Engineering 4 (2018) 491–499

Contents lists available at ScienceDirect

Engineering

journal homepage: www.elsevier.com/locate/eng

Research
Robotics—Article

Control of Velocity-Constrained Stepper Motor-Driven Hilare Robot for Waypoint Navigation

Robins Mathew, Somashekhar S. Hiremath *

Department of Mechanical Engineering, Indian Institute of Technology Madras, Chennai 600036, India

ARTICLE INFO

Article history:
Received 26 June 2017
Revised 11 February 2018
Accepted 12 July 2018
Available online 19 July 2018

Keywords:
Trajectory tracking
Adaptive control
Waypoint navigation
Hilare robot
Particle swarm optimization
Probabilistic road map

ABSTRACT

Finding an optimal trajectory from an initial point to a final point through closely packed obstacles, and controlling a Hilare robot through this trajectory, are challenging tasks. To serve this purpose, path planners and trajectory-tracking controllers are usually included in a control loop. This paper highlights the implementation of a trajectory-tracking controller on a stepper motor-driven Hilare robot, with a trajectory that is described as a set of waypoints. The controller was designed to handle discrete waypoints with directional discontinuity and to consider different constraints on the actuator velocity. The control parameters were tuned with the help of multi-objective particle swarm optimization to minimize the average cross-track error and average linear velocity error of the mobile robot when tracking a predefined trajectory. Experiments were conducted to control the mobile robot from a start position to a destination position along a trajectory described by the waypoints. Experimental results for tracking the trajectory generated by a path planner and the trajectory specified by a user are also demonstrated. Experiments conducted on the mobile robot validate the effectiveness of the proposed strategy for tracking different types of trajectories.

1. Introduction

The application of mobile robots, and particularly wheeled mobile robots (WMRs), is exponentially increasing in today's fast-growing market. These robots play a major role in many sectors, with their applications ranging from service robots to military robots. The Hilare robot is one type of robot among WMRs with a number of applications, primarily in indoor environments. Fig. 1 provides a schematic diagram of a Hilare robot. This type of mobile robot has two independently actuated wheels mounted on either side of the body in such a way that the central axis of wheels coincides with each other, and has one or more passive wheels to balance the robot [1]. The ease of design, ease of manufacturing, and ease of predicting the dynamics for this type of mobile robot make it a simple, suitable, and economic choice among the different ground contact mobile robots that are currently available [2].

The motion of a Hilare robot is controlled by controlling the velocities of the independent right and left wheels. When the wheels roll on the contact surface without lateral slip, non-holonomic behavior is induced in the robot motion. In the inverse kinematics of the mobile robot, this non-holonomic behavior appears as a constraint on the robot velocity that cannot be transformed into a position constraint [3]. Although the robot cannot move sideways, it can reach any desired position and orientation in the workspace by taking a complex trajectory. Since the robot cannot move sideways, finding an optimal trajectory from an initial point to a final point through closely packed obstacles, and controlling the robot through this trajectory, are challenging tasks. To serve this purpose, path planners and trajectory-tracking controllers are usually included in a control loop. The same tasks would be much easier if the mobile robot had the ability to move sideways.

A trajectory-tracking controller is the lowest level in a layered control architecture, and is essential for the motion control of any mobile robot. The main objective of the trajectory-tracking controller is to reduce the cross-track error [4]. Over the years, many approaches have been developed for the control of the Hilare robot [5]. The simplest approach to control the motion of this mobile robot is to use motion primitives, such as straight motion and in-place turning. A suitable combination of these primitives is used to control the robot through the predefined trajectory

* Corresponding author.
E-mail address: somashekhar@iitm.ac.in (S.S. Hiremath).

https://doi.org/10.1016/j.eng.2018.07.013

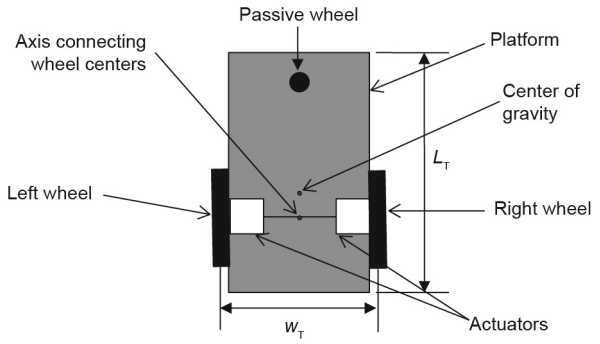

Fig. 1. Schematic diagram of a Hilare robot. L_T: length of the platform; w_T: width of wheel track.

[6,7]. However, this strategy may cause unwanted jerks and oscillations that affect the smoothness of the motion and the stability. Research on path-tracking controllers for a car-like robot [8] has shown that the stability of the controller is affected also by look-ahead distance. That work also suggests that a large look-ahead distance will result in cutting corners. According to the literature, other control methods such as adaptive control, the back-stepping method, artificial neural networks, fuzzy control, biologically inspired methods, potential field-based control, and so forth have been evolved over time for the smooth and stable tracking control of mobile robots [9–11]. Of these methods, a well-established method to control a mobile robot is a stable tracking controller, as proposed by Kanayama et al. [12]. This method is applicable to all types of autonomous mobile robots. The controller works such that the position and orientation error of the mobile robot with respect to a reference trajectory is reduced. The literature shows that this method can be modified further to develop smooth tracking controllers [13–15].

A reference trajectory for the controller can be obtained from the operator or from a path-planning algorithm. In general, path planning comes before the trajectory-tracking controller in the control hierarchy, and is very important in mobile robot navigation. Different approaches have been proposed, with different levels of complexity, accuracy, and applicability, for planning a feasible path for mobile robots [16]. Geometric path-planning algorithms [17], rapidly-exploring random trees (RRTs) [18], and probabilistic road maps (PRMs) [19] are some of the algorithms that can provide a reference trajectory to the tracking controller. A PRM is a well-known technique for planning an admissible path for a non-holonomic mobile robot. A PRM-based path planner can find the trajectory that must be followed by the mobile robot in order to reach the destination and to avoid the various obstacles.

Although many control methods have been developed for trajectory-tracking control, few efforts have been made to study a controller that is implemented in a layered control architecture in which the output of a path planner acts as an input to the controller. The available literature also lacks an explanation of the implementation of these controllers in Hilare robots with stepper motor-driven wheels. The current paper focuses on addressing these gaps in the literature. We propose a control strategy that can guide a mobile robot along a reference trajectory that is specified as a set of discontinuous waypoints through which the mobile robot must move in order to reach the destination. Parameter optimization has been carried out to minimize the average cross-track error and average linear velocity error of the mobile robot.

