Mobile Robot Motion Planning based on Synthesized Optimal Control with Particle Swarm Optimization*

Elizaveta Shmalko[1] and Konstantin Yamshanov[2]

Abstract— This paper presents a new motion planning algorithm for mobile robots that uses synthesized optimal control and time-adaptive particle swarm optimization (PSO). The proposed approach aims to address the challenge of creating collision-free paths in complex environments with static obstacles, while optimizing both spatial and temporal constraints. The algorithm dynamically adjusts the intermediate waypoints and time budget to minimize a combined cost function that includes factors such as positional error, travel time, and obstacle avoidance costs.

The effectiveness of the proposed algorithm is demonstrated through simulations in Gazebo using the ROSBot 2.0 model. The algorithm successfully navigates the robot through the optimized waypoints in scenarios with obstacles, avoiding collisions and minimizing the travel time. In environments without obstacles, the trajectory is collapsed into a direct path, significantly reducing the total travel time. The results highlight the adaptability and efficiency of the algorithm, making it a promising solution for real-world applications. Key contributions of this work include the integration of optimal control with time-adaptive particle swarm optimization (PSO), the dynamic adjustment of waypoints and time constraints, and the demonstration of algorithm performance in both complex and simplified environments.

I. INTRODUCTION

The purpose of motion planning of a mobile robot is to safely without collisions move the robot to accomplish a given goal with the best value of the quality criteria. Today, there are many motion planning algorithms, and the choice of the appropriate one depends on the requirements for the result. For example, one can use existing application software packages, such as Kineo CAM (Kineo Computer Aided Motion) [1], OMPL (Open Motion Planning Library) [2], CUIK Suite [3]. However, the mathematical arsenal of the mentioned packages basically does not assume the definition of extended collision-free trajectories with a complex structure of the phase space. Such problems require a more universal mathematical apparatus.

Ideally, we desire to perform the motion planning task optimally as solutions to problems for maximum or minimum. Leonard Euler said: "Since the structure of the universe is perfect and created by a wise creator, nothing arises in the universe in which one could not see the meaning of some maximum or minimum." For the same reason, developers of autonomous robots strive to perform the task of controlling the robot's motion optimally, based on the minimization of a given quality criterion, for example, by length, execution time or energy consumption.

The optimal feasible path, that takes into account the kinematic or dynamic features of the robot, must be generally obtained as a solution of the optimal control problem which is fundamental in the control theory. Main analytical methods based on Pontryagin maximum principle [4], [5] and Bellman's dynamic programming [6] for solving this problem cover only a some class of problems with a certain type of quality functionals and a certain type of phase space constraints.

To solve it in practical applications, various parametrization techniques are usually used. The most general approach to parameterization of control is the so-called direct approach [7], which consists, for example, of a piecewise polynomial approximation of the control function and calculation of the corresponding optimal parameters [8]. Here optimality is compensated by computational complexity [12]. We must be ready to accept increased computational complexity if we demand strong optimal motion laws from our planner.

The simplest and at the same time most common approach to parameterization is to use spatial decomposition methods, such as regular spatial decomposition with partitioning of space by a grid with a fixed cell size [9], [10], [11], which involves dividing the phase space, free of phase constraints, into a set of simple regions using certain decomposition methods, determining the adjacency of regions, and forming a connectivity graph, which can then be used for navigation, or methods based on route networks that combine conflict-free transitions [12], [13]. In fact, these approaches implement a scheme for reducing a computationally complex optimal control problem in Euclidean space to a typical pathfinding problem in a graph. But at the same time, the question of obtaining further implementations of the robot's movement along the obtained optimal trajectory and creating the necessary controllers remains open.

This problem in the general mathematical formulation is a problem of control system synthesis which involves finding optimal control as a function of the object state. The synthesis of optimal control from a mathematical point of view is a nonlinear programming problem in

*This scientific research was partially supported by Educational Organizations in 2024-2026 Project under Grant FSFS-2024-0012

[1]Elizaveta Shmalko is with the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 119333, Vavilova str. 44, Moscow, Russia e.shmalko@gmail.com

[2]Konstantin Yanshanov is with the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 119333, Vavilova str. 44, Moscow, Russia yam.kostya@yandex.ru

function spaces and, in the general case, it has no generic solution methods. Therefore, in practice, engineers usually decompose the problem of general synthesis into problems of planning the optimal path, distribution of stabilization points with the speed profile on the given path and synthesis of the stabilization system of longitudinal and angular motion [14]. However, with this approach, additional questions related to determining the optimal speed profile and the optimal distribution of stabilization points on the trajectory, taking into account the initially specified quality functional, will remain open. And these issues in applied problems are usually solved manually at the discretion of the developer so far.

