Received 5 December 2023, accepted 4 January 2024, date of publication 11 January 2024, date of current version 19 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3353186

RESEARCH ARTICLE

Optimal Kinodynamic Trajectory Planner for Mobile Robots in an Unknown Environment Using Bézier Contours

AMNA MAZEN[1,2], MARIAM FAIED[2], (Senior Member, IEEE), AND MOHAN KRISHNAN[2], (Life Senior Member, IEEE)
[1]Faculty of Engineering, Electrical Engineering Department, Fayoum University, Faiyum 63514, Egypt
[2]Department of Electrical and Computer Engineering and Computer Science, University of Detroit Mercy, Detroit, MI 48221, USA

Corresponding author: Amna Mazen (amazen@udmercy.edu)

ABSTRACT Trajectory planning in the field of mobile robotics involves the generation of a trajectory to navigate a robot from a start state to a goal state. One widely employed technique involves a two-step approach: a path planner generates a path made up of piecewise linear segments with sharp turns, which are then smoothed in the trajectory generation step. In contrast, this work formulates trajectory generation as an optimization problem based on the Bézier curve, denoted as 'BTP', to generate the robot's trajectory in one step. It uses a weighted objective function of trajectory length and navigation time to suit different optimization strategies while considering the robot's kinematics and dynamics limitations. BTP adopts matrix-based formulations for all mathematical operations to enable dynamic adjustment of the degree of the Bézier curve during the optimization process, if convergence is not obtained with the current degree. Additionally, BTP guarantees that the robot's trajectory is always within the open space identified by the robot's sensors. The efficacy of BTP has been evaluated through simulations and real-world experimentation, including soccer games and cluttered environment scenarios. Finally, the performance is benchmarked against some of the existing trajectory planners. BTP reduced the robot's navigation time by a minimum of 11% up to 55% compared to other tested trajectory planners, ensuring C^2 continuity rather than just C^1 continuity. Furthermore, it consistently achieved precise goal configuration, unlike the tested trajectory planners, which exhibited deviations of up to 0.6 meters.

INDEX TERMS Bézier curve, kinodynamic robot's trajectory, nonlinear optimization, trajectory planner.

I. INTRODUCTION

The last decade has witnessed remarkable advancements in the application of autonomous mobile robots in various fields, such as the warehousing industry [1], food delivery [2], search and rescue [3], and planetary exploration [4]. Companies in the commercial sector, including established ones like Amazon and some startups [5], are actively deploying autonomous robots for delivery purposes. Mobile robots also play vital roles in challenging tasks like detecting landmines, exploring other planets, delivering goods to customers, and operating in container terminals. One significant advantage of mobile robots is their ability to operate in hazardous environments, such as areas with radiation or pollution, where humans are at risk [6]. These applications depend significantly on developing an efficient motion planner to enable the robot to move from a start to a goal position.

Motion planning of a robot is generally divided into two main sub-areas: path planning and trajectory generation. The broad interpretation of these two sub-areas is that path planners generate a time-independent sequence of waypoints from the start to the goal position, without considering curvature continuity. Then, trajectory generation focuses on converting the path to a trajectory by incorporating a driveable velocity profile. Thus, it splits finding an optimal kinodynamic trajectory into a two-step approach for problem manageability. The first step finds an optimal obstacle-free path generally made up of piecewise linear segments with

The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad AlShabi.

sharp turns. The second step then smooths it by generating a motion profile that obeys velocity and acceleration limits. Some researchers have attempted to solve these two aspects as an integrated problem. However, it is computationally demanding when this strategy is adopted.

A. RELATED WORK

Path planning algorithms are classified into graph-based, sampling-based, sensor-based, and learning-based approaches [7]. Graph-based path planners such as A* [8] generally generate an optimal path with sharp changes in direction that need to be smoothed further with additional techniques, such as the Bézier curve. Sampling-based path planners, such as Rapidly Exploring Random Tree (RRT) [9], randomly sample the state space to find a feasible path between a start and a goal state while avoiding obstacles. Sensor-based path planners such as Bug 1 and Bug 2 [10] rely on local sensor data to determine the path, by following the obstacle boundary when one is encountered and otherwise just moving in a straight line. Learning-based path planners, such as reinforcement learning techniques, use a reward/penalty strategy to teach the robot to choose a path to follow [11].

Trajectory generation work is split into model-based and geometry-based methods. Most algorithms for model-based trajectory generation adhere to the Model Predictive Control (MPC) framework [12]. MPC creates a linear/nonlinear model of the robot's dynamics and environment. Then, it uses this model and the robot's current state to predict its future over a finite time horizon. Finally, it optimizes the control inputs to the robot to satisfy desirable criteria, such as increasing trajectory smoothness, while satisfying constraints such as velocity, acceleration, and obstacle avoidance. MPC suffers from three main drawbacks: firstly, it generates trajectories without immediate feasibility checks, wasting effort on infeasible trajectories; secondly, the optimization process heavily depends on the initial parameters provided, such as prediction horizon. Thirdly, each optimization iteration necessitates running MPC simulations, significantly slowing the trajectory generation process.

In geometry-based trajectory generation, various geometric primitives, including arcs [13], B-splines [14], and Bézier curves [15], have been utilized. Dubins' early research [13] introduced the concept of creating the shortest paths using straight lines and circular arcs. Although the tangent continuity is maintained in Dublin's paths, there is a sudden change in curvature at the points where straight lines meet circular arcs. Li et al. [14] used a cubic B-spline to smooth curves generated by sampling-based path planners in cluttered environments. They also adopted a local trajectory adjustment strategy to avoid collisions.

Most state-of-the-art Bézier curve work focuses on the second step of the two-step trajectory generation process. Chen et al. [16] employed a quartic Bézier curve for robot trajectory generation, meeting constraints on velocity and curvature. Their trajectory generation methodology was split into two phases - a trajectory satisfying curvature restrictions was initially constructed, followed by a velocity-constrained implementation phase. However, the algorithm did not guarantee the feasibility of the obtained trajectory with the incorporation of the robot's velocity constraints. Moreover, the two-stage method increased computational time and required over 500 iterations to converge to a feasible solution in the example provided. Zhang et al. [17] developed two-step trajectory generation with a quintic Bézier curve. The first step involves using Dijkstra's algorithm as a global path planner to generate the path's waypoints within a known environment. Subsequently, the second step smooths the path created in the first step using a quintic Bézier curve. This process includes optimizing the curve by adjusting the magnitude of each segment's initial and final tangents, aiming to create the shortest possible path, while adhering to the robot's kinematic and dynamic (kinodynamic) constraints. However, this method encounters issues with discontinuity when the angle difference between the robot's current heading and the next path segment exceeds 90 degrees. In such scenarios, the algorithm directs the robot to rotate in place to align with the upcoming segment's angle, aiming to minimize the curve's length.

Trajectory planning algorithms are employed in robotics applications to generate an obstacle-free trajectory between a starting and target configuration (configuration includes position and some kinodynamic constraints, as discussed later). Trajectory planners adopt a systematic approach to find the optimal trajectory that minimizes a cost function, such as minimizing navigation time, trajectory length, or energy consumption while satisfying a set of constraints [18]. Kim and Kim [19] proposed a time-optimal trajectory planning algorithm for environments with multiple circular obstacles. This algorithm takes into consideration the robot's dynamics. They tackled the nonlinearity resulting from the complexity of the dynamics by dividing the trajectory into small sections. Kielas-Jensen and Cichella introduced the Bernstein/Bézier Optimal Trajectories (BeBOT) approach [20], which formulates optimal trajectory generation as a NonLinear Programming (NLP) problem based on Bézier curves. The authors demonstrated that Bézier curves have desirable properties to enforce constraints along the robot's trajectory. Their approach considered forward and angular velocity constraints, while minimizing a single optimization criterion for theoretical examples that did not include simulated or real robots. While drawing some inspiration from their work, we significantly extend it to unknown environments with irregular obstacle shapes while incorporating extensive testing in simulated and real-world scenarios. Furthermore, we consider various optimization criteria discussed in the next subsection.

B. CONTRIBUTION

Most state-of-the-art Bézier curve work suffers from two problems. The first is fixing the degree of the Bézier curve, requiring all analyses to be repeated if convergence is not obtained with the chosen degree. The second is limiting the focus to smoothing the discontinuities of a path planner. This paper proposes a Bézier curve-based Trajectory Planner, denoted as 'BTP', which circumvents this first limitation by adopting matrix-based formulations for all mathematical operations. As a result, the degree of the Bézier curve can be dynamically adjusted during the optimization process, in cases where the ongoing optimization process has difficulty converging to a viable solution. Also, instead of just smoothing the path discontinuities, BTP generates a robot's entire trajectory in one step by formulating trajectory generation as an optimization problem. It uses a weighted objective function to accommodate different optimization strategies, while considering its kinodynamic limitations in unknown environments. In addition, an algorithm based on Delaunay Triangulation (DT) is proposed to describe the robot's open (obstacle-free) space. The effectiveness of 'BTP' is evaluated through simulations and real-world experiments using the Pioneer P3-DX robot in various scenarios. Furthermore, the performance of 'BTP' is benchmarked against some of the well-known trajectory planners.

