Chapter 3

Adaptive Steering and Trajectory Control of Wheeled Mobile Robots for Autonomous Navigation

Mariam Al-Sagban and Rached Dhaouadi

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/64227

Abstract

This chapter presents a new reactive navigation algorithm for a wheeled mobile robot (WMR) with a differential drive mechanism moving in unknown environments [1]. The mobile robot is controlled to travel to a predefined goal position safely and efficiently without any prior map of the environment. The navigation is achieved by modulating the steering angle and turning radius. To avoid obstacles while seeking the goal position, the dimensions and shape of the robot are incorporated to determine the set of all possible collision-free steering angles. The algorithm then selects the optimum steering angle candidate to contour the obstacle. Simulation and experimental results on a WMR prototype are used to validate the proposed algorithms.

Keywords: recurrent neural networks, obstacle avoidance, robots

1. Introduction

Over the years, mobile robots have evolved rapidly incorporating a wide spectrum of applications. They aid within the field of medical technologies, assist in vehicle driving, and can be occupied for use within hazardous rescue missions. Mobile robots helped monitor the spread of oil during the catastrophic spill on the Gulf of Mexico [2]. Additionally, robots were of great aid during the Japanese Fukushima nuclear crisis in monitoring the radiation levels and cleaning up leftover debris [3].

In performing all previous tasks, mobile robots must be equipped with autonomous navigation. This is described as the arrival at of the robot at a target location without any assistance, and whilst avoiding obstacles present around. Perception, path planning, localization, and

motion control are the key elements that build toward autonomous navigation. With perception, sensors pick up information regarding the robots immediate environment that are then perceived and translated within the robot. The next step is path planning, which is the robot's ability to come to a consensus regarding the required action as a result of its expected goal. The final step occurs with motion control through which the robot executes the required action through its actuators [4].

In this chapter, we focus on path planning in unknown environments [5]. Path planning represents a key feature of autonomous navigation. The problem of path planning is usually classified in three categories according to the given environment and constraints:

- Global Path Planning—This framework requires full knowledge of the robot workspace; a global map is supplied as given input.
- Local Path Planning (Sensor-Based Path Planning)—This framework requires partial knowledge of the workspace. Therefore, only an incomplete map is supplied.
- Reactive Navigation (Obstacle Avoidance)—In this framework, no a priori information is required about the workspace. Instead, obstacles are discovered in real time while the robot is executing its motion.

Due to the inherent nature of the obstacle avoidance problem, navigation algorithms do not produce efficient paths and do not guarantee global convergence as would global path planning algorithms. This would result the robot in producing inefficient paths or failing to reach the goal position (trap position).

The implementation of path planning and obstacle avoidance techniques on a real mobile robot imposes different types of constraints including kinematic, dynamic, and time constraints. The differential drive robot must follow a curve path due to the kinematic constraint as it is unable to reach the desired point place instantly and in a short period of time. Disregarding this limitation, when dealing with path planning and obstacle avoidance techniques, would lead to an unsafe design as the transitional curve may potentially intersect with obstacles and, therefore, result in a collision.

2. Autonomous navigation concepts

2.1. The configuration space

For a two-dimensional robot, the robot configuration can be fully described by rigidly attaching a frame to the robot and then specifying the position and orientation of this frame in the global frame. The complete specification of the location of every point on the robot is called a configuration, q. The set of all possible configurations is called a configuration space or C-space ($q \in \mathcal{C}$). A rigid object moving in a plane is, therefore, specified by the triple configuration, $q = (x, y, \theta)$, and the configuration space can be represented by $\mathcal{C} = \mathbb{R}^2 \times SO(2)$, where $SO(2)$ is the special orthogonal group of 2-D rotations [6].

http://dx.doi.org/10.5772/64227

The introduction of new notation is important for the description of collisions. The workspace in which the robot moves will be denoted as $\mathcal{W}$. When the robot moves in a plane, the workspace is now denoted as $W = \mathbb{R}^2$. The subset of this workspace occupied by obstacles is then denoted as $\mathcal{O} \subset \mathcal{W}$, and the subset occupied by the robot at configuration is denoted as $\mathcal{A}(q) \subset \mathcal{W}$. The robot must avoid a configuration causing it to come into physical contact with any obstacle, as this would cause a collision otherwise. The set of configurations in which the robot would come into a collision with an obstacle is defined as the obstacle configuration space,

$$\mathcal{C}_{obst} = \{q \in \mathcal{C} \mid \mathcal{A}(q) \bigcap \mathcal{O} = \emptyset\}. \tag{1}$$

On the other hand, the set of all collision-free configurations is defined as the free configuration space. It is defined as the set difference

$$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst} \tag{2}$$

The configuration space of a rigid robot translating in the plane $\mathcal{W} = \mathbb{R}^2$ is two-dimensional, easily visualized in **Figure 1**. The circular-shape robot is presented with an obstacle in the workspace. When sliding the robot around the obstacle, the boundary configuration can be determined. As a result, motion planning for the robot in the workspace is converted to motion planning for a point robot in the configuration space [7].