In this paper, we propose an approach based on the principle of synthesized optimal control [15], [16] that involves a two-stage approach to planning the motion of an autonomous robot. At the first stage, a stabilization system is introduced into the control object, ensuring stabilization of the object at a certain point in the state space. Then, the main idea of the approach is to search for a control function in which the system of differential equations describing the object will always have a stable equilibrium point in the state space, ensured by the stabilization system introduced at the first stage. In this case, the control function contains parameters that affect the position of the equilibrium point. Consequently, the object is controlled by changing the position of the equilibrium point, namely, its optimal location at specified intervals. The paper presents the application of this approach to control a mobile robot in the Gazebo simulation environment ROS and proposes time-adaptive Particle Swarm Optimization for robotic motion planning.

II. PRINCIPLE OF SYNTHESIZED OPTIMAL CONTROL

According to the principle of synthesized optimal control, it is necessary to find a control function

$$\mathbf{u} = \mathbf{g}\left(\mathbf{x}, \mathbf{q}^*\right), \tag{1}$$

such that the system of differential equations describing the model of the control object

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{2}$$

where $\mathbf{x}$ is a state vector, $\mathbf{u}$ is a control vector, $\mathbf{q}^*$ is a parameter vector,

$$\begin{array}{rcl} \mathbf{x} & = & [x_1 \dots x_n]^T, \\ \mathbf{u} & = & [u_1 \dots u_m]^T,\ m \leq n, \\ \mathbf{f}(\mathbf{x}, \mathbf{u}) & = & [f_1(\mathbf{x}, \mathbf{u}) \dots f_n(\mathbf{x}, \mathbf{u})]^T, \end{array}$$

always has a stable equilibrium point in the state space.

The control function (1) ensures that the system

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, \mathbf{g}\left(\mathbf{x}, \mathbf{q}^*\right)\right)$$

has an equilibrium point

$$\mathbf{f}\left(\mathbf{x}^*\left(\mathbf{q}^*\right), \mathbf{g}\left(\mathbf{x}^*\left(\mathbf{q}^*\right), \mathbf{q}^*\right)\right) = 0,$$

where $\mathbf{x}^*\left(\mathbf{q}^*\right)$ is the vector of coordinates of the equilibrium point, depending on the parameter vector $\mathbf{q}^*$.

At the same time, the value of the parameter vector affects the position of the equilibrium point in the state space. Thus, the following optimal control problem is solved:

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, \mathbf{g}\left(\mathbf{x}, \mathbf{x}^*\right)\right), \tag{3}$$

$$\mathbf{x}^* \in X_1 \subseteq \mathbb{R}^n, \tag{4}$$

$$\mathbf{x}(0) = \mathbf{x}^0, \tag{5}$$

$$\mathbf{x}(T) = \mathbf{x}^f, \tag{6}$$

$$J = \int_0^T f_0\left(\mathbf{x}, \mathbf{g}\left(\mathbf{x}, \mathbf{x}^*\right)\right) dt \to \min_{\mathbf{x}^* \in X_0}. \tag{7}$$

It is necessary to find the control function as a function of time

$$\mathbf{x}^* = \mathbf{v}^*(t), \tag{8}$$

such that the system

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, \mathbf{g}\left(\mathbf{x}, \mathbf{v}^*(t)\right)\right)$$

will have a particular solution $\mathbf{x}\left(t, \mathbf{x}^0\right)$, which, from the given initial condition $\mathbf{x}^0$, will reach the specified final condition $\mathbf{x}\left(T, \mathbf{x}^0\right) = \mathbf{x}^f$, $T < t^+$, with the optimal value of the given quality criterion

$$J(\mathbf{v}^*(t)) = \int_0^T f_0\left(\mathbf{x}\left(t, \mathbf{x}^0\right), \mathbf{g}\left(\mathbf{v}^*(t), \mathbf{x}\left(t, \mathbf{x}^0\right)\right)\right) dt \to \min_{\mathbf{v}^* \in X_1}. \tag{9}$$