The paper is organized as follows: Section 2 presents the kinematic model and controller design; Section 3 presents the simulation and optimization of control parameters; experimentation is presented in Section 4; and conclusions and future work are detailed in Section 5.

2. Kinematic model and controller design

Consider a Hilare robot, as shown in Fig. 2, located on a two-dimensional surface for which a global coordinate system (inertial frame (X_I, Y_I)) is defined. The robot has three degrees of freedom on the surface. The posture (P_i) of the mobile robot at any given instant i constitutes the position (x_i, y_i) of the mobile robot and its heading angle (θ_i) (Eq. (1)). Here x_i and y_i are the inertial frame intercepts of the center of the axis connecting the actuated wheels. The direction perpendicular to the axis connecting the center of the actuated wheels is considered to be the heading direction of the mobile robot—that is, the X axis of the local frame connected to the mobile robot. The heading angle (θ_i) of the mobile robot is the angle between the X_I axis of the inertial frame and X axis of the local frame. Thus, in further calculations, the mobile robot is assumed to be a point body located at the center of the axis connecting the actuated wheels. The locus of the points (x_i, y_i) over time is considered to be the trajectory being tracked by the mobile robot.

$$P_i = \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} \tag{1}$$

In the present study, the trajectory to be tracked by the mobile robot is specified as a path connecting the waypoints from a start position (P_{w0} (x_{w0}, y_{w0})) to a destination position (P_{wn} (x_{wn}, y_{wn})) through a set of discontinuous waypoints (P_{wk} (x_{wk}, y_{wk})). Here, k is an integer between zero and the total number of waypoints (n) and w stands for waypoint. The linear and angular velocities of the mobile robot are given by v_i and ω_i, respectively. The rate of change in the posture ($\dot{P}_i$) of the mobile robot with respect to the linear velocity (v_i) and angular velocity (ω_i) of the robot can be resolved into the horizontal velocity component ($\dot{x}_i$), vertical velocity component ($\dot{y}_i$), and angular velocity components ($\dot{\theta}_i$) (Eq. (2)).

$$\dot{P}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i & 0 \\ \sin\theta_i & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} \tag{2}$$

To calculate the linear velocity and angular velocity, the right and the left wheels of the mobile robot are considered to be rotating at angular velocities of ω_{1i} and ω_{2i}, respectively. The linear velocity of the mobile robot due to the rotation of these wheels can be calculated by combining the individual wheel velocities, as shown in Fig. 3(a); the angular velocity can be calculated as shown in Fig. 3(b).

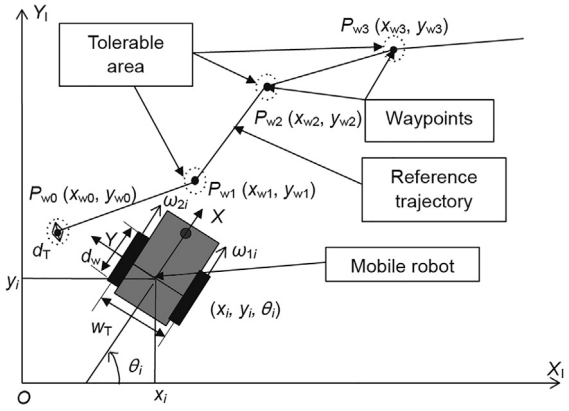

Fig. 2. Hilare robot tracking waypoints. P_i (x_i, y_i, θ_i): posture of the mobile robot; d_T: accepted transition distance; d_w: wheel diameter; ω_i: angular velocity.

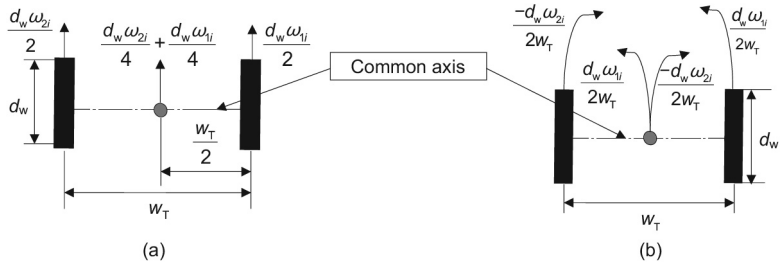

Fig. 3. (a) Linear velocity of the mobile robot; (b) angular velocity of the mobile robot.

The linear velocity v_i and the angular velocity ω_i of the mobile robot due to the rotation of the wheels is given in Eq. (3). Here, w_T is the width of the wheel track and d_w is the wheel diameter. To calculate these velocities, it is assumed that the wheels roll on the surface without slipping, and that these actuated wheels have the same diameter.

$$\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = \frac{1}{4}\begin{bmatrix} d_w & d_w \\ \frac{2d_w}{w_T} & \frac{-2d_w}{w_T} \end{bmatrix}\begin{bmatrix} \omega_{1i} \\ \omega_{2i} \end{bmatrix} \tag{3}$$

Eq. (4) can be obtained by taking the inverse transformation of Eq. (3). This equation describes the transformation of the velocities (v_i, ω_i) of the mobile robot into the wheel angular velocities (ω_{1i}, ω_{2i}).

$$\begin{bmatrix} \omega_{1i} \\ \omega_{2i} \end{bmatrix} = \frac{1}{d_w}\begin{bmatrix} 2 & w_T \\ 2 & -w_T \end{bmatrix}\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} \tag{4}$$

The posture error ($P_{ei} = [x_{ei}, y_{ei}, \theta_{ei}]^T$) of the mobile robot can be found by transforming the global posture error of the mobile robot into the local coordinates (X, Y) of the mobile robot, as shown in Eq. (5). The global posture error is calculated with respect to the inertial frame (X_I, Y_I). Here, $[x_{wk}, y_{wk}, \theta_{wk}]^T$ is the required waypoint posture in the global coordinates.

$$P_{ei} = \begin{bmatrix} x_{ei} \\ y_{ei} \\ \theta_{ei} \end{bmatrix} = \begin{bmatrix} \cos\theta_i & \sin\theta_i & 0 \\ -\sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_{wk} - x_i \\ y_{wk} - y_i \\ \theta_{wk} - \theta_i \end{bmatrix} \tag{5}$$

θ_{wk} is calculated as follows:

$$\theta_{wk} = \tan^{-1}\left(\frac{y_{wk} - y_i}{x_{wk} - x_i}\right) \tag{6}$$

