Implementation of Rapidly-Exploring Random Tree (RRT) for Path Planning and Stanley Control Method for Path Tracking on Wheeled Soccer Robots

Hendawan Soebhakti[1], Budiana[2], Fahri Rahmat[3], Masdika Aliman[4], and Nyoman Krisna Prebawa[5]

{hendawan@polibatam.ac.id[1], budiana@polibatam.ac.id[3], fahri.rahmat46@gmail.com[3], masdikaaliman@gmail.com[4] , and nyomankrisnaprebawa@gmail.com[5]}

Barelang Robotics Artificial and Intellegence Lab (BRAIL), Department of Electrical Engineering, Politeknik Negeri Batam, Batam, Indonesia[1,2,3,4,5]

Abstract. A wheeled soccer robot, designed for autonomous play, faces challenges in motion planning due to dynamic player interactions. This involves generating optimal paths for goal-oriented movements, utilizing techniques like Rapidly-exploring Random Trees (RRTs) for path planning and Stanley control for path tracking. Results at 80 cm/s show a Root Mean Square Error (RMSE) of 14.02 cm (x-axis) and 12.9 cm (y-axis). In the third test, RMSE is 6.04 cm (x-axis) and 14.16 cm (y-axis), with a 7.12-second travel time. The motion planning system, employing RRTs and Stanley Control, produces collision-free trajectories, tracked effectively in real-time. Obstacle positioning impacts travel time but doesn't impede trajectory selection. The system adeptly generates and tracks optimal paths for wheeled soccer robots.

Keywords: RRT, Stanley Control, Robot Soccer

1 Introduction

RoboCup is an international robot competition dedicated to developing AI and intelligent robotics research [1]. Robocup middle-size league is a competitive sport where teams of robots play soccer against each other. Robots are usually autonomous, meaning they can make their own decisions and execute their own actions without human intervention and have a vision that is 2050 against humans [2]. Most MSL teams adopt omniwheels-based drive configurations for the sake of agility [3-6]. One of the key challenges in robot soccer is motion planning, which refers to the process of determining the optimal path or trajectory that a robot should follow to achieve a specific goal as well as path tracking to control the movement of the robot along the predetermined path[7]. This can be a complicated task, especially in the dynamic and unpredictable environment of a soccer game, where the positions and movements of other robots and the ball are constantly changing[8].

ICAE 2023, November 07, Batam, Indonesia
Copyright © 2024 EAI
DOI 10.4108/eai.7-11-2023.2342945

To address this challenge, researchers and teams participating in RoboCup have developed various motion planning algorithms that allow robots to navigate the field and interact with teammates and opponents in real-time. These algorithms take into account various factors such as the current position of the robot, the position of the ball, the position of other robots, and the rules of the game, to determine the best action for the robot to take[9-10].
Path planning is the process of determining the optimal path that a robot should follow to reach a specific destination while avoiding obstacles and meeting other constraints. This process usually involves the use of algorithms that consider the robot's current location, the location of the goal, and any form of obstacles in the environment[11-13]. Therefore, the goals of this research is to create a pathplanning system for robots by implementing Rapidly-Exploring Random Tree (RRT) and Stanley control methods to help robots navigate the field, avoid other robots and obstacles, and position themselves to score goals.
RRT (Rapidly-Exploring Random Tree) is a type of motion planning algorithm commonly used in robotics and other fields. It works by building a tree-like structure in the robot workspace, where each node in the tree represents a possible robot configuration. The algorithm starts with a randomly generated initial configuration and iteratively adds new configurations to the tree until it reaches the desired goal configuration. The tree is constructed in a way that allows the algorithm to efficiently explore the workspace and find a path to the goal that avoids obstacles[14-15].
Once the RRT algorithm generates a feasible path, the robot can use control algorithms such as Stanley control to follow it. Stanley control is a feedback-based control method that allows the robot to track the desired path by comparing its actual position and heading to the desired path and generating control inputs to correct any deviations. This can be done by calculating the error between the robot's current position and orientation and the desired position and orientation on the path, and then using the feedback control loop to generate control inputs to move the robot towards the desired state[16].
By combining RRT for path planning and Stanley control for path tracking, an all-directional three-wheeled robot can navigate through complex environments and follow the desired path well.