The organization of this paper is as follows: Section II discusses the robot's kinodynamic terms and the dependency of properties of the Bézier curve on them. Section III proposes BTP, which formulates the trajectory generation as an optimization problem using the Bézier curve, considering the robot's kinodynamic terms and open space. Section IV presents the deployment of BTP to diverse scenarios, employing the Pioneer P3-DX robot in both simulated and real environments. In addition, BTP is benchmarked against some existing trajectory planners based on different performance metrics. Finally, Section VI summarizes the main contributions and identifies avenues for future research.

II. MODELING

In this work, we aim to optimize the trajectory of a mobile robot moving from an initial to a final configuration within an unknown environment, taking the robot's kinodynamics into account. The trajectory is represented using a Bézier curve, which possesses favorable characteristics for incorporating the robot's kinematic and dynamic limitations within its definition. Furthermore, our approach requires depicting the open space available to the robot within the environment and ensuring that the trajectory generated by the optimization algorithm in each iteration is restricted to this open space.

This section reviews the concept of the Bézier curve and examines how its properties facilitate incorporating a robot's kinodynamic behavior. Specifically, the focus is on representing the robot's velocities, accelerations, orientation, and curvature as Bézier curves. The notation $P(t)$ is utilized to represent a continuous curve, while $P_m(t)$ denotes its m^{th} degree Bézier curve with control points $(\bar{P}_m = \{\bar{p}_{i,m}, i = 0, \ldots, m\})$.

A. ROBOT'S KINODYNAMIC TERMS

This work employs a differentially-steered robot with position and orientation represented by $P(t) = [P^x(t), P^y(t)]$ and $\psi(t)$, respectively. The change in the robot's position and orientation at any time, t, is determined by using the kinematic equation (1), which relates these quantities to the robot's forward and angular velocities.

$$\begin{bmatrix} \dot{P}^x(t) \\ \dot{P}^y(t) \\ \dot{\psi}(t) \end{bmatrix} = \begin{bmatrix} cos(\psi(t)) & 0 \\ sin(\psi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{1}$$

While equation (1) describes how the instantaneous rate of change of the robot's pose is related to its instantaneous velocities, the effect of robot dynamics needs to be accommodated. That is, how these velocities are reached, keeping in mind the limitations of dynamics. As a first step in the modeling strategy, this requires that the robot's orientation, linear and angular velocities, tangential and radial accelerations, and curvature be expressed solely as a function of the first and second derivatives of position, as shown in the equation (2). These relationships can be derived by applying basic math operations such as summation, subtraction, multiplication, and division to equation (1).

$$\psi(t) = \arctan \frac{\dot{P}^y(t)}{\dot{P}^x(t)} \tag{2a}$$

$$||v(t)||^2 = (\dot{P}^x(t))^2 + (\dot{P}^y(t))^2 \tag{2b}$$

$$\omega(t) = \frac{\dot{P}^x(t)\ddot{P}^y(t) - \dot{P}^y(t)\ddot{P}^x(t)}{(\dot{P}^x(t))^2 + (\dot{P}^y(t))^2} \tag{2c}$$

$$a^t(t) = \frac{\dot{P}^x(t)\ddot{P}^x(t) + \dot{P}^y(t)\ddot{P}^y(t)}{v(t)} \tag{2d}$$

$$a^r(t) = \frac{\dot{P}^x(t)\ddot{P}^y(t) - \dot{P}^y(t)\ddot{P}^x(t)}{v(t)} \tag{2e}$$

$$k(t) = \frac{\omega(t)}{v(t)} \tag{2f}$$

In addition to incorporating the kinematic and dynamic aspects of the robot, this work also attempts to prevent wheel sideslip by ensuring that its tangential and radial accelerations remain within an ellipse defined by their respective limits (a^t_{max}, a^r_{max}) [21], as described below:

$$\frac{(a^t(t))^2}{(a^t_{max})^2} + \frac{(a^r(t))^2}{(a^r_{max})^2} \leq 1 \tag{3}$$

Continuing with our optimization model formulation, the robot's position is represented by a Bézier curve. This automatically makes its orientation, linear and angular velocities, tangential and radial accelerations, and curvature Bézier curves because of the relationships described in equations (2). In each iteration of the algorithm, the control points of the Bézier curve for the robot's position alone are tuned while satisfying all the constraints. The control points of the Bézier curves corresponding to the robot's orientation, velocities, accelerations, and curvature are affected indirectly through their dependencies on the position control points.

The dependencies are used to enforce compliance with the robot's limitations that relate to the dynamics. The following sub-section reviews the Bézier curve and its associated properties.

B. BÉZIER CURVE DEFINITION

An m^{th} degree Bézier curve is a parametric curve defined by a set of control points that control its shape as indicated in equation (4). It is commonly referred to as Bernstein Polynomials (BPs). In this work, the Bézier curve parameter is the robot's navigation time (t), and ranges from '0' to the final navigation time 't_f', as follows:

$$P_m(t) = \begin{bmatrix} P^x_m(t) \\ P^y_m(t) \end{bmatrix} = \sum_{i=0}^{m} \bar{p}_{i,m} B_{i,m}(t), \qquad t \in [0, t_f] \tag{4}$$

where $P_m(t)$ is the m^{th} degree Bézier curve, $(P^x_m(t), P^y_m(t))$ are the x and y components of $P_m(t)$ respectively, $(\bar{p}_{i,m} \in \mathbb{R}, i = 0, \ldots, m)$ is the i^{th} control point of $P_m(t)$, and $B_{i,m}(t) = \binom{m}{i} \frac{t^i (t_f - t)^{m-i}}{t_f^m}$ is the basis of $P_m(t)$ where $\binom{m}{i} = \frac{m!}{i!(m-i)!}$ is the binomial coefficient.

To express the robot's orientation, angular velocity, tangential and radial accelerations, and curvature, we need to use rational Bézier curves. The rational Bézier curve is an extension of the Bézier curve that incorporates weights. It is used to represent the division of two one-dimensional Bézier curves. An m^{th} degree rational Bézier curve, $Q_m(t)$, is defined as:

$$Q_m(t) = \frac{\sum_{i=0}^{m} \bar{q}_{i,m} w_{i,m} B_{i,m}(t)}{\sum_{i=0}^{m} w_{i,m} B_{i,m}(t)}, \qquad t \in [0, t_f] \tag{5}$$

where $(\bar{q}_{i,m} \in \mathbb{R}, i = 0, \ldots, m)$ is the i^{th} control point of $Q_m(t)$ and $(w_{i,m} \in \mathbb{R}, i = 0, \ldots, m)$ is referred to as the weight of the i^{th} control point.

In this work, the robot's position is represented by a Bézier curve $(P_m(t))$, equation (4), with control points $(\bar{p}_{i,m}, \quad i = 0, \ldots, m)$ which facilitates the formulation of the trajectory optimization problem as an NLP. The robot's trajectory and kinodynamic terms are represented by Bézier curves, which possess properties that relate well to the robot's drivability, as explained earlier. Here, we will discuss the essential properties of the Bézier curve and illuminate how these properties are utilized to deduce the control points of the robot's kinodynamic terms, using the robot's position control points as a reference. For a comprehensive analysis of these properties, the reader is encouraged to consult the work of Farouki [22].

Property 1: Convex hull- A Bézier curve is always contained within the area enclosed by its control points, known as the convex hull. In our work, if the convex hull of the robot's position Bézier curve does not contain any obstacles, it is guaranteed that the position Bézier curve will be free of obstacles.

Property 2: Endpoint Values- A Bézier curve always passes through its initial and final control points, known as endpoints. This property ensures that the robot's position Bézier curve always passes through its start and goal positions $(P_m(0) = \bar{p}_{0,m}, P_m(t_f) = \bar{p}_{m,m})$.