A kinematic controller is employed to reduce this posture error, and thus control the robot through the desired trajectory. The stable tracking controller proposed by Kanayama et al. [12] is utilized to control the mobile robot such that the posture error will converge to zero. This control law is given by Eq. (7), in which v_{ci} and ω_{ci} are the control velocities, v_{ri} and ω_{ri} are the reference velocities of the mobile robot, and k_1, k_2, and k_3 are the control gains.

$$\begin{bmatrix} v_{ci} \\ \omega_{ci} \end{bmatrix} = \begin{bmatrix} v_{ri}\cos\theta_{ei} + k_1 x_{ei} \\ \omega_{ri} + k_2 v_{ri} y_{ei} + k_3 v_{ri}\sin\theta_{ei} \end{bmatrix} \tag{7}$$

The reference waypoint of the controller should be updated based on the instantaneous posture of the mobile robot so that the mobile robot can be controlled through the trajectory-connecting waypoints. A waypoint update function is applied in the present work in order to calculate the Euclidean distance from the current position of the mobile robot to the reference position. If this distance is within an accepted transition distance d_T, the reference posture will be updated and the next waypoint posture will be selected as the new reference posture. Hence, the accepted transition area will be a circle of fixed radius d_T around the reference posture. If the robot reaches any point within this circle, then the reference posture will be updated. The constraint on this transition is given by Eq. (8). The distance d_T works in a similar way to the look-ahead distance [8] mentioned in the previous section; hence, the selection of d_T is critical. With consideration to cutting corners and making wide turns, the d_T selected in the present work is 27 mm, which is half of the wheel track width.

$$\sqrt{x_{ei}^2 + y_{ei}^2} \le d_T \tag{8}$$

The architecture of the waypoint tracking controller is shown in Fig. 4. It consists of a waypoint planner, which provides a set of waypoints from P_{w0} to P_{wn} that make up the reference trajectory. The waypoint selector module selects the reference waypoint from this set. The waypoint selector takes the waypoint P_{w1} as the first reference point; later, it updates the waypoint when the criterion given in Eq. (8) is satisfied. The error estimation block uses Eq. (5) to generate the posture error. The controller uses this posture error and reference velocity to generate the control velocities v_{ci} and ω_{ci}, as given in Eq. (7). The next module takes care of the input constraints and calculates the wheel velocities corresponding to the control velocities. The step command generator produces

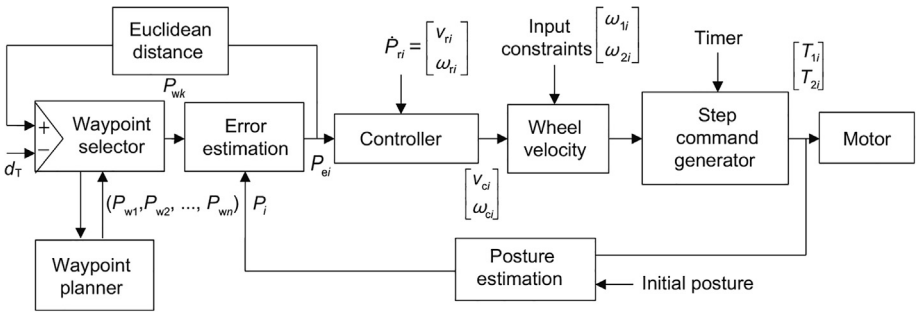

Fig. 4. Architecture of waypoint tracking controller. T_{1i} and T_{2i} are the triggering signals of the right and left motors, respectively.

the trigger to control the stepper motor. The new posture of the mobile robot is calculated by the posture estimation module. Output of this module acts as feedback. A flow chart of the control algorithm is provided in Appendix A.

The output of the controller module cannot be applied directly to a real robot. The magnitudes of the control velocities will be very large because of the discontinuities in the reference path. Various constraints on the motor may restrict high-magnitude control velocities being implemented on a real mobile robot. These constraints are referred to as "input constraints." Examples of input constraints include the maximum attainable angular velocity of the driving motor, and user-defined restrictions on the maximum linear and angular velocities of the mobile robot. Fig. 5 shows the different input constraints applied to the velocity space of a forward-moving Hilare robot. To satisfy these input constraints, the control output (Eq. (7)) must be modified. The user-defined restrictions on the linear velocity, v_{max} and v_{min}, of the mobile robot can be introduced into the controller by modifying the control velocity v_{ci} using Eqs. (9) and (10), respectively. Here, v_{im1} and v_{im2} are the modified control velocities.

$$v_{im1} = \alpha(v_{max},\ v_{ci}) \cdot (v_{ci} - v_{max}) + v_{max} \tag{9}$$

$$v_{im2} = \alpha(v_{ci},\ v_{min}) \cdot (v_{ci} - v_{min}) + v_{min} \tag{10}$$

where α is a condition parameter , which is defined by Eq. (11).

$$\alpha(m,\ n) = \begin{cases} 0,\ \mathrm{m} \le n \\ 1,\ \mathrm{m} > n \end{cases} \tag{11}$$

where m and n can be any arbitrary value.

Similarly, the user-defined restrictions on the angular velocity of the mobile robot, ω_{max} and ω_{min}, can be introduced into the controller by modifying the control velocity ω_{ci}, as shown in Eqs. (12) and (13), respectively. Here, ω_{im1} and ω_{im2} are modified control velocities.

$$\omega_{im1} = \gamma(\omega_{ci}) \cdot [\alpha(\omega_{max},\ |\omega_{ci}|) \cdot (|\omega_{ci}| - \omega_{max}) + \omega_{max}] \tag{12}$$

$$\omega_{im2} = \gamma(\omega) \cdot [\alpha(|\omega_{ci}|,\ \omega_{min}) \cdot (|\omega_{ci}| - \omega_{min}) + \omega_{min}] \tag{13}$$

where $\gamma(\omega)$ is a condition parameter for the angular velocity, which is given by Eq. (14):

$$\gamma(\omega) = \begin{cases} -1,\ \omega < 0 \\ 1,\ \omega \ge 0 \end{cases} \tag{14}$$

Eq. (15) is obtained by combining Eqs. (9) and (10). Here, v_{im} indicates the modified linear velocity.

$$\begin{aligned} v_{im} = {} & \alpha(v_{max},\ v_{ci}) \cdot (v_{ci} - v_{max}) + v_{max} + \alpha(v_{ci},\ v_{min}) \\ & \cdot (v_{ci} - v_{min}) + v_{min} - v_{ci} \end{aligned} \tag{15}$$

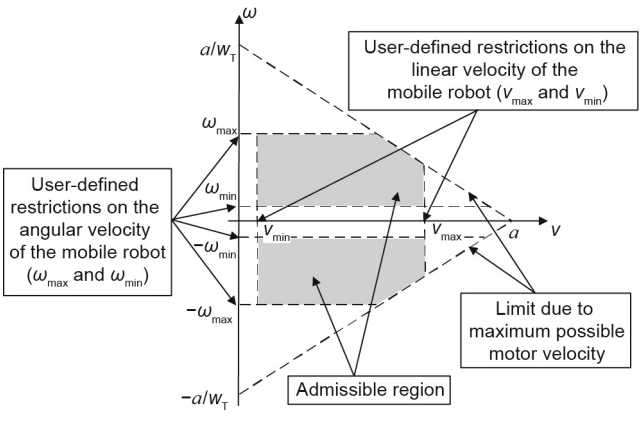

Fig. 5. Different input constraints applied to the velocity space.