2 Method

2.1 Inverse Kinematics

Inverse kinematics refers to the robot's kinematic equations to determine the joint parameters that result in the desired final position. The kinematic equations define several parts of the robot's motion, including the velocities Vx, Vy and V_x, V_y and Omega to be the robot's angular velocity equation. To get the speed of each wheel must set the position of the omniwheels, namely the position $wheel_1 = 30$ °, $wheel_2 = 150$ °, $wheel_3 = 270$ °.

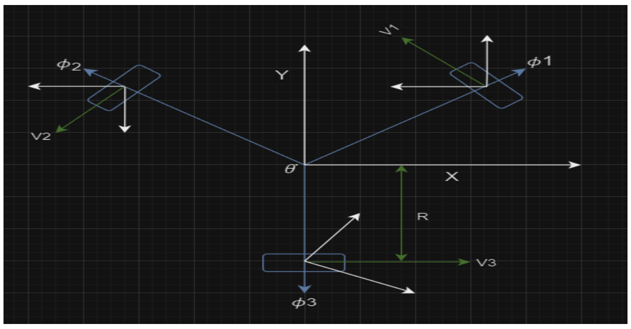


Fig. 1 A diagram inverse kinematics

$$\begin{bmatrix}\phi_1\\ \phi_2\\ \phi_3\end{bmatrix} = \frac{1}{r}\begin{bmatrix}-\sin(\alpha_1) & \cos(\alpha_1) & R\\ -\sin(\alpha_2) & \cos(\alpha_2) & R\\ -\sin(\alpha_3) & \cos(\alpha_3) & R\end{bmatrix}\begin{bmatrix}Vx\\ Vy\\ \theta\end{bmatrix} \qquad (1)$$

From the inverse kinematic equation can be known where $\dot{\phi}$ is the speed of each wheel in rad/s, vx is the speed towards the x axis, vy is the speed towards the y axis with units of cm/s, $\dot{\theta}$ is the angular velocity of the robot with units of rad/s, R is a parameter resulting from the result of the distance between the wheel and the center of the robot in centimeters, r is the radius of the omniwheel wheel used.

2.2 Odometry

Forward Kinematics: Forward Kinematics is a matrix formula that determines the position and direction of motion of the robot. was built with a configuration similar to that shown in Figure 4, which shows the symmetrical mounting positions of the three omniwheels. The symmetrically mounted omni wheel position means that the center of gravity is at the center of the robot, position a1 = 180°, a2 = 300°, a3 = 60°.

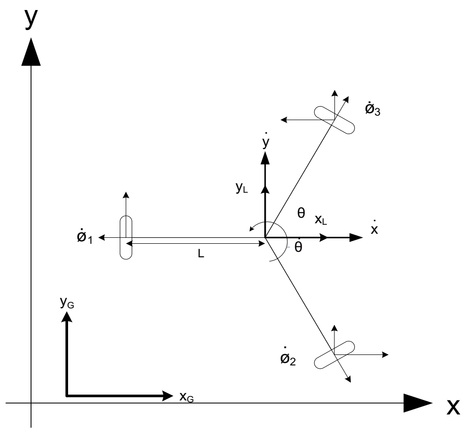


Fig. 2 Forward Kinematic Diagram

To get the desired movement, it can be calculated using the Jacobian matrix equation by calculating the previous motion:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\psi}\end{bmatrix} = R \begin{bmatrix} -\frac{2\,\mathrm{Sin}[\psi]}{3} & -\frac{\mathrm{Cos}[\psi]}{\sqrt{3}}+\frac{\mathrm{Sin}[\psi]}{3} & \frac{\mathrm{Cos}[\psi]}{\sqrt{3}}+\frac{\mathrm{Sin}[\psi]}{3} \\ -\frac{2\,\mathrm{Cos}[\psi]}{3} & \frac{\mathrm{Cos}[\psi]}{3}+\frac{\mathrm{Sin}[\psi]}{\sqrt{3}} & \frac{\mathrm{Cos}[\psi]}{3}-\frac{\mathrm{Sin}[\psi]}{\sqrt{3}} \\ \frac{I}{3\,L} & \frac{I}{3\,L} & \frac{I}{3\,L} \end{bmatrix} \begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{bmatrix} \qquad (2)$$

Where $\dot{x}$ is the velocity in the X axis in cm/s, $\dot{x}$ is the velocity in the Y axis in cm/s, $\dot{\psi}$ is the angular velocity of the robot rad/s, R is the radius of the omni wheel in cm, L is the distance from the center of the robot to the center of the wheel in cm. $\dot{\psi}$ is the rotation speed of the wheel in rad/s.