Moreover, the second control point $(p_{1,m})$ and the second last control point $(p_{m-1,m})$ are located on its tangent at the robot's start and goal positions, respectively as shown in equation (6). In the context of our work, this property guarantees that the robot's predefined initial and final orientations are satisfied through the appropriate choice of $p_{1,m}$ and $p_{m-1,m}$.

$$\begin{aligned} \dot{P}_m(0) &= \frac{m}{t_f}(p_{1,m} - p_{0,m}) \\ \dot{P}_m(t_f) &= \frac{m}{t_f}(p_{m,m} - p_{m-1,m}) \end{aligned} \tag{6}$$

Property 3: Derivative- The derivative of an m^{th} degree Bézier curve, $\dot{P}_{m-1}(t)$, is a Bézier curve of degree $(m-1)$ described by equation (7). Its control points are calculated as the difference of two successive control points of the original Bézier curve $(P_m(t))$. Alternatively, the control points can be calculated in matrix form by multiplying the control points of the original m^{th} degree Bézier curve $(\bar{P}_m)$ with the difference matrix $(D_m \in \mathbb{R}^{(m+1)\times m})$. This work adopts matrix-based formulations for all mathematical operations encompassing addition, subtraction, multiplication, division, differentiation, and degree elevation to facilitate the dynamic change of the Bézier curve degree during the optimization process if convergence is not achieved with the current degree.

$$\begin{aligned} \dot{P}_{m-1}(t) &= \sum_{i=0}^{m-1} \frac{m}{t_f}(p_{i+1,m} - p_{i,m}) B_{i,m}(t), \qquad t \in [0, t_f] \\ &= \sum_{i=0}^{m-1} \bar{P}_m D_m B_{i,m}(t), \qquad t \in [0, t_f] \\ D_m &= \frac{m}{t_f} \begin{bmatrix} -1 & 0 & \ldots & 0 \\ 1 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -1 \\ 0 & \ldots & 0 & 1 \end{bmatrix} \end{aligned} \tag{7}$$

In our work, this property is used to calculate the robot position's first and second derivatives. When applied to the position Bézier curve $(P_m(t))$, it results in the first derivative components $(\dot{P}^x_{m-1}(t)$ and $\dot{P}^y_{m-1}(t))$ as $(m-1)^{th}$ degree Bézier curves, and the second derivative components $(\ddot{P}^x_{m-2}(t)$ and $\ddot{P}^y_{m-2}(t))$ as $(m-2)^{th}$ degree Bézier curves.

Property 4: Degree Elevation- Elevating the degree of a Bézier curve increases its degree without altering its shape. To elevate the m^{th} degree Bézier curve, equation (4), to n^{th} degree Bézier curve, $S_n(t)$,$(n > m)$, the elevated Bézier curve must have the same endpoints $(s_{0,n} = p_{0,m}, s_{n,n} = p_{m,m})$. The remaining control points of the elevated Bézier curve are calculated in a matrix form by multiplying the control points of the original m^{th} degree Bézier curve $(\bar{P}_m)$ by the degree elevation matrix (E_n).

$$\bar{S}_n = \bar{P}_m E_n \tag{8}$$

where $E_n = \{e_{i,i+j}\} \in \mathbb{R}^{(m+1)\times(n+1)}$ is the degree elevation matrix with $(i, i+j)$ elements values given by equation (9) and the remaining elements are zero.

$$e_{i,i+j} = \frac{\binom{n-m}{j}\binom{m}{i}}{\binom{n}{i+j}}, \quad i = 0, \ldots, m, \quad j = 0, \ldots, n-m \tag{9}$$

This technique is utilized in our work to increment the degree of the Bézier curve by one after using the derivative property. Consequently, this allows for applying arithmetic operations that require the Bézier curves to have the same degree. The application of this property to the first and second derivatives of the robot's position Bézier curve results in the $(m)^{th}$ degree Bézier curves, represented by $\dot{P}^x_m(t)$, $\dot{P}^y_m(t)$, $\ddot{P}^x_m(t)$, and $\ddot{P}^y_m(t)$.

Property 5: Arithmetic Operations- Arithmetic operations include addition, subtraction, product, and division of two Bézier curves. As an example, these operations will be illustrated by their application in calculating the control points of the robot's angular velocity Bézier curve (equation 2c).

- When multiplying two Bézier curves, the resulting curve will have a degree equal to the sum of the degrees of the original curves. In the context of the robot's angular velocity Bézier curve, the products $\dot{P}^x_m(t)\ddot{P}^y_m(t)$, $\dot{P}^y_m(t)\ddot{P}^x_m(t)$, $(\dot{P}^x_m(t))^2$, and $(\dot{P}^y_m(t))^2$ yield $(2m)^{th}$ degree Bézier curves.
- The degree of a Bézier curve remains unchanged when adding or subtracting two Bézier curves of the same degree. In the context of the robot's angular velocity Bézier curve, the numerator subtraction $\dot{P}^x_m(t)\ddot{P}^y_m(t) - \dot{P}^y_m(t)\ddot{P}^x_m(t)$ results in a $(2m)^{th}$ degree Bézier curve, and the denominator sum $(\dot{P}^x_m(t))^2 + (\dot{P}^y_m(t))^2$ also yields an $(2m)^{th}$ degree Bézier curve.
- Furthermore, the ratio of two Bézier curves can be represented as a rational Bézier curve. In the context of the robot's angular velocity Bézier curve, the division of the $(2m)^{th}$ degree numerator and denominator Bézier curves is represented as a $(2m)^{th}$ degree rational Bézier curve. It is important to note that without the application of degree elevation on the first and second derivatives of the robot's position Bézier curve, the numerator will be an $(2m-3)^{th}$ degree Bézier curve, and the denominator will be an $(2m-2)^{th}$ degree Bézier curve, thus making it impossible to apply the ratio property directly. In this work, the robot's orientation, angular velocity, tangential and radial accelerations, and curvature are represented as rational Bézier curves due to the involvement of a division operation between two Bézier curves.

Property 6: De Casteljau algorithm- The De Casteljau algorithm [23] is an efficient method for evaluating the value of a Bézier curve at any given time. By utilizing the properties of Bézier curves previously discussed, the control points of the robot's kinodynamic Bézier curves can be derived and formulated. The De Casteljau algorithm can then evaluate these terms at any time instance $t \in [0, t_f]$ during the optimization problem.

Using equation (2) and the properties of the Bézier curve, the control points of the robot's kinodynamic terms are expressed as functions solely of the position control points. For instance, the control points of the robot's forward velocity Bézier curve (equation 2b) are calculated through a three-step process. Firstly, the robot's position control points are differentiated using the elevated derivative property. Secondly, the product property is applied to square the differentiated terms. Lastly, the squared terms are summed using the sum property. Note that a squared version of the forward velocity ($v^2_m(t)$) must be used to maintain a polynomial form for the equation because working with $v_m(t)$ would involve the use of a square root operation, which would not yield a polynomial. As a result, squared versions of the robot's curvature and tangential and radial accelerations will also be used since this corresponds to the squared version of velocity. For the same reason, a tangential version of the robot's orientation must be used to maintain its equation in polynomial form. In the next section, the robot's trajectory is optimized while considering the constraints imposed by the robot's kinodynamic limitations.

III. METHODOLOGY

This section discusses in detail the proposed Bézier curve-based Trajectory Planner, denoted as 'BTP', that generates the optimal trajectory for a mobile robot while considering its kinodynamic limitations. To accomplish this, we employ the Bézier curve to represent the continuous-time optimization problem as an NLP problem that can be solved using a numerical optimization algorithm. Moreover, we propose a safety check algorithm to ensure the generated trajectory falls within the robot's available open space.

A. ROBOT'S TRAJECTORY SAFETY CHECK ALGORITHM

A trajectory planner requires an initial description of the environment, as ascertained by the robot's sensors, as input to identify the open space available to restrict the optimal trajectory. To accomplish this, we propose algorithm 1 that utilizes LiDAR readings and Delaunay Triangulation [24] to represent this open space. Algorithm 1 has as input the LiDAR readings (Li_{read}) with maximum range (R_{max}), and the robot's position Bézier curve ($P_m(t)$), and outputs a flag (*safe_flag*) indicating whether the robot's trajectory lies within the open space or not, in the current optimization iteration. The algorithm is demonstrated through a simple example in which the robot's start and goal positions (C_0 and C_g) are located at (0,0) and (4,4), respectively, and two cubical and one spherical obstacle are contained within the environment, as depicted in Figure 1. This simplified obstacle configuration is only to illustrate the safety check

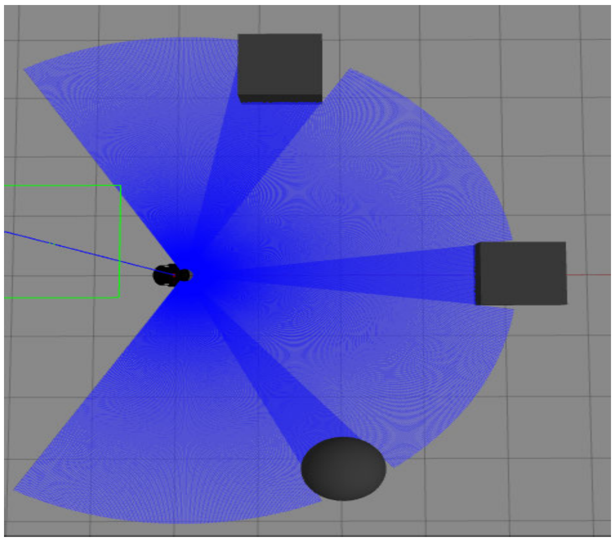


FIGURE 1. Simplified Gazebo environment obstacle layout to understand safety check algorithm.