In the general case, the control function (8) can be some function of time $\mathbf{x}^* = \mathbf{v}^*(t)$. More often in the applied problems, the control function (8) is considered in the form of a piecewise-constant function of time by dividing the time interval $[0; t^+]$ into intervals Δt. Thus, in the optimal control problem, it is necessary to find the values of the coordinates of the stability point for each interval by determining the optimal value of the parameter vector $\mathbf{q}^*$ that minimizes the quality functional (9):

$$\mathbf{v}^*(t) = \mathbf{q}^{*,j},\ \text{if}\ t \in \left[(j-1)\Delta t; j\Delta t\right), j = 1, \dots, k, \tag{10}$$

where k is the number of intervals,

$$k = \left\lfloor \frac{t^+}{\Delta t} \right\rfloor + 1.$$

In this case, at the second stage, we solve the optimal control problem as a finite-dimensional optimization problem for the parameter vector $\mathbf{q}^* = \left[\mathbf{q}^{*,1}, \dots, \mathbf{q}^{*,k}\right]^T$.

Algorithmically, according to the principle, the solution of two problems is implemented sequentially:

1) the problem of synthesizing a stabilization system to ensure the stability of an object relative to a point in the state space,

2) and the problem of optimal control, which consists in parametric optimization of the points of the state space relative to which the control system synthesized at the first stage must ensure stability.

III. SPATIAL STABILIZATION SYSTEM

A. Control object model

As part of this study, we used the well-known Gazebo simulator. The choice was made in favor of this simulator due to its compatibility with ROS. Gazebo is a high-precision 3D physics simulator designed to simulate robotic systems. It allows you to simulate the dynamics of objects, sensors, and interaction with the environment, which allows you to test control algorithms without using real equipment. ROS (Robot Operating System) is a flexible platform for developing robotic applications. It provides tools for modular program construction, messaging between processes, hardware resource management, and integration of various management algorithms. The combination of ROS and Gazebo creates a powerful environment for the development, testing and debugging of autonomous robots. As part of this study, a ready-made model of the ROSbot 2.0 robot was used, integrated into the Gazebo environment and controlled via ROS. To determine the location of the robot, a plugin was used that provides accurate coordinates. A visual representation of the robot model in the simulator is shown in 1.

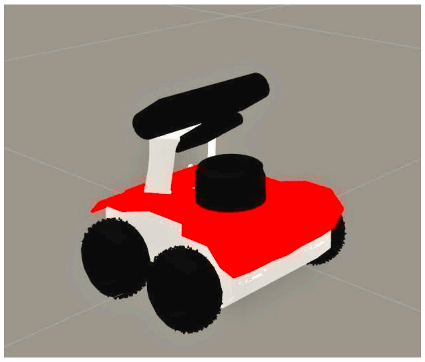

Fig. 1. The Rosbot 2.0 robot model in the Gazebo simulator

The robot is a platform supported by four wheels that cannot rotate around a vertical axis. An electric motor is attached to each of the wheels. The control is based on the principle of a differential circuit: forward and backward movement is achieved by applying the same voltage to all four electric motors. A right or left turn occurs due to an increase in the voltage on the electric motors of the wheels on the corresponding side.

The coordinate system used to define the robot model is shown in Figure 2. The state of the robot at each moment is determined by a set of numbers: $\{x, y, \theta, v, \omega\}$, where x, y are the coordinates of the robot on the plane, θ is the angle between the x-axis and the direction of the robot, v is the projection of velocity in the movement direction of the robot, and ω is the angular velocity. It is assumed that the vectors $\mathbf{v}$ and $\mathbf{h}$ are collinear.

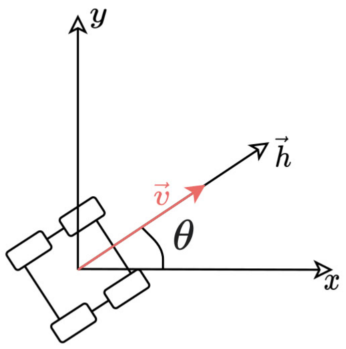

Fig. 2. 2. Robot coordinate system. The vector $\vec{h}$ indicates the direction of the robot when moving forward. The vector $\vec{v}$ represents the velocity of the robot.