Similarly, Eq. (16) is obtained by combining Eqs. (12) and (13). Here, ω_{im} indicates the modified angular velocity.

$$\begin{aligned} \omega_{im} = {} & \gamma(\omega_{ci}) \cdot [\alpha(\omega_{max},\ |\omega_{ci}|) \cdot (|\omega_{ci}| - \omega_{max}) + \omega_{max} \\ & + \alpha(|\omega_{ci}|,\ \omega_{min}) \cdot (|\omega_{ci}| - \omega_{min}) + \omega_{min} - |\omega_{ci}|] \end{aligned} \tag{16}$$

The maximum possible linear velocity (a) can be calculated as shown in Eq. (17), by assuming that both motors have the same properties and both wheels are of the same dimension. Here, ω_{1max} and ω_{2max} are the maximum possible angular velocities of the right and left motors, respectively.

$$a = \frac{d_w \cdot \omega_{1max}}{2} = \frac{d_w \cdot \omega_{2max}}{2} \tag{17}$$

The constraints on the linear velocity and on the angular velocity of the mobile robot due to this upper limit on the angular velocity of the motor can be found by combining Eqs. (17) and (4), as shown below, and by replacing ω_1 and ω_2 with ω_{1max} and ω_{2max} in Eq. (4).

$$\begin{bmatrix} \omega_{1max} \\ \omega_{2max} \end{bmatrix} \ge \frac{1}{d_w} \begin{bmatrix} 2 & w_T \\ 2 & -w_T \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} \tag{18}$$

$$\begin{bmatrix} \frac{d_w}{2}\omega_{1max} \\ \frac{d_w}{2}\omega_{2max} \end{bmatrix} \ge \begin{bmatrix} v_i + \frac{\omega_i w_T}{2} \\ v_i - \frac{\omega_i w_T}{2} \end{bmatrix} \tag{19}$$

$$\begin{bmatrix} a \\ a \end{bmatrix} \ge \begin{bmatrix} v_i + \frac{\omega_i w_T}{2} \\ v_i - \frac{\omega_i w_T}{2} \end{bmatrix} \tag{20}$$

$$a \ge v_i + \frac{\omega_i w_T}{2} \text{ and } a \ge v_i - \frac{\omega_i w_T}{2} \tag{21}$$

Considering that ω_i in the above equations can be positive or negative, the constraint on the linear velocity and on the angular velocity is given by Eq. (22).

$$v_i + \left|\frac{w_T \omega_i}{2}\right| \le a \tag{22}$$

The linear velocity and angular velocity calculated in Eqs. (15) and (16) are further modified, as shown in Eqs. (23) and (24), to satisfy the restrictions on the maximum attainable angular velocity of the driving motor, and thus to generate the final control velocities. These control velocities can be implemented on an actual mobile robot. Eqs. (23) and (24) are implemented in the wheel velocity module of the control loop.

$$v_i = \alpha\left[\left(v_{im} + \left|\frac{w_T \omega_{im}}{2}\right|\right),\ a\right] \cdot \left(\frac{v_{im} \cdot a}{v_{im} + \left|\frac{w_T \omega_{im}}{2}\right|} - v_{max}\right) + v_{im} \tag{23}$$

$$\omega_i = \gamma(\omega_{im}) \left\{ \alpha\left[\left(v_{im} + \left|\frac{w_T \omega_{im}}{2}\right|\right),\ a\right] \cdot \left(\frac{|\omega_{im}| \cdot a}{v_{im} + \left|\frac{w_T \omega_{im}}{2}\right|} - |\omega_{im}|\right) + |\omega_{im}| \right\} \tag{24}$$

To apply the calculated control velocities and execute the control action, the control velocities must be transformed into the corresponding motor control input. In the current scenario, the wheels of the mobile robot are actuated by a stepper motor. The control signal to the stepper motor is usually generated with the help of a timer module. The timer count c used in the timer provides the necessary delay between each step. This time delay Δt is given in Eq. (25), where f is the frequency of the timer.

$$\Delta t = \frac{c}{f} \tag{25}$$

The linear velocity and angular velocity of the mobile robot, which are calculated in Eqs. (23) and (24), are applied in Eq. (4) in order to obtain the angular velocities ω_{1i} and ω_{2i} at which the

right and left motors must rotate. Angles $\Delta\theta_{1i}$ and $\Delta\theta_{2i}$, which should be moved by each motor within the given time interval in order to achieve the required linear velocity and angular velocity of the mobile robot, can be found by multiplying the angular velocity of the wheel with the time interval, as shown in Eq. (26). Here, T_{lag} is the time lag that may occur in calculating the required wheel velocity.

$$\begin{bmatrix}\Delta\theta_{1i}\\ \Delta\theta_{2i}\end{bmatrix} = \begin{bmatrix}\omega_{1i}\\ \omega_{2i}\end{bmatrix} \cdot T_{\text{lag}} \tag{26}$$

However, the stepper motor can only move by a fixed angle (β) in each step. This imposes restrictions on the wheel rotation; that is, the wheels should rotate only if the required angle of rotation is greater than β. To satisfy this restriction and to obtain smooth motion without the effect of T_{lag}, the motor step command (i.e., motion command) is generated with the help of a timer interrupt-driven subroutine, as shown in Fig. 4, while the wheel velocity calculation runs in the main control loop. This allows the robot to move and update the posture (i.e., execute the previously calculated velocity commands) while calculating the required wheel velocity commands for the next instance. The timer is kept such that it will operate at the maximum step frequency of the stepper motor, which is 1 kHz in the present study. In the subroutine, the calculated values of $\Delta\theta_{1i}$ and $\Delta\theta_{2i}$ are added, respectively, to the unmoved angles $\theta_{1(i-1)}$ and $\theta_{2(i-1)}$ that were calculated in the previous cycle of the subroutine. If the sum calculated using Eq. (27) is larger than β, then stepper motor triggering occurs. This triggering criterion is provided in Eq. (28), where T_{1i} and T_{2i} are the triggering signals of the right and left motors, respectively. This timer-driven motor step command generation ensures real-time control of the robot.