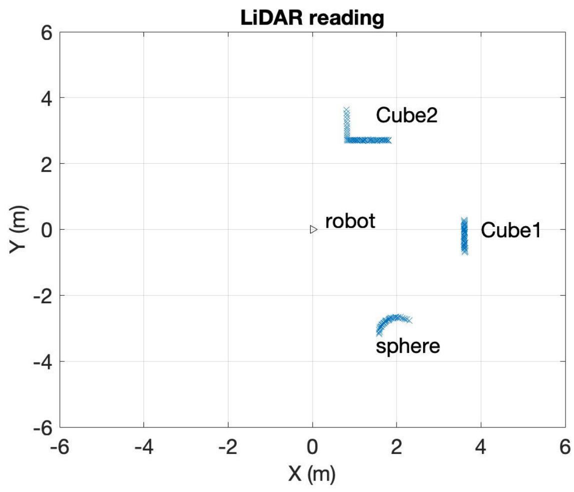


FIGURE 2. LiDAR readings from simplified obstacle layout for conceptual understanding of open space.

algorithm - actual testing is done on more complex obstacle configurations in the results section. The LiDAR view provides the initial description of this environment as depicted in Figure 2.

The MATLAB® built-in "delaunayTriangulation" function then decomposes this initial description into a set of non-overlapping triangle meshes (DT), line 1 in the algorithm, as indicated in Figure 3. The endpoints of the robot's position Bézier curve are fixed at the start and goal positions, as indicated in line 2 of Algorithm 1. The Delaunay triangles that intersect the line connecting the robot's start and goal positions are referred to as "*Intersect_tri*" and are depicted as colored triangles in Figure 3. These intersection triangles are then concatenated into a single polygon (Pol), line 5 in the algorithm, to represent the open space within which the robot's trajectory must be restricted. Finally,

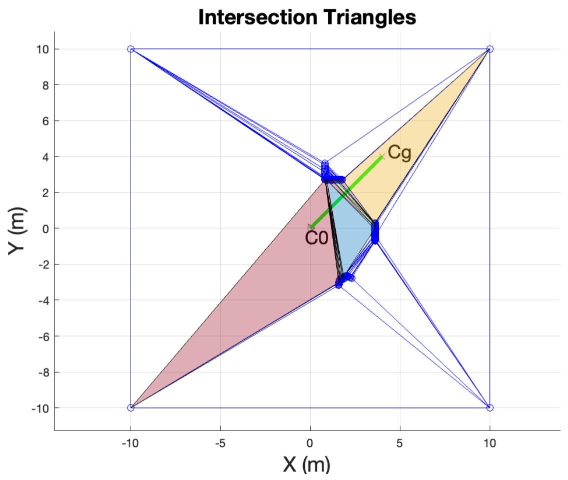


FIGURE 3. Colored Intersection Triangles from Delaunay Deconstruction.

the MATLAB® built-in function "inpolygon" is utilized to restrict the robot's trajectory points to the obtained open space described by equation (10). The (*safe_flag*) is triggered, indicating an obstacle-free trajectory, as shown in lines (8)-(12).

$$\min V_{Pol} \leq P_m(t) \leq \max V_{Pol}, \qquad \forall t \in [0, t_f] \tag{10}$$

where V_{Pol} are vertices of the robot's open space polygon (Pol).

Algorithm 1 Robot's Trajectory Safety Check Algorithm

```
    Input: $Li_{read}$, $R_{max}$, and $P_m(t)$
    Output: $safe\_flag$
 1: $DT \leftarrow$ delaunayTriangulation($Li_{read}, R_{max}$)
 2: $P_m(0) = C_0, \quad P_m(t_f) = C_g$
 3: $Intersect\_tri \leftarrow DT$.Triangles $\cap$ line($C_0, C_g$)
 4: $Pol \leftarrow$ union( $Intersect\_tri$)
 5: $V_{Pol} \leftarrow Pol$.Vertices
 6: $In \leftarrow$ inpolygon($P_m(t), V_{Pol}$)
 7: $N_{out} \leftarrow$ numel($\sim In$)
 8: if $N_{out} = 0$ then
 9:     $safe\_flag \leftarrow 1$
10: else
11:     $safe\_flag \leftarrow 0$
12: end if
13: return $safe\_flag$
```

B. CONTINUOUS OPTIMIZATION PROBLEM

We aim to optimize the robot's trajectory contained within its open space, which is obstacle-free, as discussed in the previous sub-section, while considering the robot's kinodynamic limits. This problem can be formulated as an optimization problem with an objective function (f) that is subject to boundary conditions and path constraints, as shown in equation (11). In this formulation, the objective function is a linear combination of the robot's navigation time and the length of its trajectory, with weights β_1 and β_2, respectively, such that $\beta_1 + \beta_2 = 1$. The values of these weights assign relative importance to the two aspects to evaluate different navigation strategies. The robot's initial and final states ($S_0 = \{C_0, \psi_0, v_0, \omega_0\}$ and $S_f = \{C_g, \psi_f, v_f, \omega_f\}$) including its position, orientation, forward velocity, and angular velocity are imposed as boundary conditions. The forward and angular velocity limits, maximum curvature, and sideslip prevention check are imposed as trajectory constraints.

$$\begin{aligned}
&Minimize \quad f = \beta_1 * \int_0^{t_f} dt + \beta_2 * \int_0^{t_f} \frac{dP(t)}{dt} dt\\
&\text{Subject to: } \textbf{Boundary conditions}\\
&\qquad P(0) = C_0, \quad P(t_f) = C_g\\
&\qquad tan(\psi(0)) = tan(\psi_0), \quad tan(\psi(t_f)) = tan(\psi_f)\\
&\qquad v^2(0) = v_0^2, \quad v^2(t_f) = v_f^2\\
&\qquad \omega(0) = \omega_0, \quad \omega(t_f) = \omega_f\\
&\qquad \textbf{Path constraints}\\
&\qquad v^2(t) \leq v_{max}^2\\
&\qquad -\omega_{max} \leq \omega(t) \leq \omega_{max}\\
&\qquad \frac{a^t(t)^2}{(a^t_{max})^2} + \frac{a^r(t)^2}{(a^r_{max})^2} \leq 1\\
&\qquad k(t)^2 \leq k_{max}^2\\
&\qquad \min V_{Pol} \leq P(t) \leq \max V_{Pol}
\end{aligned} \tag{11}$$

where C_0 and C_g are the robot's start and goal positions, ψ_0 and ψ_f are the robot's initial and final orientations, v_0 and v_f are the robot's initial and final forward velocity, ω_0 and ω_f are the robot's initial and final angular velocity, v_{max} and ω_{max} are the robot's maximum forward and angular velocity, a^t_{max} and a^r_{max} are the robot's maximum tangential and radial acceleration and $k_{\max}$ is the robot's curvature limit.

C. DISCRETIZED OPTIMIZATION PROBLEM

Once the problem is formulated in continuous form, the next step is to express it as a constrained numerical optimization problem that can be solved using one of the off-the-shelf optimization algorithms. In this work, we use a Bézier curve of degree m to approximate the robot's trajectory, with $(m + 1)$ control points ($\bar{p}_{0,m}$,$\bar{p}_{1,m}$, $\ldots$, $\bar{p}_{m,m}$), where $\bar{p}_{0,m}$ and $\bar{p}_{m,m}$ correspond to the start and goal positions of the robot, respectively. The objective function is a weighted combination of the robot's navigation time and the length of its trajectory. The length of a Bézier curve trajectory is approximated by accumulating incremental Euclidean distances between successive points along it rather than integrating trajectory segments in the continuous form. In this work, we split the trajectory into N_a segments and then calculate the sum of these straight-line distances between each pair of points. This sum gives us an approximation of the curve's total length. This approximation method is explicitly detailed in the objective function's second part, as shown in equation (12). After defining the objective function, it is important to define the tuning parameters that change during optimization to get the optimal solution, while satisfying the constraints. In this work, the tuning parameters are the control points for the position Bézier curve other than the endpoints and the robot's navigation time (t_f). The optimization problem involves tuning the x and y components of these control points, as well as the robot's navigation time, resulting in the state vector ($x = [\bar{p}_{1,m}, \bar{p}_{2,m}, \ldots, \bar{p}_{m-1,m}, t_f]$). The control points of the kinodynamic terms depend on these tuning parameters, according to equation (2), and will automatically be tuned to ensure compliance with the robot's kinodynamic limits.

