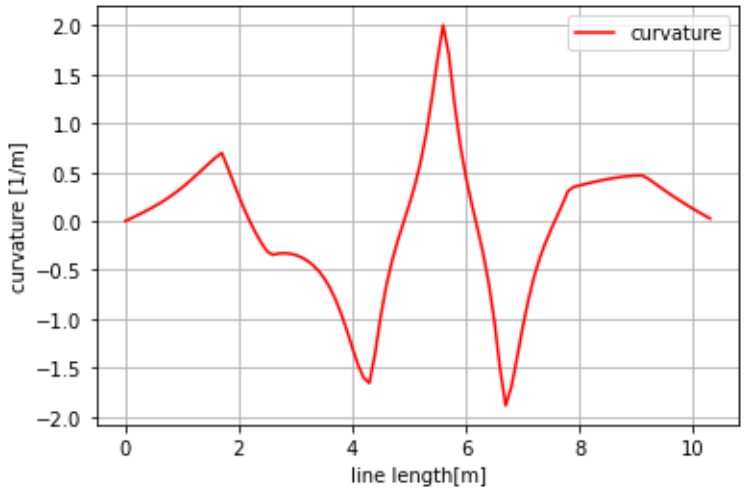

Figure 15: Curvature of the generated path in Few obstacles.

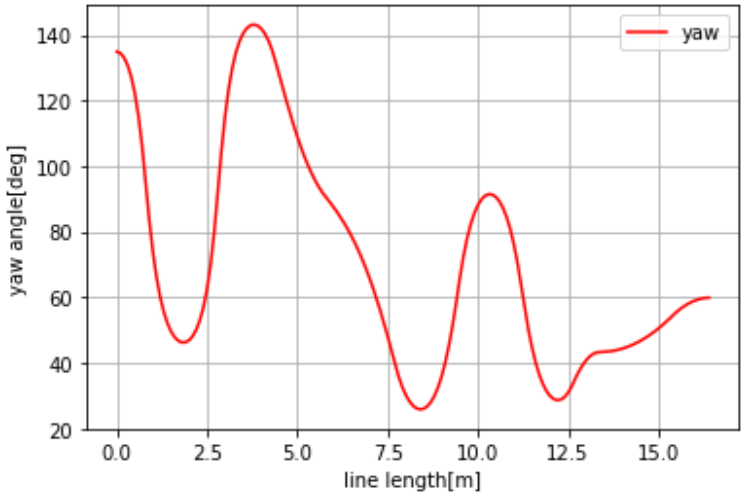

Figure 16: Yaw angle changes in many obstacles.

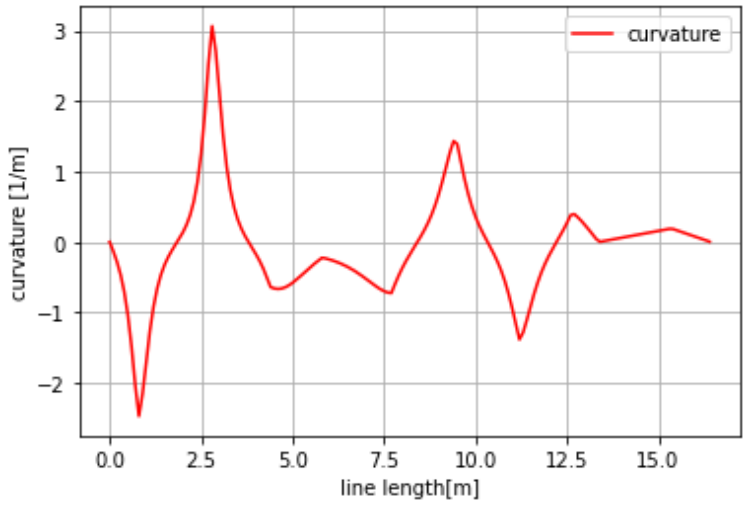

Figure 17: Curvature of the generated path in many obstacles.