$$\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix} = \begin{bmatrix}\theta_{1(i-1)}\\ \theta_{2(i-1)}\end{bmatrix} + \begin{bmatrix}\Delta\theta_{1i}\\ \Delta\theta_{2i}\end{bmatrix} \tag{27}$$

$$T_{1i} = \begin{cases}1, & \theta_1 \geq \beta\\ 0, & \text{otherwise}\end{cases} \text{ and } T_{2i} = \begin{cases}1, & \theta_2 \geq \beta\\ 0, & \text{otherwise}\end{cases} \tag{28}$$

After generating the signals to the motors, the value of the unmoved angle is updated, as shown in Eq. (29):

$$\begin{bmatrix}\theta_{1i}\\ \theta_{2i}\end{bmatrix} = \begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix} - \begin{bmatrix}T_{1i}\cdot\beta\\ T_{2i}\cdot\beta\end{bmatrix} \tag{29}$$

Eqs. (26)–(29) are implemented in the step command generator module of the control loop. The actual angular velocity of the motor is calculated using Eq. (30):

$$\begin{bmatrix}\omega_{1i}\\ \omega_{2i}\end{bmatrix} = \frac{1}{\Delta t}\begin{bmatrix}T_{1i}\cdot\beta\\ T_{2i}\cdot\beta\end{bmatrix} \tag{30}$$

The new posture P_{i+1} of the mobile robot, as given in Eq. (31), is obtained by combining Eqs. (1) and (2). Here, the rate of change of the posture is obtained by applying Eq. (30) in Eq. (3), in order to obtain the actual velocities v_i and ω_i of the mobile robot in the given time interval, and then substituting the linear and angular velocities into Eq. (2).

$$P_{i+1} = \begin{bmatrix}x_{i+1}\\ y_{i+1}\\ \theta_{i+1}\end{bmatrix} = \begin{bmatrix}x_i\\ y_i\\ \theta_i\end{bmatrix} + \begin{bmatrix}\dot{x}_i\\ \dot{y}_i\\ \dot{\theta}_i\end{bmatrix} \cdot \Delta t \tag{31}$$

This equation is implemented in the posture estimation module of the controller in order to find the instantaneous posture of the mobile robot.

3. Simulation and optimization of control parameters

The multi-objective particle swarm optimization (MOPSO) technique [20] was performed to select the control parameters, such as the control gains (k_1, k_2, k_3) and the maximum linear and angular velocity limits ($v_{\max}$, $\omega_{\max}$) of the proposed controller, so that the tracking errors would be minimal. The objectives of the optimization were to minimize the average cross-track error and the average linear velocity error of the mobile robot, while tracking a step-shaped trajectory using the controller mentioned in the previous section. The cross-track error was calculated as the distance from the mobile robot to the line connecting the current waypoint with the previous waypoint. The linear velocity error is the difference between the measured velocity and the reference velocity (v_{ri}). The optimization was carried out based on the simulation results. In this simulation, these errors were calculated by simulating the trajectory that will be tracked by the mobile robot when the proposed controller is used. The parameters used for the simulation are listed in Table 1. These parameters were selected based on the specifications of a commercially available research platform e-puck robot, on which the experiments were carried out. The MOPSO algorithm performed 20 000 evaluations to produce local Pareto front of 50 particles. Each evaluation consisted of simulating the trajectory tracked by the mobile robot for a particular set of control parameters selected by MOPSO. The range within which the control parameters were optimized is given in Table 2.

Fig. 6 shows the Pareto front produced with the help of the MOPSO for the objective functions. To produce this Pareto front, the MOPSO utilized the average linear velocity error and the average cross-track error calculated in the simulation. Each point on the Pareto front represents a set of optimal parameters. The reference step-shaped trajectory and the simulated trajectory for an optimized parameter set (k_1 = 0.0247, k_2 = 2.9671, k_3 = 0.552, $v_{\max}$ = 41 mm·s^{-1}, $\omega_{\max}$ = 0.747 rad·s^{-1}) selected from the Pareto front is shown in Fig. 7. The point on the Pareto front corresponding to this set is marked as (A) in Fig. 6.

Fig. 8 shows the linear velocity of the mobile robot while tracking the step-shaped trajectory during the simulation. The average cross-track error and the average linear velocity error obtained from this simulation were 2.63 mm and 2.62 mm·s^{-1}, respectively. These errors were calculated by taking the average of the error produced in each iteration during the simulation.

Table 1
Parameters used for simulation.

Parameter	Symbol	Value
Wheel track width	w_{T}	54 mm
Diameter of wheel	d_{w}	40 mm
Step size	–	0.13 mm
Maximum motor velocity	$\omega_{1\max}$, $\omega_{2\max}$	3.25 rad·s^{-1}
Time delay	Δt	1 ms
Reference linear velocity	v_{ri}	40 mm·s^{-1}
Reference angular velocity	ω_{ri}	0 rad·s^{-1}
Minimum linear velocity	$v_{\min}$	0 mm·s^{-1}
Minimum angular velocity	$\omega_{\min}$	0 rad·s^{-1}

Table 2
Parameter range selected for optimization.

Parameter	Symbol	Range
Control gain parameter	k_1	0–5
	k_2	0–5
	k_3	0–5
Maximum linear velocity	$v_{\max}$	40–100 mm·s^{-1}
Maximum angular velocity	$\omega_{\max}$	0–0.75 rad·s^{-1}

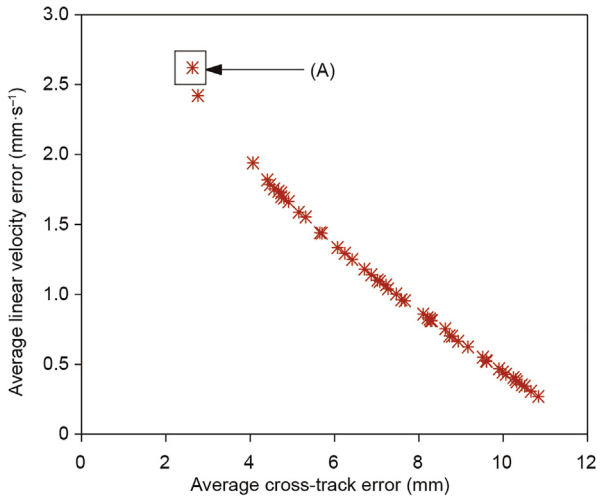

Fig. 6. Pareto front produced with the help of the MOPSO.