The robot's motion model is described by the well-known system of differential equations:

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \omega \end{cases} \tag{11}$$

The physical control of the robot in Gazebo is implemented using two signals: v^d (the desired linear velocity) and ω^d (the desired angular velocity). In many works, a model is used to calculate optimal movements, according to which control signals are directly transmitted to the system as $v = v^d$, $\omega = \omega^d$. In this case, equations (1) are transformed into the following:

$$\begin{cases} \dot{x} = u^v\cos(\theta) \\ \dot{y} = u^v\sin(\theta) \\ \dot{\theta} = u^w \end{cases} \tag{12}$$

B. Stabilization system

The task of designing a stabilization system is a key objective in control theory. Typically, this task is addressed using a technical approach that employs a model of the control object and proportional-integral-derivative (PID) controllers, with parameters being manually adjusted. Additionally, methods based on linearizing the model, such as linear-quadratic regulator (LQR), can also be utilized. Therefore, the quality of the stabilization process largely depends on the regulator settings. These methods all require prior knowledge of the system in order to establish the controller structure optimally. However, even with this knowledge, there is no guarantee that the chosen structure is the best possible. Rather, the parameters are optimized according to a particular criterion. This paper presents a numerical machine learning approach based on symbolic regression for solving the problem of designing a stabilization system. This approach not only optimizes the parameters but also finds the optimal structure for the feedback control function.

IV. Time-Adaptive Particle Swarm Optimization for Robotic Path Planning

A. Problem Formulation

The algorithm addresses the multi-objective optimization problem of finding intermediate waypoints $\mathbf{W} = \{w_1, w_2, ..., w_n\}$ that minimize positional error, temporal constraints, and obstacle avoidance cost:

$$\min_{\mathbf{W}} \left(\alpha \cdot \|x_T - x_g\|^2 + \beta \cdot \mathcal{T} + \gamma \cdot \mathcal{C}_{obs} \right) \tag{13}$$

where:

- x_T: Final robot position
- x_g: Target position (x_g, y_g, θ_g)
- $\mathcal{T} = \max(0, T_{total} - T^+) + \frac{T_{used}}{T^+}$: Temporal penalty term
- T^+: Adaptive time budget (initially set to T_{max})
- $\mathcal{C}_{obs} = \sum_{i=1}^{n} \not{\mathbb{K}}(w_i \in \mathcal{O})$: Obstacle penalty, where
  - $\mathcal{O}$: Set of forbidden regions (obstacle zones)
  - $\not{\mathbb{K}}(w_i \in \mathcal{O})$ is an indicator function that equals 1 if waypoint w_i is inside an obstacle region, and 0 otherwise
- $\alpha = 10$, $\beta = 0.25$, $\gamma = 5$: Trade-off coefficients

The term $\mathcal{C}_{obs}$ penalizes waypoints that fall into restricted obstacle regions, ensuring that the trajectory avoids unsafe areas while optimizing for positional accuracy and time efficiency.

B. PSO Implementation

PSO is used to optimize the positioning of the intermediate waypoints. The particles in the swarm iteratively update their positions by combining their own best positions and the global best position found by the swarm. The goal is to minimize both the positional error and temporal penalties, with adaptive adjustments of the time budget and time step to improve the robot's efficiency in reaching the target. Each particle represents a potential solution with $3k$ dimensions for k waypoints:

$$P_i = [x_1, y_1, \theta_1, ..., x_k, y_k, \theta_k] \tag{14}$$

The velocity update rule incorporates cognitive and social components:

$$v_{id}^{t+1} = \omega v_{id}^t + c_1 r_1 (p_{id}^* - x_{id}^t) + c_2 r_2 (g_d^* - x_{id}^t) \tag{15}$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \tag{16}$$

where:

- $\omega = 1.0$: Inertia weight to maintain exploration
- $c_1 = c_2 = 0.65$: Acceleration coefficients
- p_{id}^*: Particle's historical best position
- g_d^*: Swarm's global best position

C. Time-Adaptive Cost Function

The fitness evaluation combines spatial, temporal, and obstacle-avoidance components:

$$\mathcal{F}(P_i) = \underbrace{\alpha \cdot \|x_{end} - x_g\|}_{\text{Positional Error}} + \underbrace{\beta \cdot \left(\mathcal{T}_{penalty} + \frac{T_{used}}{T^+} \right)}_{\text{Temporal Cost}} + \underbrace{\gamma \cdot \mathcal{C}_{obs}}_{\text{Obstacle cost}} \tag{17}$$

Key innovations in the cost calculation:

- Dynamic adaptation of the time budget T^+ based on actual performance
- Progressive tightening of time constraints for subsequent waypoints
- Adaptive adjustment of time steps during trajectory simulation, with new *time_step* and T_{max} calculations
- Obstacle cost term $\mathcal{C}_{obs}$ ensuring avoidance of restricted regions

D. Time-Adaptive Mechanism

The core innovation is the real-time adjustment of temporal constraints based on the robot's performance during simulation:

```
Algorithm 1 Adaptive Time Budget Update
Input: $T_{max}$, $T_{used}$, time_step
Output: Updated $T^+$, time_step
Initialize $T^+ \leftarrow T_{max}$   for each waypoint segment do
    Simulate movement with current time_step  if goal reached before $T_{max}$ then
        $T^+ \leftarrow T_{used}$  Update time_step
    end
end
```

The robot is guided to intermediate waypoints, and each segment of the trajectory is evaluated based on how much time was used to reach the target. If the robot reaches the goal faster than expected, the time budget T^+ and the *time_step* (frequency of switching between waypoints) are adjusted accordingly, as shown in Algorithm 1.

V. Performance Evaluation and Results

The primary objective of this experiment was to plan an optimal trajectory for a mobile robot navigating from an initial state $\mathbf{x}^0 = [0, 0, 0, 0]^T$ to a target state $\mathbf{x}^f = [x^f, y^f, \theta^f]^T$ in an environment with static circular obstacles. The problem was formulated as a multi-objective optimization task, where the goal was to minimize the quality functional (13). The control inputs were constrained by:

$$-0.5 \le u_i \le 0.5,\ i = 1, 2,$$

where u_1 and u_2 represent the linear and angular velocity controls, respectively.

The experiment was conducted in the Gazebo simulation environment using the ROSbot 2.0 model. The initial and target states were set as follows:

$$\mathbf{x}^0 = [0\ 0\ 0\ 0]^T, \quad \mathbf{x}^f = [x^f\ y^f\ \theta^f]^T.$$

Static circular obstacles were placed in the environment to test the algorithm's ability to avoid collisions while optimizing the trajectory. Initially, the maximum allowed time T_{max} for the robot to reach the target was set to 18 seconds, with a fixed time step of 3 seconds between intermediate waypoints. This setup provided a baseline for evaluating the algorithm's performance in terms of temporal efficiency and path quality. The choice of parameters $\alpha = 10$, $\beta = 0.25$, and $\gamma = 5$ was based on empirical testing. These values were found to provide a good balance between positional accuracy, temporal efficiency, and obstacle avoidance. However, the algorithm's performance may vary depending on the specific environment and task requirements, and further tuning may be necessary for different scenarios.

Using the proposed time-adaptive Particle Swarm Optimization (PSO) algorithm, the system dynamically adjusted the time budget and time steps during the simulation. The algorithm optimized the trajectory by reducing the total time to reach the target while maintaining smooth and collision-free motion. As a result, the total time was optimized from the initial 18 seconds to 4.5 seconds, and the time step between waypoints was reduced accordingly. This optimization was achieved by iteratively adjusting the time budget T^+ and the time step based on the robot's performance during the simulation.

The algorithm calculated the following set of intermediate waypoints to guide the robot from the initial state to the target state: $(-0.46, -0.3)$, $(-0.9, 0.24)$, $(-0.37, 0.4)$, $(-0.3, 0.64)$, $(0, 1.7)$ $(2, 2)$.

Figure 3 shows the robot's trajectory and velocity profiles in an environment with static circular obstacles. The top plot illustrates the 2D path of the robot, where the optimized intermediate waypoints (blue circles) guide the robot around the obstacles (black circles) toward the target position (red star). The bottom plot displays the linear velocity v (blue line) and angular velocity ω (orange line) over time. The velocities are dynamically adjusted to ensure smooth and collision-free motion, demonstrating the algorithm's ability to handle complex environments. The robot successfully navigated from the initial position to the target state by following the calculated intermediate waypoints. The trajectory was smooth and avoided all obstacles, demonstrating the effectiveness of the proposed approach. In a scenario where no obstacles are present in the environment, the algorithm demonstrates its ability to optimize both spatial and temporal efficiency. Specifically, all intermediate waypoints $\mathbf{W} = \{w_1, w_2, \ldots, w_n\}$ are optimized to converge to the initial state $\mathbf{x}^0 = [0, 0, 0, 0]^T$. This behavior effectively collapses the trajectory into a direct path from the initial position to the target, as the intermediate points no longer play a role in guiding the robot. This can be expressed as:

$$w_i \to \mathbf{x}^0 \quad \forall i \in \{1, 2, \ldots, n\},$$

where w_i represents the i-th intermediate waypoint. This optimization ensures that the robot follows the most efficient path in terms of both space and time.