2.3 Path Planning with Rapidly-Exploring Random Tree (RRT)

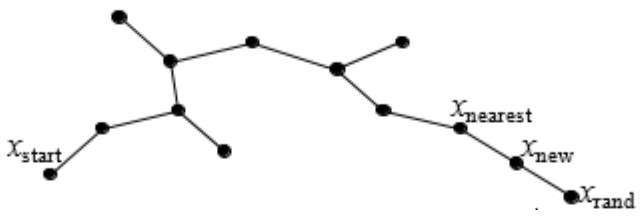


Fig. 3 Rapidly-exploring Random Tree

The RRT algorithm involves RandomSample, NearestNeighbor, Steer, and InsertNode processes. RandomSample selects samples in the field (Xrand). NearestNeighbor finds the closest node (Xnearest) in the search tree to Xrand. A new node (Xnew) is created between Xnearest and Xrand, positioned ΔX away from Xnearest. If no obstacles are in the path, Xnew is added to the tree, and the iteration repeats n times.

RRT Pseudo Code

```
Qgoal //region that identifies success
Counter = 0 //keeps track of iterations
lim = n //number of iterations algorithm should run for
G(V,E) //Graph containing edges and vertices, initialized as empty
While counter < lim:
    Xnew  = RandomPosition()
    if IsInObstacle(Xnew) == True:
        continue
    Xnearest = Nearest(G(V,E),Xnew) //find nearest vertex
    Link = Chain(Xnew,Xnearest)
    G.append(Link)
    if Xnew in Qgoal:
        Return G
Return G
```

2.4 Path Tracking with Stanley Control

Stanley Control is a method of tracking the trajectory of an autonomous robot in real-time by adjusting the robot's current position and facing direction against a position and orientation reference. This Stanley controller method is used on 3-wheeled omnidirectional robots where the position of the axle is at the center point of the robot to correct the distance to the robot In general, stanley controllers use the front axle position, in addition to correcting yaw differences, stanley also corrects cross trajectories from the distance between the front axle and the nearest point on the trajectory.

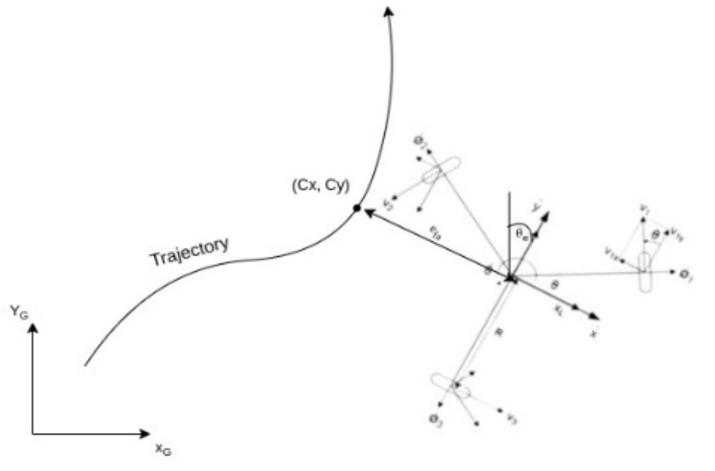


Fig. 4 Stanley method geometry

Where θ is the robot's current facing direction and θ_p is the trajectory direction at (cx, cy). When e_{fa} is non-zero, the robot adjusts and intersects the trajectory tangent of (cx, cy) at unit kv(t). Figure 2 illustrates the geometry relationship of the control parameters. The resulting steering control law is given as :

$$\delta(t) = \theta_e(t) + tan^{-1}\left(\frac{ke_{fa}(t)}{v_x(t)}\right) \qquad (3)$$

Where, k is the gain parameter so that the desired can be achieved with this formula. As e_{fa} increases the wheels will try to head towards the trajectory. Cross track error can be calculated as the distance of the closest point (reference point) to the center point of the robot by finding the distance of all points along the current track segment from the front axle and taking the minimum set of distances.