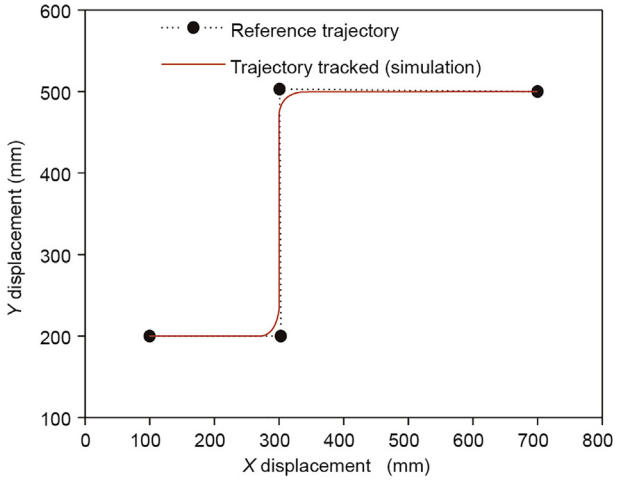

Fig. 7. Trajectory of the mobile robot when tracking the step-shaped reference trajectory.

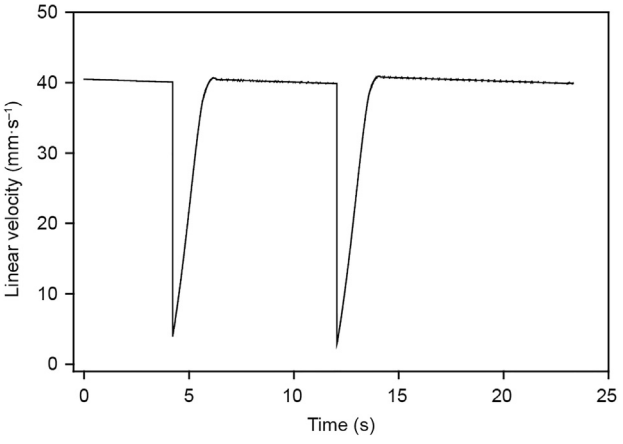

Fig. 8. Linear velocity of the mobile robot when tracking the step-shaped reference trajectory.

4. Experimentation

4.1. Experimental setup

Fig. 9 shows the experimental setup, which consists of a mobile robot in an indoor environment on top of a smooth platform that

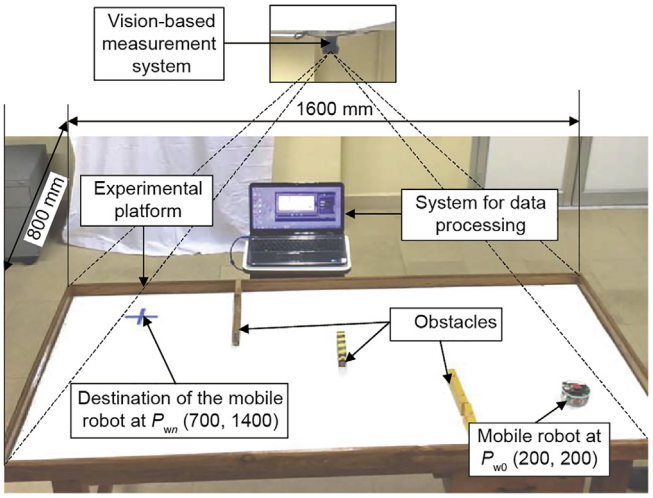

Fig. 9. Photographic view of the instrumented experimental setup.

was developed for the experiments. A vision-based measurement system placed over the platform was utilized to analyze the motion of the mobile robot. This system measures the linear velocity of the mobile robot as well as the locus of the points through which the mobile robot moved. The velocity of the mobile robot was limited by user-defined restrictions: $v_{max} = 41$ mm·s^{-1}, $v_{min} = 0$ mm·s^{-1}, $\omega_{min} = 0$ rad·s^{-1}, and $\omega_{max} = 0.747$ rad·s^{-1}. The selected control gains for the experiment were $k_1 = 0.0247$, $k_2 = 2.9671$, and $k_3 = 0.552$. The point corresponding to this combination of control gains was selected from the Pareto front by giving proper weightage to the average cross-track error and average linear velocity error. The maximum velocity limits and control gains were selected based on the optimization results explained in the previous section. The mobile robot uses the motor step count as a feedback signal. The control cycle consists of two parts—the control loop and a timer interrupt subroutine—which was implemented in a dsPIC30F6014A microcontroller system. The main control loop calculates the required wheel velocity, based on the control strategy developed in Section 2. The interrupt subroutine generates the control command to the motors to rotate (step command generator module in Fig. 4) and calculates the new posture of the mobile robot (position estimation module in Fig. 4). The control cycle was 5 ms, but the control signal to the motor was produced every millisecond with the help of the timer interrupt ($T_{lag} = 5$ ms, $\Delta t = 1$ ms). This strategy allows the mobile robot to move smoothly, without any jerks occurring due to lag in the motor control signal. Since the motor control commands are generated every millisecond, the mobile robot can be instructed to move at a linear velocity of 130 mm·s^{-1}, which is equal to 1000 steps·s^{-1}.

The mobile robot was initially placed on the platform at a known location and orientation (x_0, y_0, θ_0) = (200 mm, 200 mm, 0 rad). It was connected wirelessly to a computer, and the reference trajectory to be tracked was communicated to the mobile robot through this connection. These reference trajectories were communicated as a set of waypoints that connect to form the trajectory. During the experiment, video of the mobile robot motion over the platform was captured using the vision-based measurement system. This video was post-processed to obtain the actual linear velocity of the mobile robot and the trajectory tracked by the mobile robot. These measurements were then compared with the reference linear velocity and trajectory to determine the effectiveness of the controller in tracking different trajectories. Self-localization of the robot was carried out with the help of the posture estimation module in the control loop, as defined in Eq. (31) in Section 2.

4.2. Experimental results

Experiments were carried out to track three different trajectories: namely, a trajectory generated by a path planner, a rectangular trajectory, and a trajectory that resembles cursive script in English. These trajectories were selected based on the different applications this mobile robot may be used for (i.e., transportation, manufacturing, human assistance). Fig. 9 shows the experimental setup, with obstacles placed on the experimental platform. This particular configuration was used to carry out experiments to track the path-planner-generated trajectory. In the other two experiments, these obstacles were not present.