$$\begin{aligned}
&\underset{\bar{p}_{j,m},t_f}{\text{minimize}}\\
&\qquad \beta_1 * t_f + \beta_2 * \sum_{i=1}^{N_a} P_m(i\frac{t_f}{N_a}) - P_m((i-1)\frac{t_f}{N_a})^2,\\
&\qquad\qquad j \in \{1, \ldots, m-1\}\\
&\text{subject to } \textbf{Boundary conditions}\\
&\qquad P_m(0) = C_0, \quad P_m(t_f) = C_g\\
&\qquad tan(\psi_m(0)) = tan(\psi_0), \quad tan(\psi_m(t_f)) = tan(\psi_f)\\
&\qquad v_m(0)^2 = v_0^2, \quad v_m(t_f)^2 = v_f^2\\
&\qquad \omega_m(0) = \omega_0, \quad \omega_m(t_f) = \omega_f\\
&\qquad \textbf{Path constraints} \quad \forall t \in [0, t_f]\\
&\qquad v_m(t)^2 \leq v_{max}^2\\
&\qquad -\omega_{max} \leq \omega_m(t), \leq \omega_{max}\\
&\qquad \frac{a^t_m(t)^2}{(a^t_{max})^2} + \frac{a^r_m(t)^2}{(a^r_{max})^2} \leq 1\\
&\qquad k_m(t)^2 \leq k_{max}^2\\
&\qquad \min V_{Pol} \leq P_m(t) \leq \max V_{Pol}
\end{aligned} \tag{12}$$

Numerical optimization methods require an initial estimate of the states. In this study, the robot's initial trajectory is assumed to be a straight line connecting the start and goal positions. In the initial estimate, the robot is assumed to be oriented towards the goal and moving at the maximum velocity. Therefore, the robot's navigation time is initialized as the ratio of the Euclidean distance between the start and goal points to the robot's maximum velocity. The second and second-to-last control points, represented as $\bar{p}_{1,m}$ and $\bar{p}_{m-1,m}$, are initialized to satisfy the robot's initial and final orientations, as described in equation (13). The remaining control points, $\bar{p}_{2,m}, \ldots, \bar{p}_{m-2,m}$, are initially placed equidistant between the second and second-to-last control points. The initial estimate is tuned during the optimization process to reach an optimal feasible solution that satisfies all the constraints.

$$\begin{aligned}
\bar{p}_{1,m} &= \bar{p}_{0,m} + \frac{v_0 * t_f}{m} \angle(\psi_0)\\
\bar{p}_{m-1,m} &= \bar{p}_{m,m} - \frac{v_f * t_f}{m} \angle(\psi_f)
\end{aligned} \tag{13}$$

D. PERFORMANCE METRICS

Various performance metrics are used to assess a trajectory planner's performance, such as navigation time, trajectory length, trajectory smoothness, and target deviation. Multiple performance metrics will better capture the overall system quality [25]. In this work, we consider three metrics to compare BTP with some existing trajectory planners: trajectory smoothness, navigation time, and target deviation. The trajectory smoothness can be measured by Bending Energy (BE), which refers to the energy needed to bend something into a desired shape [26]. Mathematically, it is calculated by adding up the squared curvature values at each point along a robot's trajectory as indicated in equation (14). In the context of mobile robots and trajectory planning, C^0, C^1, and C^2 continuity refer to different levels of smoothness in the robot's trajectory or path. C^0 continuity represents a lower level of smoothness, with continuous position but potentially abrupt changes in velocity and acceleration. C^1 continuity refers to continuous position and velocity profiles. Finally, C^2 continuity represents a higher level of smoothness, with continuous positions, velocities, and accelerations, leading to a more stable and smooth robot trajectory without any jerky or sudden changes in direction or velocity. C^0 continuous trajectories have large BE values due to infinite curvature in the robot's halts, C^1 continuous trajectories have medium values, and C^2 continuous trajectories have small ones with smooth curvature.

$$BE = \frac{1}{N_a}\sum_{i=1}^{N_a} k^2(t_i) \tag{14}$$

where N_a represents the total number of points that form the robot's trajectory, and $k(t_i)$ denotes the curvature at each specific point along the robot's trajectory.

The navigation time is the time that the robot takes to reach the target configuration from an initial configuration. Finally, the deviation of the robot's target refers to the displacement from the exact target position. This is crucial, especially when the robot's movement is part of a larger task that involves additional actions. The closer the robot is to the intended target, the more accurately it can perform subsequent tasks. The optimum trajectory should reach the target precisely (pass through the target position with zero target deviation) with zero curvature in the minimum possible time.

IV. RESULTS AND DISCUSSION

In this section, we apply the proposed BTP optimization algorithm to various scenarios while also benchmarking its performance against other trajectory planners. This work uses the MATLAB® NLP function called "fmincon", a gradient-based method that finds a constrained minimum of a scalar function while considering linear/nonlinear equality/inequality constraints and lower and upper bounds of the states. The constraints on the robot's movement are represented in the optimization problem as nonlinear equality constraints for boundary conditions and nonlinear inequality constraints for path constraints. The optimization problem is solved using the Sequential Quadratic Programming (SQP) method.

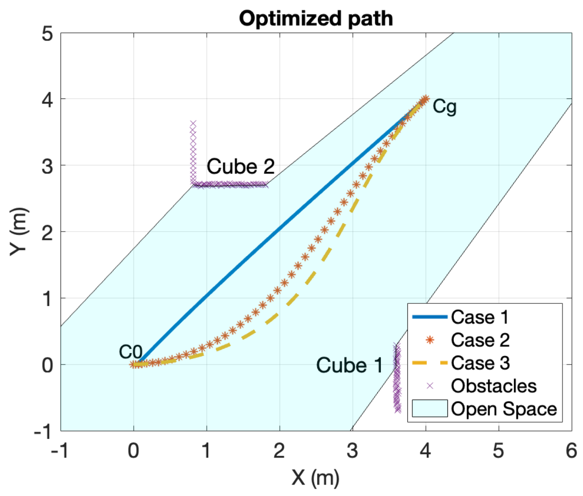


FIGURE 4. Robot's trajectory in Example 1.

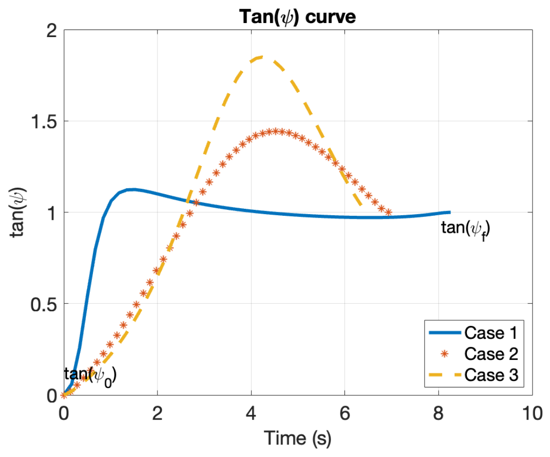


FIGURE 5. Tangent of robot's orientation in Example 1.

The Pioneer P3-DX differentially-steered robot equipped with a Hokuyo LiDAR sensor is used as the test bed in both simulated Gazebo environments and in real-world situations. The numerical values of the robot's kinodynamic limits are obtained from the Pioneer P3-DX manual [27]. BTP initially employs a fifth-degree Bézier curve for the trajectory to find the optimal solution. If convergence is not realized using this degree, the algorithm incrementally increases it until a solution is reached. BTP consistently achieved convergence using the fifth-degree Bézier curve for the case studies investigated. For trajectory length calculation in the objective function, the robot's trajectory is segmented into 50 equal parts ($N_a = 50$).

A. TEST EXAMPLES

Two scenarios are used to evaluate BTP: The first scenario assesses different weight values in the objective function,

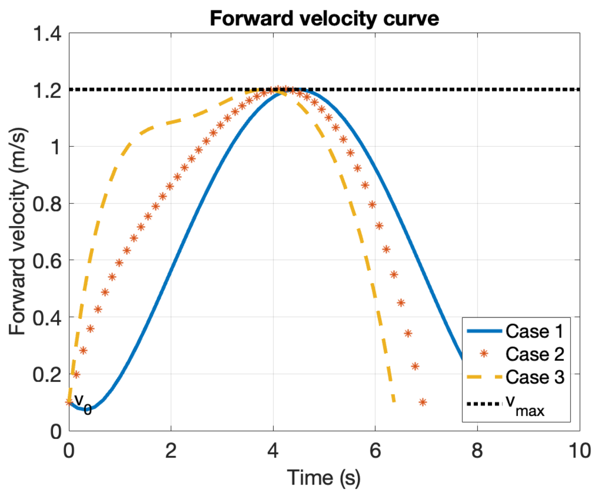


FIGURE 6. Robot's forward velocity in Example 1.