2.5 Hardware Design

Figure 7 is a block diagram of the robot hardware. In this research, the robot uses 2 cameras as sensors, mini PC as the main device, STM32F4 has ARM Cortex-M4 architecture, and Arduino Mega 2560 is used as a special sub-controller for further control such as sensors, drible control and kicker control on the robot, proximity sensor is used to detect ball possession, buttons are used as input to the robot, gyroscopic sensor is used to determine the direction of the robot, and finally there is rotation sensor. used to find out how far the robot is moving. Figure 2 shows the block diagram of the currently used KRSBI (BARELANG 63) wheeled robot system as a whole.

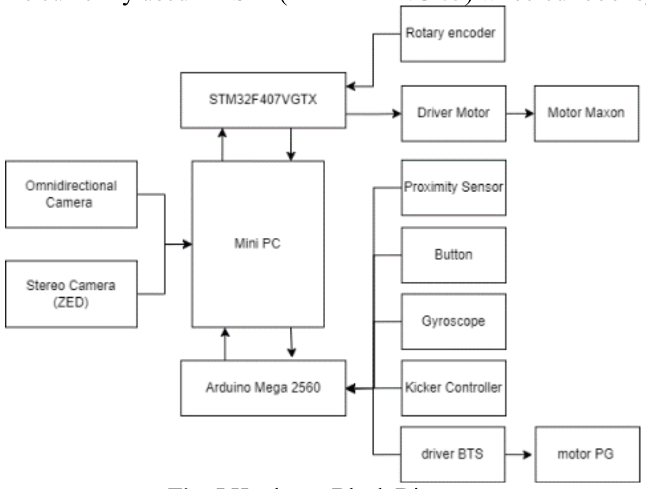


Fig. 5 Hardware Block Diagram

2.6 Software Design

To support and implement the method used in the robot for the purposes of the movement planning system, a software design is needed with a motion planning system diagram according to Figure 6 as follows:

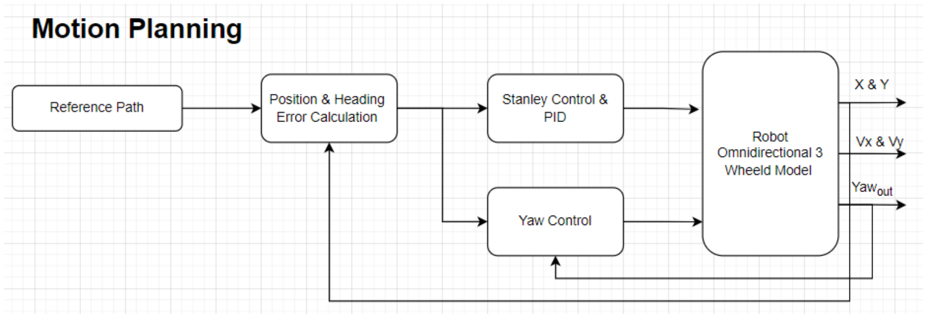


Fig. 6 Block diagram of motion planning

The reference path is the target position x and y. The block calculates the longitudinal lateral error and orientation error with the input of the robot position and the desired position goal. Furthermore, the calculation results in the form of lateral and orientation errors into the Stanley controller to get the closest destination coordinates according to the path and PID control to get the speed of the x and y axis directions. yaw control block calculates the moment value with the desired yaw input with the yaw generated by the robot.

3 Result

This chapter covers system testing and results, focusing on gyro sensor testing and evaluating path planning and tracking for a 3-wheeled soccer robot. Various scenarios involving robot and obstacle positions are used to assess algorithm performance. The evaluation utilizes the Root Mean Squared Error (RMSE) formula, a unitless measure, to determine the system's accuracy by comparing predicted and observed data and calculating the average of squared errors.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(A_t - F_t)^2}{n}} \qquad (4)$$

Where, RMSE is the root mean square error value, A is the observed value, F is the predicted value, t is the data sequence value in the database and n is the number of data.

The algorithm parameter values are shown in Table. 2 and Table 3 as follows.