4.2.1. Path-planner-generated trajectory

In a fully autonomous system, the path planner provides the path that should be tracked by the mobile robot. Thus, in this experiment, the mobile robot was controlled from a start position (200, 200) to a destination position (700, 1400) and avoided obstacles through a trajectory described by a set of waypoints generated by a path planner. The PRM-based path planner implemented in the computer connected to the mobile robot was utilized to generate the waypoints. The locations of the obstacles were manually fed to the PRM to generate the waypoints. The algorithm provides a set of waypoints that describe the reference trajectory. The waypoints were then communicated to the robot using the wireless connection mentioned earlier in this section.

Fig. 10(a) and (b) shows the trajectory tracked by the mobile robot and the linear velocity error, respectively. The average cross-track error during the experiment was found to be 12.88 mm, and the average linear velocity error was found to be 1.24 mm·s^{-1}. The average linear velocity error was minimal when compared with the other trajectory-tracking experiments explained in this paper. This was due to the closeness of the waypoints to each other and to a lower number of sharp turns in the path. Sharp turns are difficult for the mobile robot to follow, due to angular velocity constraints.

4.2.2. Rectangular trajectory

A rectangle with 400 mm of length and 300 mm of width was tracked using the developed controller. The corner points of the rectangle, at (200, 200), (600, 200), (600, 500), and (200, 500), were selected as the waypoints. These waypoints were manually selected and passed to the robot through a wirelessly connected computer. Fig. 11(a) shows the trajectory tracked by the mobile robot and Fig. 11(b) shows the linear velocity error while tracking the rectangular trajectory. The average cross-track error during the experiment was found to be 6.75 mm, and the average linear velocity error was found to be 2.91 mm·s^{-1}.

4.2.3. Cursive script trajectory

An attempt was made to track a complex trajectory that resembles cursive script in English. The waypoints in the trajectory were generated through a user interface: A human operator used a mouse to mark the waypoints corresponding to the text/path to be followed by the mobile robot. An algorithm linked with the user interface selected these waypoints and communicated them to the mobile robot. The user interface was implemented on the computer that was wirelessly connected to the robot. The text selected in this case was the word "So." This particular word was selected because it provides a trajectory with no isolated points, and can be tracked by the forward motion of the mobile robot. Fig. 12(a) shows the trajectory tracked by the mobile robot and Fig. 12(b) shows the linear velocity error while tracking the trajectory of the cursive script. The average cross-track error was found to be 9.45 mm and the average linear velocity error was found to be 2.46 mm·s^{-1}.

Table 3 provides a summary of the results. The results show that the average cross-track error was highest for the path-planner-generated trajectory (12.88 mm) and lowest for the rectangular trajectory (6.75 mm). This finding contradicts the simulation results. The main reason for the differences between the simulation results and the experimental results is that a wheel slip occurred in the experiment that was not accounted for in the model. Uncertainty in the start position and in the orientation of the mobile robot when it was placed on the experiment platform also contributed to these differences. The average linear velocity error was lowest (1.24 mm·s^{-1}) for the path-planner-generated trajectory and highest for the rectangular trajectory (2.91 mm·s^{-1}). The linear velocity errors predicted in the simulation closely matched those of the experiments. Although there was a mismatch between the predicted and actual cross-track error, the errors were small when compared with the specified trajectory, so the controller was reliable.

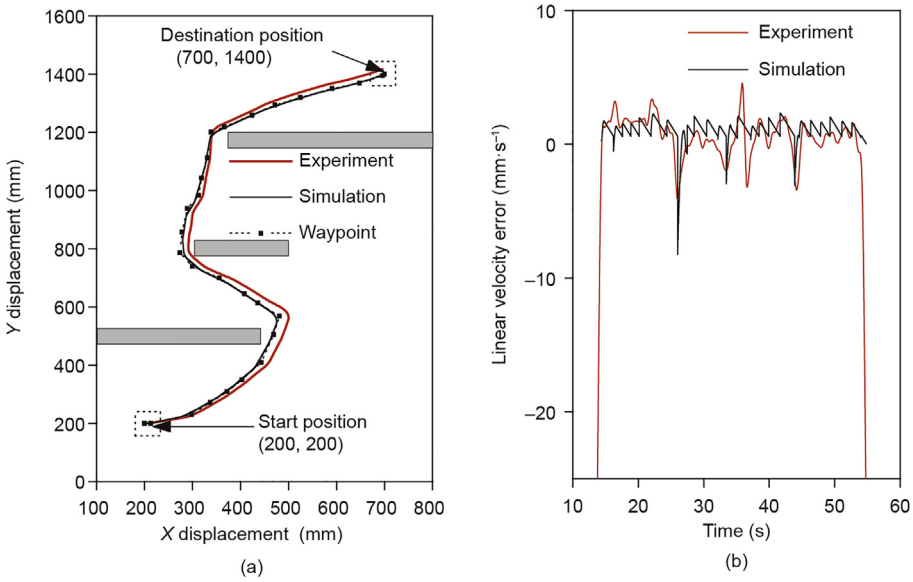

Fig. 10. Path-planner-generated trajectory. (a) Trajectory tracked by the mobile robot; (b) linear velocity error.

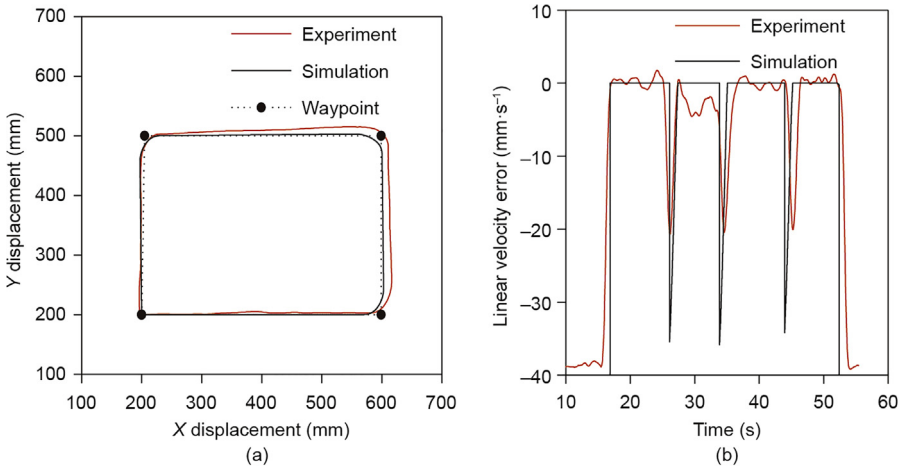

Fig. 11. Rectangular trajectory. (a) Trajectory tracked by the mobile robot; (b) linear velocity error.