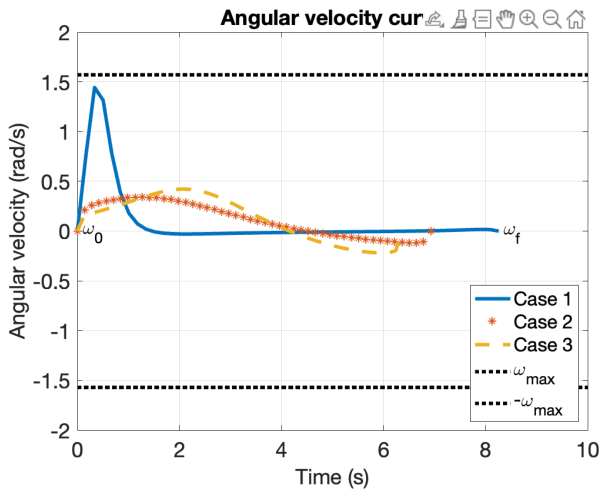


FIGURE 7. Robot's angular velocity in Example 1.

TABLE 1. Example 1 results.

Case	Weights	#iterations	l (m)	t_{nav}(s)
1	β_1=0,β_2=1	22	**5.67**	8.26
2	β_1=0.2,β_2=0.8	11	5.82	6.93
3	β_1=1,β_2=0	7	5.95	**6.49**

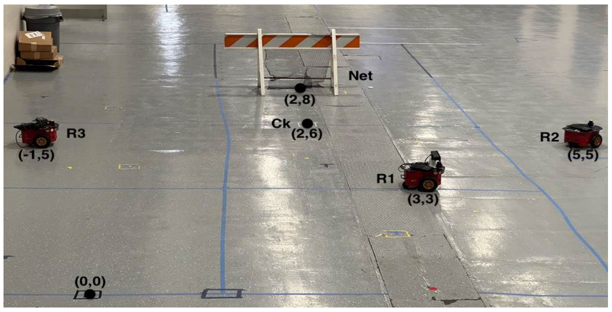


FIGURE 8. Soccer game environment of Example 2.

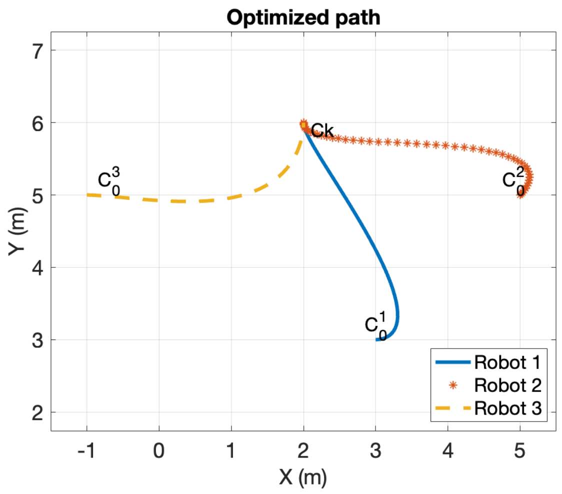


FIGURE 9. Robot's trajectory in the Example 2 soccer game.

while the second scenario involves the selection of the best-positioned robot from a group of three to kick a ball in a soccer game. This robot will be the quickest to get to the ball.

Example 1: Consider a Pioneer P3-DX robot navigating from an initial configuration $S_0 = (C_0, \psi_0, v_0, w_0) = ([0, 0], 0, 0.1, 0)$ to a goal configuration $S_f = (C_g, \psi_f, v_f, w_f) = ([4, 4], \frac{\pi}{4}, 0.1, 0)$, while satisfying the kinodynamic constraints, in the environment illustrated in Figure 1. To evaluate the effectiveness of the proposed method, three cases with different weight values for the objective function are tested, each demonstrating a different optimization strategy. Given the same initial and final conditions, case 1 ($\beta_1 = 0$, $\beta_2 = 1$) represents the shortest distance, while case 3 ($\beta_1 = 1$, $\beta_2 = 0$) represents the minimum time optimization problems respectively. Finally, case 2 ($\beta_1 = 0.2$, $\beta_2 = 0.8$) represents a weighted combination between the fastest and shortest trajectory with a weight ratio of 1:4. For each case, the number of iterations required for the algorithm to converge to an optimal solution, the length of the robot's trajectory (l), and the navigation time required to follow the obtained trajectory (t_{nav}) are recorded, as shown in Table 1. As expected, the results show that the robot reaches the goal configuration in minimum time for case 1 and minimum distance for case 3. Figure 4 shows the robot's generated trajectory for each case within the open space in cyan color, with every trajectory passing through the start and goal positions as expected. Figure 5 shows the tangent of the robot's orientation for each case, where the robot's initial and final orientations are satisfied. Figures 6 and 7 show the robot's forward and angular velocities for each case, where the robot's initial and final velocities are satisfied, and the velocity limits are not exceeded over the entire trajectory.

Example 2: In a soccer game, three robots are positioned at $C_0 = [3, 3], [5, 5], [-1, 5]$, with initial orientations

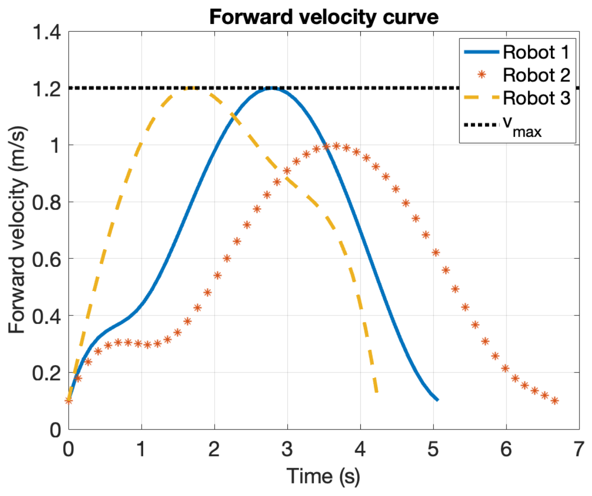


FIGURE 10. Robot's forward velocity in the Example 2 soccer game.

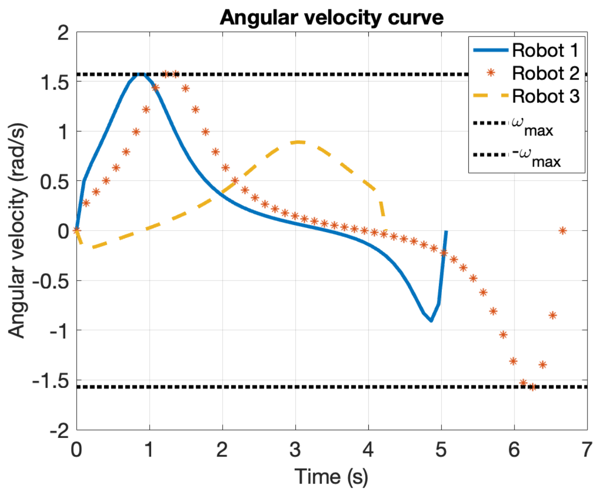


FIGURE 11. Robot's angular velocity in the Example 2 soccer game.

$\psi_0 = [0, \frac{\pi}{4}, 0]$ respectively, as indicated in Figure 8. The objective is for a robot to score a goal by kicking the ball into the net located at [2, 8] at an angle $\psi_k = \frac{\pi}{2}$. The kick location is given by $C_k = [2, 6]$, and all three robots are equidistant from this location. Turning in place is prohibited during the game to satisfy trajectory smoothness as well as immediately not contending for optimality. BTP determines the robot that should execute the kick based on the navigation time to the kick position from the initial position. Figure 9 illustrates the three robots' optimal trajectories, where they all pass through the start and goal positions. The robots' forward and angular velocity profiles are shown in Figures 10 and 11, respectively. As indicated in Table 2, robot "3" should be selected to kick the ball as it is the fastest to reach the kick position with the desired orientation. This example is an elementary practical implementation of BTP in determining robot soccer strategy, obviously with significance beyond a soccer game.

TABLE 2. Example 2 soccer game results.

Robot	ψ_0	# iterations	l (m)	t_{nav} (s)
1	0	16	3.51	5.07
2	$\frac{\pi}{4}$	12	3.67	6.67
3	0	9	3.58	**4.24**

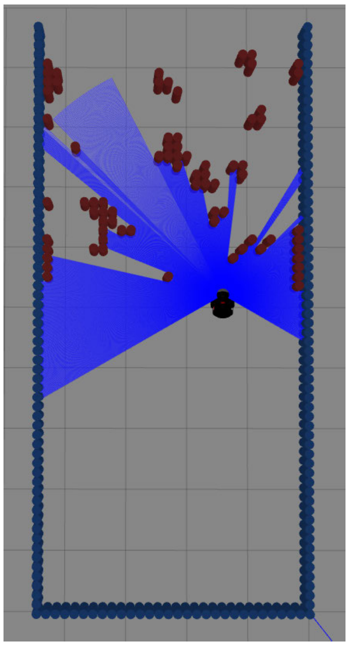


FIGURE 12. World 4 in BARN dataset [33].