Additionally, the algorithm significantly reduces the total time required to reach the target. Initially, the maximum allowed time T_{max} was set to 15 seconds. However, after optimization, the total time is compressed to just 3 seconds. This reduction in time is achieved by dynamically adjusting the time budget T^+ and eliminating unnecessary delays between waypoints.

An example of this behavior is illustrated in Figure 4, where the trajectory is shown to collapse into a direct path, and the time intervals are minimized. This demonstrates the algorithm's ability to handle simplified environments and optimize performance when obstacles are absent.

Figure 4 presents the results for an obstacle-free environment. The top plot shows the robot's 2D path, where the trajectory collapses into a direct path to the target (red star). The bottom plot shows the velocity profiles, where the linear and angular velocities are

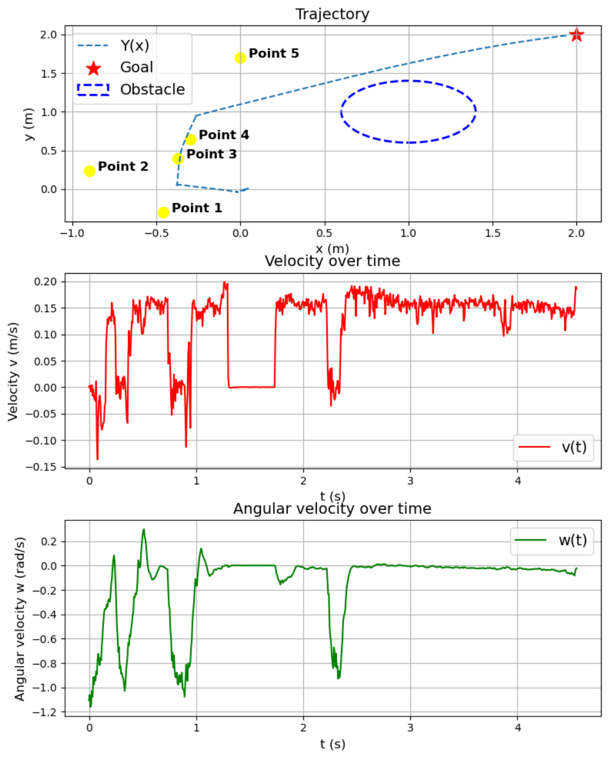

Fig. 3. Trajectory and velocity profiles of the robot in an environment with static obstacles.

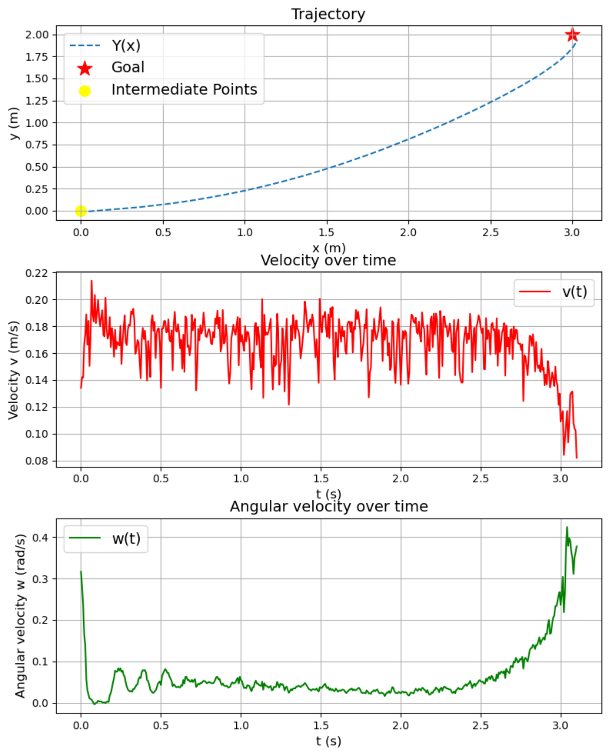

Fig. 4. Trajectory and velocity profiles in an obstacle-free environment.

optimized to minimize travel time.