Table 1 Parameter RRTs

Random Sampling Area [min, max] (cm)	**Goal Sample Rate (Hz)**	**Max Iteration**	**Robot Radius (cm)**
[0,1200] ,[0,800]	25	10.000	50

Based on table 1 random sampling area defines the boundary of the random sampling area to build a tree, this area adjusts the size of a soccer field. Goal sample rate is the frequency rate or sampling rate of about 25 times per second the algorithm tries to connect the random sample point to the goal point. Max iteration is the maximum iteration parameter that determines the number of iterations the algorithm will perform before terminating. A higher value of the iteration parameter allows the algorithm to explore the search space more widely depending on the size of the sampling area or the size of the field.

Table 2 Parameter Stanley Control

k	k_p	**L (cm)**	**Rate (Hz)**
0.5	1.2	25	50

Based on table 2 where, k is a gain control parameter determining how strongly the control system responds to errors between the trajectory or reference path and the actual position of the robot and also the heading error on the robot. k_p is the proportional control for speed. L is the distance between the robot center point (robot position) and the nearest coordinate point (reference path). Rate is the frequency speed of about 50 times per second Stanley control algorithm. This parameter is obtained based on try and error done on the striker robot.

3.1 Path Tracking Testing

This test was conducted with a scenario without obstacles. There are 5 destination points that the system must reach. Figure 3.1 is the initial position of the scenario. This scenario test is carried out to determine the performance of the system in performing translational movements. It can be observed that the red line is the path that must be traveled to the target point. The robot will always follow the path to the target point.

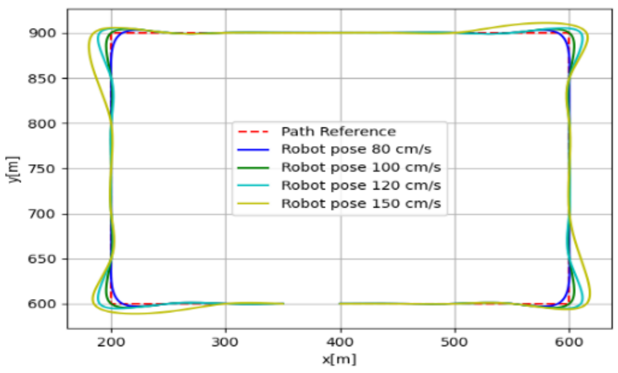


Fig. 7 Robot path tracking response to trajectory

Figure 7 depicts path tracking comparison across various robot speeds. Figure 10 demonstrates effective path tracking at a speed of 80 cm/s. However, as the robot speed increases, the lateral error generated by the controller also increases. Sharp turns result in larger lateral errors during tracking.

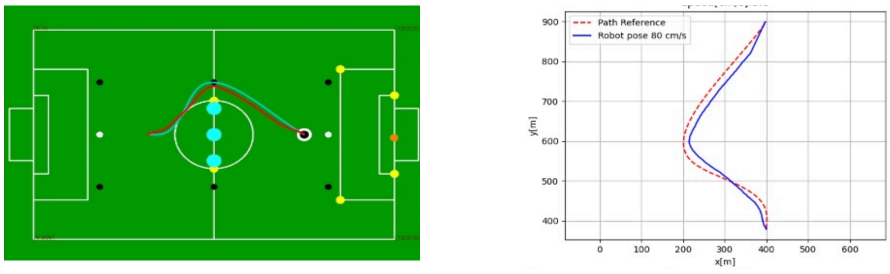
Table 3 The performance of response algorithm on the system

Speed robot cm/s	**Time (s)**	**RMSE X (cm)**	**RMSE Y (cm)**
80	17.2	14.02	12.9
100	14.5	14.80	13.92
120	12.7	15.68	14.79
150	10.6	20.21	18.21

Table 4 shows four tests with different speeds of 80 cm/s, 100 cm/s, 120 cm/s, and 150 cm/s. RMSE is the root mean square error value of the Stanley controller for the vehicle speed variation. The speed of 150 cm/s has the largest tracking error of 20.21 cm in the x-axis and 18.21 in the y-axis. This shows that the tracking path using a speed of 150 cm/s robot has not given good results. In this study, Stanley's control gives the smallest error value when the speed of 80 cm/s is 14.02 cm on the x axis and 12.9 cm on the y axis.