B. COMPARSION OF BTP WITH THE EXISTING TRAJECTORY PLANNERS

The performance of BTP is benchmarked against the other trajectory planners based on three metrics: navigation time, trajectory smoothness, and target deviation. Five trajectory planners are considered in this benchmark: Learning from Hallucinations (Lfh) [28], Learning from learned Hallucination (Lflh) [29], Dynamic-Window Approach (DWA) [30], Elastic Band (EBand) [31], and end-to-end machine learning method (e2e) [32]. The environment used for this comparison is the fourth environment of the Benchmark for Autonomous Robot Navigation (BARN) dataset [33], as shown in Figure 12. Robots that ran Lfh and e2e collided with obstacles in the test environment. Thus, the performance of BTP is benchmarked against the remaining three trajectory planners: DWA, EBand, and Lflh, with detailed outcomes presented in Table 3. BTP surpasses the considered trajectory planners by generating the fastest path, while considering the constraints imposed by the robot's limitations. BTP navigated this particular environment using two fifth-degree Bézier curves as indicated in Figure 13. DWA and Lflh exhibited numerous oscillations in forward movement during testing, alternating between advancing and retreating, as illustrated in the forward velocity graph (Figure 14). This behavior significantly increased the time it took for DWA and Lflh

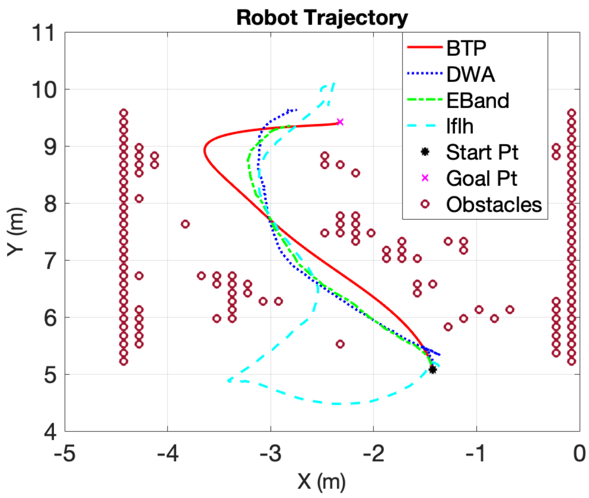


FIGURE 13. Robot's trajectory in World 4 of BARN dataset [33].

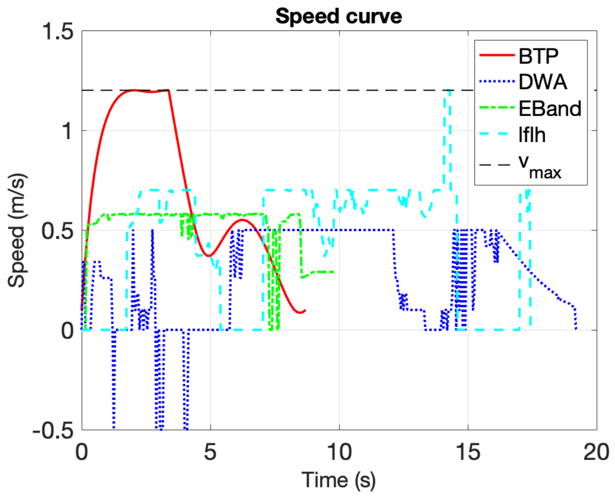


FIGURE 14. Robot's forward velocity in World 4 of BARN dataset.

TABLE 3. Comparison between BTP and other trajectory planners.

Algorithm	t_{nav} (s)	t_{red} (%)	Smoothness ($\frac{rad^2}{m^2}$)	Target deviation (m)
BTP	**8.70**	**-**	**10.59**	**0**
DWA	19.20	54.67%	2903.42	0.56
EBand	9.75	10.74%	409.67	0.50
lflh	17.43	50.08%	3502.52	0.30

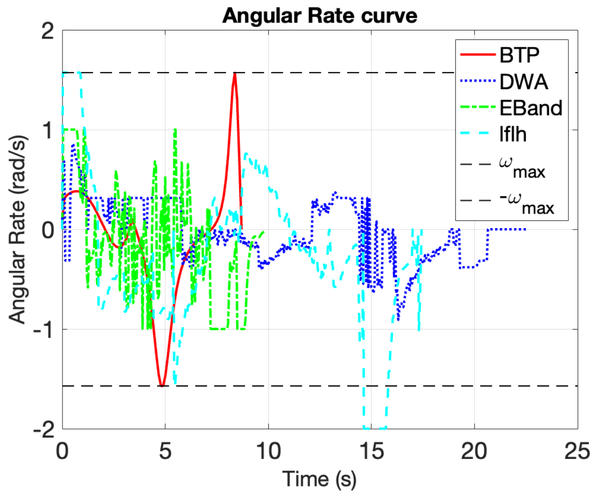


FIGURE 15. Robot's angular Velocity in World 4 of BARN dataset.

to reach the goal. The percentage of reduction in the navigation time of the tested trajectory planners compared to BTP, as a reference, is documented within column "t_{red}" of Table 3.

When the robot was run with DWA, Lflh, and EBand, it experienced several halts, causing the curvature to become infinite. These infinite curvature values were substituted with a value of 100 in numerical analyses to keep them in the comparison. As indicated in Table 3, BTP exhibited the smoothest trajectory with the lowest BE value, followed by EBand, DWA, and Lflh. BTP ensures C^2 continuity, whereas other trajectory planners only account for C^0 continuity. This implies that the robot's motion exhibits jerky or sudden changes in direction and velocity as evidenced by the displayed fluctuations in angular velocity of the tested trajectory planners (see Fig 15). In contrast, BTP maintained consistent forward and angular velocities, Figure 14 and 15, leading to a quicker and smoother navigation to the goal than the other trajectory planners. In addition to smoothness, BTP consistently passes directly through the designated goal position unlike the other trajectory planners, which aim to stop the robot within a 1-meter radius of the goal position, resulting in a slight deviation from the target.

C. CONVERGENCE CRITERIA AND COMPUTING EFFICIENCY

All the simulation scenarios were successfully implemented in real-world experimentation. The recorded navigation time for robots only exhibited some dilation by 60.83% because of variable ground friction between the simulation and real-world experimentation and speed limitation in the actual robot.

BTP has adopted two convergence criteria: optimality Tolerance and constraint tolerance. Optimality tolerance monitors variations in the objective function throughout iterative processes, prompting the algorithm's termination when the change in the objective function falls below this specified tolerance threshold. On the other hand, constraint tolerance specifies the permissible deviation within which the constraints are deemed satisfied. For our work, both optimality and constraint tolerance are established at 1e-6. If BTP fails to converge with the current degree under these convergence criteria, BTP incrementally increases the degree of the Bézier curve by one until convergence.

We have employed factorization techniques to mitigate computational burdens and accelerate computations to optimize computational efficiency and scalability. Additionally, we have integrated the deCasteljau method into BTP to decrease computational time and ensure numerical stability throughout the calculation processes.

V. CONCLUSION

This paper presents a novel method denoted as "BTP" to generate optimal trajectories based on Bézier curves for a differential robot navigating an unknown environment. The Bézier curve structure facilitates the incorporation of the robot's kinodynamic limits, while also guaranteeing continuous curvature. A weighted objective function of the robot's navigation time and trajectory length is formulated to permit different optimization strategies. Furthermore, a Delaunay Triangulation-based algorithm is incorporated to ensure the trajectory is within the robot environment's open space. Unlike existing Bézier curve-based work with a fixed degree, BTP uses a matrix formulation that facilitates the dynamic change of Bézier curve degree during the optimization process to obtain convergence. The effectiveness of the proposed method has been evaluated with various scenarios in the Gazebo simulator as well as on a real Pioneer P3-DX robot. BTP outperformed the considered trajectory planners based on navigation time, smoothness, and target precision. Although BTP can generate smooth and continuous trajectories, it is unsuitable for online navigation due to its high computational overhead. This inhibits real-time execution, particularly when the robot is moving at higher speeds in a cluttered environment. Instead, the proposed algorithm can serve as a benchmark trajectory for training other real-time (online) trajectory planners to improve their overall performance.

Future research will extend the work to multi-robot scenarios. This expansion necessitates the integration of measures addressing robot-to-robot collision. As the number of robots in the mission escalates, the number of tuning parameters will exponentially evolve, which, in turn, will require more computational resources and processing time. The matrix formulation within the BTP offers a systematic and practical framework for implementing this expansion.