VI. CONCLUSIONS

In this paper, we present a novel approach to mobile robot motion planning that combines synthesized optimal control with time-adaptive Particle Swarm Optimization (PSO). Our algorithm effectively addresses the challenges of navigating complex environments with static obstacles, while optimizing both spatial and temporal constraints. By dynamically adjusting the time budget and intermediate waypoints, our algorithm generates smooth, collision-free trajectories that minimize a quality functional that includes positional error, time to reach the target, and obstacle avoidance costs. This allows the robot to navigate complex environments efficiently and safely. The experimental results demonstrate the flexibility and efficiency of the algorithm. In scenarios with obstacles, the algorithm successfully guides the robot through a series of optimized intermediate waypoints, avoiding collisions and minimizing travel time. Without obstacles, the trajectory is collapsed into a direct path, ensuring that all intermediate points return to the initial state. This adaptability makes the approach suitable for various real-world applications, including industrial automation and autonomous vehicles. Future work will focus on extending the algorithm to deal with dynamic obstacles and more complex environments. Additionally, computational efficiency will be optimized for real-time applications. These findings provide a solid basis for advancing robotic motion planning and control research.

References

[1] J.-P. Laumond, Kineo CAM: A success story of motion planning algorithms. IEEE Robot. Autom. Mag., vol. 13, No 2, 2006, pp. 90–93.

[2] Sucan I., Moll M., Kavraki L.E. The Open Motion Planning Library. IEEE Robot. Autom. Mag. vol. 19, No 4, 2012, pp. 72–82.

[3] Porta J.M., Ros L., Bohigas O., Manubens M., Rosales C., Jaillet L. The CUIK Suite: Motion Analysis of Closed-chin Multibody Systems. IEEE Robot. Autom. Mag., vol. 21, No 3, 2014, pp. 105–114.

[4] E.R. Bryson, Ho Yu-Chi, Applied Optimal Control. Taylor & Francis, London, 1969.

[5] Karamzin, D., Pereira, F.L. On a Few Questions Regarding the Study of State-Constrained Problems in Optimal Control. J Optim Theory Appl, 2019, 180, 235–255.

[6] D.P. Bertsekas, Dynamic Programming and Optimal Control, Vol. I, 4th Edition, Athena Scientific, Belmont, Massachusetts, 2012.

[7] T.J. Böhme, B. Frank, Direct Methods for Optimal Control. In: Hybrid Systems, Optimal Control and Hybrid Vehicles. Advances in Industrial Control. Springer, Cham, 2017, pp. 233–273.

[8] A. Diveev, E. Shmalko, Comparison of Direct and Indirect Approaches for Numerical Solution of the Optimal Control Problem by Evolutionary Methods. In: Jaćimović, M., Khachay, M., Malkova, V., Posypkin, M. (eds) Optimization and Applications. OPTIMA 2019. Communications in Computer and Information Science, vol 1145. Springer, Cham, 2020, pp. 180–193.

[9] K.A. Kazakov, V.A. Semenov, An overview of modern methods for motion planning. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 4, 2016, pp. 241-294.

[10] R.A. Brooks, T. Lozano-Peres, A Subdivision Algorithm in Configuration Space For Find Path with Rotation. IEEE Trans. Syst. Man. Cybern., vol. SMC-15, № 2, 1985, pp. 224–233.

[11] B. Chazelle, Approximation and Decomposition of Shapes. Advances in Robotics 1: Algorithmic and Geometric Aspects of Robotics, 1987, pp. 145–185.

[12] H. Choset, K. Lynch, H. Seth, G. Kantor, W. Burgard, L.E. Kavraki, S. Thrun, Principles of Robot Motion-Theory, Algorithms , and Implementation. MIT Press, 2005.

[13] F. Aurenhammer, Voronoi Diagrams – A Survey of a Fundamental Data Structure. ACM Comput. Surv., vol. 23, No 3, 1991, pp. 345–405.

[14] Egerstedt, M. Motion Planning and Control of Mobile Robots, Doctoral Thesis, Stockholm, 2000.

[15] E. Shmalko, Computational Approach to Optimal Control in Applied Robotics. In: Ronzhin, A., Pshikhopov, V. (eds) Frontiers in Robotics and Electromechanics. Smart Innovation, Systems and Technologies, vol 329. Springer, Singapore, 2023, pp. 387–401.

[16] A. Diveev, E. Shmalko, V. Serebrenny, P. Zentay, Fundamentals of Synthesized Optimal Control. Mathematics 2021, 9, 21.