3.2 Obstacle avoiding Testing

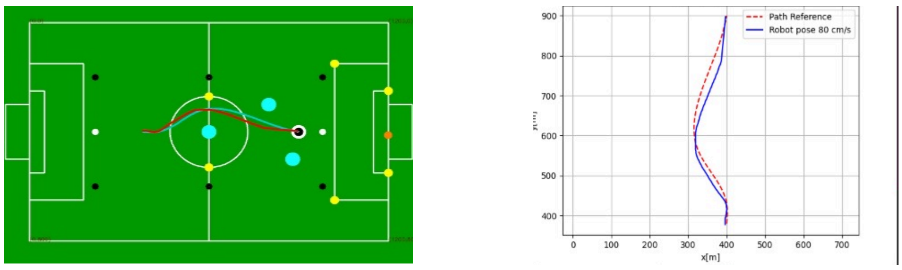
Obstacle avoidance testing uses the Rapidly-exploring Random Tree algorithm to create its own path with the assumption that obstacles are either opponent robots or friend robots. This test is carried out by providing interference in the direction of robot movement. The robot will detect the position of the opponent robot and create its own path to avoid it. Then the robot moves following the path that has been formed so that the robot can pass the obstacle and go to the destination point without hitting the obstacle.



(a) (b)

Fig. 8 First experiment robot avoided the obstacles

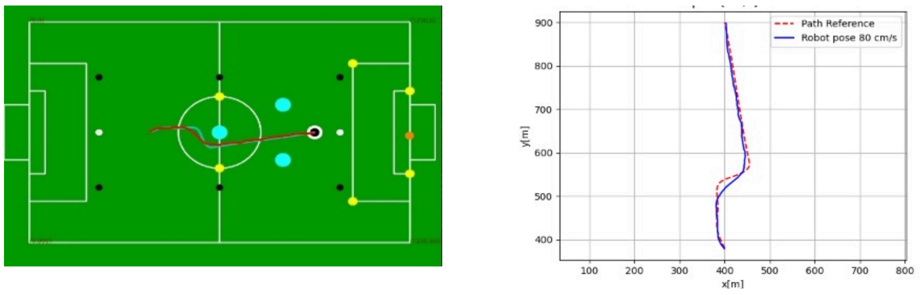
Figure 8. Shows that in the first experiment the robot avoided the obstacle (a) shows the visual localization in the field, (b) shows the reference path results and the robot trajectory results when avoiding obstacles.



(a) (b)

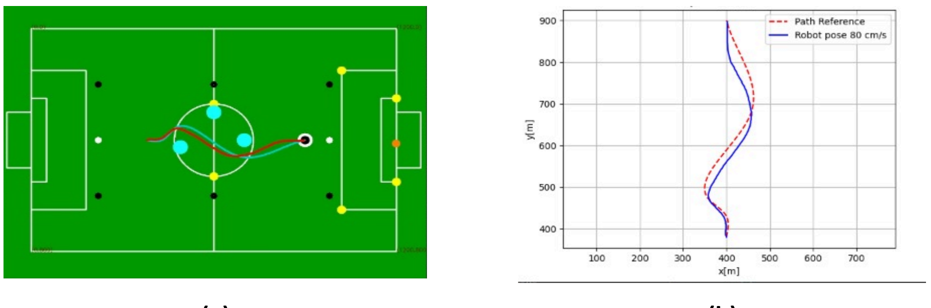
Fig. 9 Shows that in the second experiment the robot avoided the obstacle.

Figure 9. Shows that in the second experiment the robot avoided the obstacle.



(a) (b)

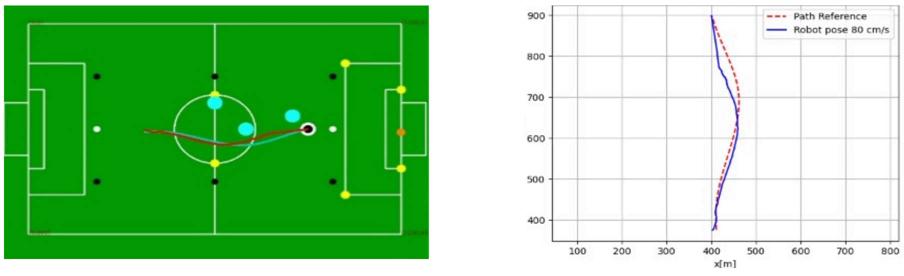
Fig. 10 Third experiment robot avoided the obstacles



(a) **(b)**

Fig. 11 Fourth experiment robot avoided the obstacles

Figure 11. Shows that in the fourth experiment the robot avoids obstacles.