4 Conclusions

This article describes planning the motion of a mobile robot by proposing a BIT* planner with a B-Spline method. The B-Spline method decreases the sharp and hard angle of the path generated by BIT. Besides, the B-Spline strategy bypassed some undeserved curves in the route. With this strategy, a reference trajectory appropriate for the robot could be acquired. Then, we use the MPC approach for path pursuit and controlling motion. Finally, we compare the performance of BIT* only, the BIT*with B-Spline approximation and the BIT* with B-Spline interpolated in two various environments, and the results display that the suggested BIT* with approximate B-Spline could discover the best trajectory with quick converging rate and shorter length. The entire framework's significance is also demonstrated in an

environment which has a few and more obstacles. The outcomes indicate the true path deviated from the reference path, but the parameters should be accommodated to balance the pursuit achievement and the direction to hold far from the obstacles. For future plans, it is important to demonstrate and test the strategy on a real robot's platform and enhance and develop our strategy to be used on distinct robots. Similarly, the dynamics of mobile robots in our controller design will be evaluated.

References

[1] H. Yang and X. Teng, "Mobile Robot Path Planning Based on Enhanced Dynamic Window Approach and Improved A∗ Algorithm," *Journal of Robotics*, vol. 2022, p. 2183229, 2022, doi: 10.1155/2022/2183229.

[2] C. Zhang, D. Chu, S. Liu, Z. Deng, C. Wu, and X. Su, "Trajectory Planning and Tracking for Autonomous Vehicle Based on State Lattice and Model Predictive Control," *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 2, pp. 29–40, 2019, doi: 10.1109/MITS.2019.2903536.

[3] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 174–179, 2017, doi: 10.1109/IVS.2017.7995716.

[4] S. Singh, "Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey," 2015.

[5] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016, doi: 10.1109/TIV.2016.2578706.

[6] G. N. Ambewadkar and S. P. Gajre, "Probe path planning for flatness measurement on coordinate measuring machine using ant colony optimization," *Advanced Engineering Forum*, vol. 41, pp. 86–91, 2021.

[7] M. Li, F. Zhang, and J. Y. Fang, "Optimal path solution based on Dijkstra algorithm," Frontiers in Economics and Management," *Frontiers in Economics and Management*, vol. 2, pp. 170–176, 2021.

[8] J. Santos, P. M. Rebelo, L. F. Rocha, P. Costa, and G. Veiga, "A∗ based routing and scheduling modules for multiple AGVs in an industrial scenario," *Robotics*, vol. 10, no. 2, p. 72, 2021.

[9] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Robotics: Science and Systems*, 2010, vol. 6, pp. 267–274. doi: 10.15607/rss.2010.vi.034.

[10] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011, doi: 10.15607/rss.2010.vi.034.

[11] J. Dai, D. Li, J. Zhao, and Y. Li, "Autonomous Navigation of Robots Based on the Improved Informed-RRT∗ Algorithm and DWA," *Journal of Robotics*, vol. 2022, p. 3477265, 2022, doi: 10.1155/2022/3477265.

[12] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, pp. 1478–1483. doi: 10.1109/ICRA.2011.5980479.

[13] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch Informed Trees (BIT∗): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2015, pp. 3067–3074. doi: 10.1109/ICRA.2015.7139620.

[14] B. Sahu, P. K. Das, M. R. Kabat, and R. Kumar, "Multi-robot cooperation and performance analysis with particle swarm optimization variants," *Multimedia Tools and Applications*, vol. 5, no. 9, pp. 1–24, 2021.

[15] H. Huang, G. Tan, and L. Jiang, "Robot Path Planning Using Improved Ant Colony Algorithm in the Environment of Internet of Things," *Journal of Robotics*, vol. 2022, p. 1739884, 2022, doi: 10.1155/2022/1739884.

[16] X. Du, K. K. K. Htet, and K. K. Tan, "Development of a Genetic-Algorithm-Based Nonlinear Model Predictive Control Scheme on Velocity and Steering of Autonomous Vehicles," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 11, pp. 6970–6977, 2016, doi: 10.1109/TIE.2016.2585079.