REFERENCES

[1] B. Yang, W. Li, J. Wang, J. Yang, T. Wang, and X. Liu, "A novel path planning algorithm for warehouse robots based on a two-dimensional grid model," *IEEE Access*, vol. 8, pp. 80347–80357, 2020.

[2] J. Lee, G. Park, I. Cho, K. Kang, D. Pyo, S. Cho, M. Cho, and W. Chung, "ODS-Bot: Mobile robot navigation for outdoor delivery services," *IEEE Access*, vol. 10, pp. 107250–107258, 2022.

[3] M. B. Alatise and G. P. Hancke, "A review on challenges of autonomous mobile robot and sensor fusion methods," *IEEE Access*, vol. 8, pp. 39830–39846, 2020.

[4] A. Kamalova, K. D. Kim, and S. G. Lee, "Waypoint mobile robot exploration based on biologically inspired algorithms," *IEEE Access*, vol. 8, pp. 190342–190355, 2020.

[5] E. Ackerman. (2015). *Startup Developing Autonomous Delivery Robots That Travel on Sidewalks*. IEEE Spectrum. [Online]. Available: https://spectrum.ieee.org/automaton/robotics/industrial-robots/starship-technologies-autonomous-ground-delivery-robots

[6] J. Zhang, "A collision-free 3D path planning strategy for mobile robots," in *Proc. Austral. New Zealand Control Conf. (ANZCC)*, Nov. 2019, pp. 1–4.

[7] H.-Y. Hsueh, A.-I. Toma, H. A. Jaafar, E. Stow, R. Murai, P. H. J. Kelly, and S. Saeedi, "Systematic comparison of path planning algorithms using PathBench," *Adv. Robot.*, vol. 36, no. 11, pp. 566–581, Jun. 2022.

[8] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Mar. 1968.

[9] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Dept. Comput. Sci., Iowa State Univ., Ames, IA, USA, Tech. Rep. 9811, 1998.

[10] H. Kong, C. Yang, G. Li, and S.-L. Dai, "A sEMG-based shared control system with no-target obstacle avoidance for omnidirectional mobile robots," *IEEE Access*, vol. 8, pp. 26030–26040, 2020.

[11] U. Orozco-Rosas, K. Picos, J. J. Pantrigo, A. S. Montemayor, and A. Cuesta-Infante, "Mobile robot path planning using a QAPF learning algorithm for known and unknown environments," *IEEE Access*, vol. 10, pp. 84648–84663, 2022.

[12] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 604–611, Apr. 2020.

[13] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *Amer. J. Math.*, vol. 79, no. 3, p. 497, Jul. 1957.

[14] X. Li, X. Gao, W. Zhang, and L. Hao, "Smooth and collision-free trajectory generation in cluttered environments using cubic B-spline form," *Mechanism Mach. Theory*, vol. 169, Mar. 2022, Art. no. 104606.

[15] K. Renny Simba, N. Uchiyama, and S. Sano, "Real-time smooth trajectory generation for nonholonomic mobile robots using Bézier curves," *Robot. Comput.-Integr. Manuf.*, vol. 41, pp. 31–42, Oct. 2016.

[16] C. Chen, Y. He, C. Bu, J. Han, and X. Zhang, "Quartic Bézier curve based trajectory generation for autonomous vehicles with curvature and velocity constraints," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 6108–6113.

[17] L. Zhang, L. Sun, S. Zhang, and J. Liu, "Trajectory planning for an indoor mobile robot using quintic Bezier curves," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2015, pp. 757–762.

[18] A. V. Rao, "Trajectory optimization: A survey," in *Optimization and Optimal Control in Automotive Systems*. Cham, Switzerland: Springer, 2014, pp. 3–21.

[19] Y. Kim and B. K. Kim, "Time-optimal trajectory planning based on dynamics for differential-wheeled mobile robots with a geometric corridor," *IEEE Trans. Ind. Electron.*, vol. 64, no. 7, pp. 5502–5512, Jul. 2017.

[20] C. Kielas-Jensen and V. Cichella, "BeBOT: Bernstein polynomial toolkit for trajectory generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 3288–3293.

[21] G. Klancar and S. Blažic, "Optimal constant acceleration motion primitives," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8502–8511, Sep. 2019.

[22] R. T. Farouki, "The Bernstein polynomial basis: A centennial retrospective," *Comput. Aided Geometric Des.*, vol. 29, no. 6, pp. 379–419, Aug. 2012.

[23] W. Boehm and A. Müller, "On de Casteljau's algorithm," *Comput. Aided Geometric Des.*, vol. 16, no. 7, pp. 587–605, 1999.

[24] Z. Liu, H. Liu, Z. Lu, and Q. Zeng, "A dynamic fusion pathfinding algorithm using Delaunay triangulation and improved A-star for mobile robots," *IEEE Access*, vol. 9, pp. 20602–20621, 2021.

[25] N. Muñoz, J. Valencia, and N. Londono, "Evaluation of navigation of an autonomous mobile robot," in *Proc. Workshop Perform. Metrics Intell. Syst.*, 2007, pp. 15–21.

[26] D. Calisi, L. Iocchi, and D. Nardi, "A unified benchmark framework for autonomous mobile robots and vehicles motion algorithms (MoVeMA benchmarks)," in *Proc. Workshop Experim. Methodol. Benchmarking Robot. Res.*, 2008.

[27] A. Robotics, "Pioneer 3 operations manual," MobileRobots Inc, Amherst, NH, USA, Tech. Rep. 3, Tech. Rep., 2006.

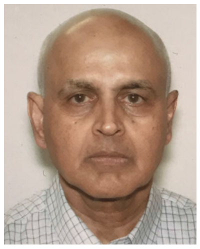
[28] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Toward agile maneuvers in highly constrained spaces: Learning from hallucination," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1503–1510, Apr. 2021.

[29] Z. Wang, X. Xiao, A. J. Nettekoven, K. Umasankar, A. Singh, S. Bommakanti, U. Topcu, and P. Stone, "From agile ground to aerial navigation: Learning from learned hallucination," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 148–153.

[30] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.
[31] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Proc. IEEE Int. Conf. Robot. Automat.*, Feb. 1993, pp. 802–807.
[32] S. Li, H.-T. Nguyen, and C. C. Cheah, "A theoretical framework for end-to-end learning of deep neural networks with applications to robotics," *IEEE Access*, vol. 11, pp. 21992–22006, 2023.
[33] D. Perille, A. Truong, X. Xiao, and P. Stone, "Benchmarking metric ground navigation," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot. (SSRR)*, Nov. 2020, pp. 116–121.

AMNA MAZEN received the B.S. and M.Sc. degrees in electrical and power engineering from Fayoum University, Faiyum, Egypt, in 2013 and 2018, respectively. She is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Detroit Mercy, Detroit, MI, USA. Since 2019, she has been a Teaching and Research Assistant of electrical and computer engineering with the University of Detroit Mercy. Her research interests include robotics, deep learning, and AI.

MARIAM FAIED (Senior Member, IEEE) was a Visiting Research Professor with the University of Michigan, Ann Arbor. During the Ph.D., she conducted research with the Aerospace, Robotics, and Control (ARC) Laboratory, University of Michigan, on collaborative control of multiple unmanned vehicles, adversarial strategies, and advanced mission planning. In 2010, she was a Postdoctoral Researcher with the University of Michigan. She supervised undergraduate, master's, and Ph.D. students, and served as the Ph.D. Committees' Member. She was the Interim Director of the ARC Laboratory for six months. She was appointed as an Assistant Professor with Fayoum University, Egypt, in 2012. She developed the Mechatronics Engineering Department Curriculum, which was audited by the Egyptian Supreme Council of Universities and approved. In 2013, she was selected to serve as the Mechatronics Program Chair of Fayoum University. She is currently an Associate Professor with the University of Detroit Mercy. Her research interests include design, analysis, and optimization of planning and control algorithms for robotics. She received the Outstanding Program Coordinator Honor and the Best Paper in Session Award at AIAA GNC conference.



MOHAN KRISHNAN (Life Senior Member, IEEE) is currently a Professor Emeritus of electrical and computer engineering with the University of Detroit Mercy. His area of specialization is autonomous mobility robotics. In particular, he has worked on problems involving the use of computational intelligence techniques in autonomous vehicle navigation and intelligent control, graph-theoretic approaches to robot path and motion planning, and modeling of mechatronic systems. He was a Fulbright U.S. Scholar, who participated collaboratively on teaching and research activities in the area of mobile robotics with the Faculty, Department of Electrical Engineering, University of Ljubljana, Slovenia, from October 2021 to April 2022.