(a) (b)

Fig. 12 Fifth experiment robot avoided the obstacles

Figure 12 exhibits the fifth experiment, highlighting successful obstacle avoidance. Part (a) illustrates visual localization on the field, while part (b) showcases the reference path results and the robot's trajectory during obstacle avoidance.
Figures 8-12 provide graphical insights into algorithm testing on the robot system, enabling a visual comparison between the reference path trajectory and the recorded robot trajectory movement. Figure 11 part (b) offers a comparison graph between the RRT-generated reference path and the actual robot trajectory. The red graph represents the reference path, and the blue graph illustrates the robot's trajectory movement. The RMS error values derived from the recorded data of the 5 experiments are summarized in the obstacle avoidance performance table.

Table 4 The performance of Obstacle Avoiding

Starting Point (x,y)	Endpoint (x,y)	Obstacle Position (x,y)	Time (s)	RMSE X (cm)	RMSE Y (cm)
(400,380)	(400,900)	(400,600) (300,600) (500,600)	9.7	11.38	14.7
(400,380)	(400,900)	(300,800) (400,600) (500,880)	8.7	7.56	17.1
(400,380)	(400,900)	(300,800) (400,600) (800,300)	7.12	6.04	14.16
(400,380)	(400,900)	(425,490) (300,600) (350,850)	9.1	8.24	16.7
(400,380)	(400,900)	(300,600) (400,700) (350,850)	8.5	6.34	17.92

Table 5 summarizes 5 tests with varying obstacle positions to optimize the robot's path to the destination. RMSE compares the RRTs-generated reference trajectory with the robot's trajectory while following the path. The first experiment exhibited the largest tracking errors: 11.38 cm on the x-axis, 14.7 cm on the y-axis, and a travel time of 9.7 seconds. Despite the longer travel time, this indicates the robot prioritizes finding an obstacle-free or low-risk collision path. Conversely, the third experiment demonstrated lower RMSE: 6.04 cm on the x-axis, 14.16 cm on the y-axis, and a shorter travel time of 7.12 seconds. This difference is attributed to obstacle positions that minimally affect the selection of robot trajectories.

4 Conclusion

This paper outlines a path planning and tracking controller designed for a wheeled soccer robot. The path planning employs the Rapidly-Exploring Random Tree (RRT) method, focusing on generating a feasible trajectory by sampling the configuration space and incrementally exploring it until reaching the destination. RRT has demonstrated effectiveness in creating collision-free paths in uncertain environments. However, challenges arise in real-time adaptation and high computational costs due to the extensive field space. Future work aims to enhance computational efficiency by developing adaptive sampling techniques and algorithms accommodating dynamic conditions.
The path tracking controller, utilizing Stanley control based on yaw rate and a kinematic model for 3-wheel omniwheels, proves effective at speeds up to 80 cm/s, demonstrating precise path following. However, at speeds exceeding 80 cm/s, an increase in mean square error is observed. This is attributed to the control parameter's optimal fit at 80 cm/s. Future research suggests

integrating this control with an adaptive approach to handle speed variations for achieving the desired trajectory effectively.

References

[1] T. Balch, T. Schmitt, F. Schreiber, and B. Cunha, "Middle Size Robot League Rules and Regulations for 2009," 2009.

[2] M. A. Ismail, D. Purwanto and A. Arifin, "Soccer Robot Localization Based on Sensor Fusion From Odometry and Omnivision," 2022 International Seminar on Intelligent Technology and Its Applications (ISITIA), 2022, pp. 273-278, doi: 10.1109/ISITIA56226.2022.9855313.

[3] A. T. A. Peijnenburg, G. P. Veldhuis and T. P. H. Warmerdam, "Philips CFT RoboCup platform selection," Proceedings of the IEEE Internatinal Symposium on Intelligent Control, Vancouver, BC, Canada, 2002, pp. 811-814, doi: 10.1109/ISIC.2002.1157866.