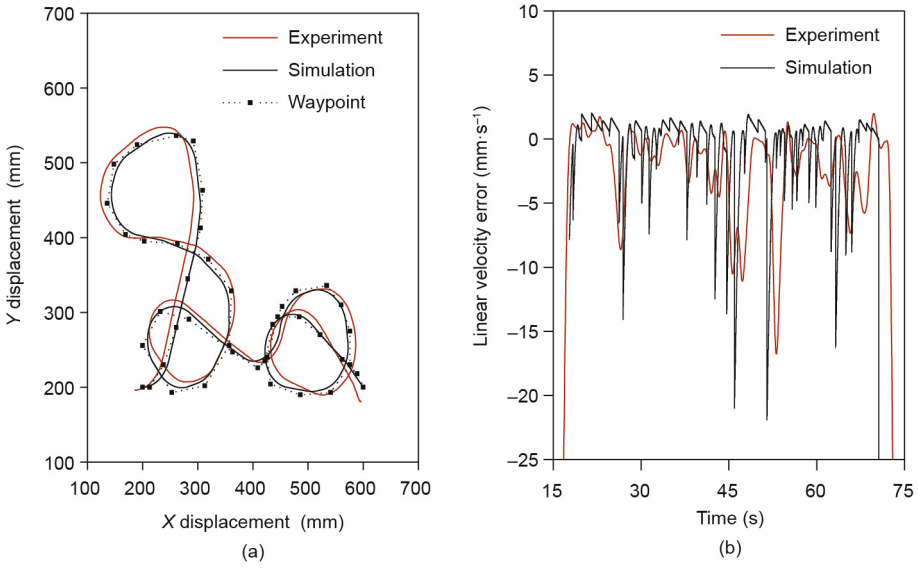

Fig. 12. Cursive script trajectory. (a) Trajectory tracked by the mobile robot; (b) linear velocity error.

Table 3
Summary of results.

Type of trajectory	Total number of waypoints	Experimentation	
		Average cross-track error (mm)	Average linear velocity error ($mm \cdot s^{-1}$)
Path-planner-generated	28	12.88	1.24
Rectangular	4	6.75	2.91
Cursive script	43	9.45	2.46

5. Conclusions

This paper addressed the control of a stepper motor-driven Hilare robot to track trajectories described by a set of waypoints, and explained the implementation of the controller on an actual mobile robot. The conclusions are as follows:

(1) The major contribution of this work is that it extends the adaptive control strategy to account for directional discontinuities and velocity constraints when controlling a stepper motor-driven mobile robot.

(2) The average cross-track error and the average linear velocity error were reduced by optimizing the control parameters.

(3) The average cross-track error was found to be 12.88 mm for the path-planner-generated trajectory, 6.75 mm for the rectangular trajectory, and 9.45 mm for the cursive script trajectory.

(4) The average linear velocity error was found to be less than 3 $mm \cdot s^{-1}$ for all the trajectories. This experimental result implies that the proposed method is robust and reliable in tracking different types of trajectories.

(5) The future work is toward reducing the cross-track error and to integrate localization module to avoid error in self localization.

Compliance with ethics guidelines

Robins Mathew and Somashekhar S. Hiremath declare that they have no conflict of interest or financial conflicts to disclose.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.eng.2018.07.013.

References

[1] Dudek G, Jenkin M. Computational principles of mobile robotics. 2nd ed. New York: Cambridge University Press; 2010.
[2] Konduri S, Torres EOC, Prabhakar R. Dynamics and control of a differential drive robot with wheel slip: application to coordination of multiple robots. J Dyn Syst Meas Control 2017;139(1):014505.
[3] Fahimi F. Autonomous robots: modeling, path planning, and control. New York: Springer; 2008.
[4] da Silva J, de Sousa J. A dynamic programming based path-following controller for autonomous vehicles. Contr Intell Syst 2011;39(4):245–53.
[5] Kolmanovsky I, McClamroch NH. Developments in nonholonomic control problems. IEEE Control Syst 1995;15(6):20–36.
[6] Nagy Á, Csorvási G, Kiss D. Path planning and control of differential and car-like robots in narrow environments. In: Proceedings of the 13th International Symposium on Applied Machine Intelligence and Informatics; 2015 Jan 22–24; Herl'any, Slovakia; 2015. p. 103–8.
[7] Mathew R, Hiremath SS. Trajectory tracking and control of differential drive robot for predefined regular geometrical path. Procedia Technol 2016;25:1273–80.
[8] Snider JM. Automatic steering methods for autonomous automobile path tracking. Pittsburgh: Carnegie Mellon University; 2009.
[9] Park B, Yoo SJ, Park JB, Choi YH. A simple adaptive control approach for trajectory tracking of electrically driven nonholonomic mobile robots. IEEE Trans Contr Syst Technol 2010;18(5):1199–206.
[10] Valbuena L, Tanner HG. Hybrid potential field based control of differential drive mobile robots. J Intell Robot Syst Theory Appl 2012;68(3–4):307–22.
[11] Chen X, Jia Y, Matsuno F. Tracking control for differential-drive mobile robots with diamond-shaped input constraints. IEEE Trans Contr Syst Technol 2014;22(5):1999–2006.
[12] Kanayama Y, Kimura Y, Miyazaki F, Noguchi T. A stable tracking control method for an autonomous mobile robot. In: Proceedings of IEEE International Conference on Robotics and Automation; 1990 May 13–18; Cincinnati, OH, USA. New York: IEEE; 1990. p. 384–9.
[13] Fukao T, Nakagawa H, Adachi N. Adaptive tracking control of a nonholonomic mobile robot. IEEE Trans Automat Contr 2000;16(5):609–15.
[14] Maalouf E, Saad M, Saliah H. A higher level path tracking controller for a four-wheel differentially steered mobile robot. Robot Auton Syst 2006;54 (1):23–33.
[15] Guo J, Lin Z, Cao M, Yan G. Adaptive control schemes for mobile robot formations with triangularised structures. IET Control Theory Appl 2010;4 (9):1817–27.
[16] Miao Y, Khamis AM, Karray F, Kame MS. A novel approach to path planning for autonomous mobile robots. Contr Intell Syst 2011;39(4):235–44.
[17] Laumond JP, Jacobs PE, Taix M, Murray RM. A motion planner for nonholonomic mobile robots. IEEE Trans Robot Autom 1994;10 (5):577–93.
[18] LaValle S, Tennoe M, Henssonow S, editors. Rapidly-exploring random trees: a new tool for path planning. Whitefish: Betascript Publishing; 1998.
[19] Kavraki L, Svestka P, Latombe JC, Overmars MH. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robot Autom 1996;12(4):566–80.
[20] Coello Coello CA, Leehuga MS. MOPSO: a proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation; 2002 May 12–17; Honolulu, HI, USA; 2002. p. 1051–6.