[17] F. Oldewurtel *et al.*, "Use of model predictive control and weather forecasts for energy efficient building climate control," *Energy and Buildings*, vol. 45, pp. 15–27, 2012, doi: 10.1016/j.enbuild.2011.09.022.

[18] M. A. Abbas, R. Milman, and J. M. Eklund, "Obstacle avoidance in real time with Nonlinear Model Predictive Control of autonomous vehicles," *Canadian Conference on Electrical and Computer Engineering*, vol. 40, no. 1, pp. 12–22, 2014, doi: 10.1109/CCECE.2014.6901109.

[19] A. Koga, H. Okuda, Y. Tazaki, T. Suzuki, K. Haraguchi, and Z. Kang, "Realization of different driving characteristics for autonomous vehicle by using model predictive control," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2016-Augus, no. Iv, pp. 722–728, 2016, doi: 10.1109/IVS.2016.7535467.

[20] C. Wang, X. Zhang, K. Guo, F. Ma, and D. Chen, "Application of Stochastic Model Predictive Control to Modeling Driver Steering Skills," *SAE International Journal of Passenger Cars - Mechanical Systems*, vol. 9, no. 1, pp. 116–123, 2016, doi: 10.4271/2016-01-0462.

[21] A. Biran, *Geometry for naval architects*, First edit. Elsevier Ltd, 2018. doi: 10.1016/C2014-0-03962-7.

[22] J. Gallier, *Curves and Surfaces In Geometric Modeling: Theory And Algorithms*. 2018.

[23] P. Xu, N. Wang, S. L. Dai, and L. Zuo, "Motion planning for mobile robot with modified BIT* and MPC," *Applied Sciences (Switzerland)*, vol. 11, no. 1, p. 426, 2021, doi: 10.3390/app11010426.

[24] Campion G, Bastin G, and D'andrea-Novel B, "Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots," *IEEE transactions on robotics and automation*, vol. 12, no. 1, pp. 47–62, 1996.

[25] F. Kühne, W. F. Lages, and J. G. da Silva Jr, "Model Predictive Control of a Mobile Robot Using Linearization," *International Journal of Control, Automation and Systems*, vol. 13, no. 4, pp. 1868–1879, 2015, [Online]. Available: http://dx.doi.org/10.1016/j.advengsoft.2015.10.008%0Ahttp://dx.doi.org/10.1016/j.robot.2015.04.005%0Ahttp://www.ece.ufrgs.br/~fetter/mechrob04_553.pdf

[26] R. Rajamani, "Vehicle Dynamics and Control," *Springer*. p. 496, 2012. doi: 10.1016/b978-0-08-100390-9.00001-4.

[27] F. Borrelli, A. Bemporad, and M. Morari, *Predictive*

Control for Linear and Hybrid Systems. 2017. doi: 10.1017/9781139061759.

[28] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch Informed Trees (BIT*): Informed asymptotically optimal anytime search," *International Journal of Robotics Research*, vol. 39, no. 5, pp. 543–567, 2020, doi: 10.1177/0278364919890396.

[29] M. Penrose, *Random Geometric Graphs*. Oxford: Oxford University Press, 2003. doi: 10.1093/acprof:oso/9780198506263.001.0001.

[30] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong Planning A∗," *Artificial Intelligence*, vol. 155, no. 1, pp. 93–146, 2004, doi: https://doi.org/10.1016/j.artint.2003.12.001.

[31] F. Xue and P. R. Kumar, "The number of neighbors needed for connectivity of wireless networks," *Wireless Networks*, vol. 10, no. 2, pp. 169– 181, 2004.

[32] E. N. Gilbert, "Random plane networks," *SIAM*, vol. 9, no. 4, pp. 533–543, 1961.

[33] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT∗: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2997–3004, 2014, doi: 10.1109/IROS.2014.6942976.