[4] Lunenburg J, Soetens R, Schoenmakers F, Metsemakers P, Molengraft, van de M, Steinbuch M. Sharing open hardware through ROP, the Robotic Open Platform. In: Proceedings of RoboCup 2013, symposium papers and team description papers, 26-30 June 2013, Eindhoven, The Netherlands. 2013. p. 584–91. http://dx.doi.org/10.1007/978-3-662-44468-9_53.

[5] K. V. Utama, R. A. Fatekha, S. Prayoga, D. S. Pamungkas, and R. P. Hudhajanto, "Positioning and Maneuver of an Omnidirectional Robot Soccer," *Proc. 2018 Int. Conf. Appl. Eng. ICAE 2018*, pp. 1–5, 2018, doi: 10.1109/INCAE.2018.8579148.

[6] Hacene, Nacer & Mendil, Boubekeur. (2019). Motion Analysis and Control of Three-Wheeled Omnidirectional Mobile Robot. Journal of Control, Automation and Electrical Systems. 30. 10.1007/s40313-019-00439-0.

[7] Azizi, M. R., Rastegarpanah, A., & Stolkin, R. (2021). Motion planning and control of an omnidirectional mobile robot in dynamic environments.*Robotics*,*10*(1).https://doi.org/10.3390/robotics10010048J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of TCP Reno congestion avoidance and control," Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.

[8] R. Dikairono, A. A. Rachman, Setiawardhana, T. A. Sardjono and D. Purwanto, "Motion planning simulator for holonomic robot soccer platform," 2017 International Seminar on Intelligent Technology and Its Applications (ISITIA), Surabaya, Indonesia, 2017, pp. 368-371, doi: 10.1109/ISITIA.2017.8124111.

[9] Ajeil, F. H., Ibraheem, I. K., Azar, A. T., & Humaidi, A. J. (2020). Autonomous navigation and obstacle avoidance of an omnidirectional mobile robot using swarm optimization and sensors deployment. *International Journal of Advanced Robotic Systems*, *17*(3). https://doi.org/10.1177/1729881420929498

[10] A. Šelek and M. Seder, "Smooth Motion Planning of an Omnidirectional Mobile Robot in Dynamic Environments," 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Maldives, Maldives, 2022, pp. 1-6, doi: 10.1109/ICECCME55909.2022.9988732.

[11] Erke, Shang & Bin, Dai & Yiming, Nie & Qi, Zhu & Liang, Xiao & Dawei, Zhao. (2020). An improved A-Star based path planning algorithm for autonomous land vehicles. International Journal of Advanced Robotic Systems. 17. 172988142096226. 10.1177/1729881420962263.

[12] M. Gavani, D. Tanpure and P. Falake, "Path Planning of Three Wheeled Omni-Directional Robot Using Bezier Curve Tracing Technique and PID control Algorithm," 2019 IEEE Pune Section International Conference (PuneCon), Pune, India, 2019, pp. 1-6, doi: 10.1109/PuneCon46936.2019.9105899.

[13] Khanal, Abhish. (2022). RRT and RRT* Using Vehicle Dynamics. 10.48550/arXiv.2206.10533.

[14] Zhao, Penglei & Chang, Yinghui & Wu, Weikang & Luo, Hongyin & Zhou, Zhixin & Qiao, Yanping & Li, Ying & Zhao, Chenhui & Huang, Zenan & Liu, Bijing & Liu, Xiaojie & He, Shan & Guo, Donghui. (2023). Dynamic RRT: Fast Feasible Path Planning in Randomly Distributed Obstacle Environments. Journal of Intelligent & Robotic Systems. 107. 10.1007/s10846-023-01823-4.

[15] Li, Jianyu & Wang, Kezhi & Chen, Zonghai & Wang, Jikai. (2022). An Improved RRT* Path Planning Algorithm in Dynamic Environment. 301-313. 10.1007/978-981-19-9195-0_25.

[16] Wang, Liang & Zhai, Zizhuo & Zhu, Zongshun & Mao, Enrong. (2022). Path Tracking Control of an Autonomous Tractor Using Improved Stanley Controller Optimized with Multiple-Population Genetic Algorithm. Actuators. 11. 22. 10.3390/act11010022.