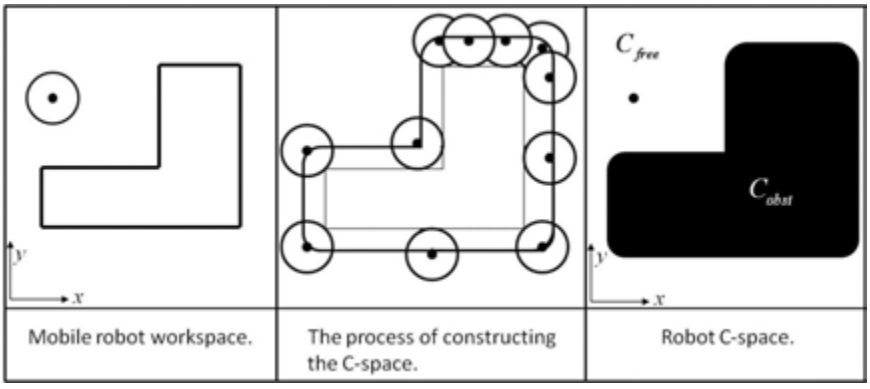

Figure 1. Construction of the configuration space.

2.2. Definition of obstacle avoidance

Let q_{target} be a target configuration. At time t_i the robot is in configuration $q(t_i)$. The robot senses a portion of the environment using its onboard sensors. Let the set of workspace obstacles seen

at configuration $q(t_i)$ be $\mathcal{O}\left(q\left(t_i\right)\right) \subset \mathcal{W}$. The objective is to compute a motion control vector u_i such that

- The robot progresses to the target location $F(q(t_i), q_{\text{target}}) < F(q(t_i + T), q_{\text{target}})$, where $F: \mathcal{C} \times \mathcal{C} \rightarrow \mathbb{R}^+$ is a function that evaluates the progress of one configuration to another [8].
- The trajectory does not collide with the obstacles $\mathcal{A}(Q_{t_i,T}) \bigcap \mathcal{O}(q(t_i)) = \emptyset$, where $Q_{t_i,T}$ is the set of configurations of the trajectory followed from $q(t_i)$ to $q(t_i + T)$. $T > 0$ is the sampling period.

The solution of the problem is a sequence of control vectors $\{u_1, \ldots, u_n\}$ computed in real time that guide the robot eventually to the target configuration while avoiding the sensed obstacles in the environment as shown in **Figure 2**.

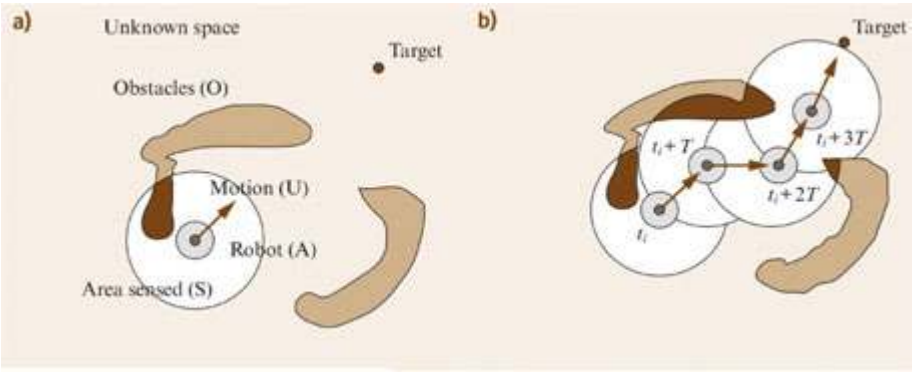

Figure 2. Obstacle avoidance problem [8].

2.3. Kinematics of a two-wheel differential drive robot

A differential drive robot is composed of one passive wheel and two coaxial wheels. The passive wheel provides stability, while the coaxial pair steer the robot through carefully modulating their velocities. A straight line motion is achieved through equal velocities in both wheels, while left and right motion occurs if the right wheel is faster than the left and the left wheel is faster than the right, respectively. Pivoting is noticed when both wheels steer equally as fast, but in opposite directions. A zero turning radius is a major advantage with this motion configuration. An initial rotation can trigger motion in any direction. Further advantages to this robot configuration include the simple mechanical structure and kinematic model and the low fabrication cost. However, this robot configuration has also a few drawbacks: the wheels must be driven with exactly the same velocity profile, which can be challenging considering the actual variations between wheels, motors, and environmental differences. It is also difficult for the robot to move on irregular surfaces. Moreover, the orientation of the robot may change abruptly if one active wheel loses contact with the ground [9].

http://dx.doi.org/10.5772/64227

There are two types of nonholonomic constraints governing the motion of the robot platform: Pure rolling constraint and no lateral slip constraint [10]. The pure rolling constraint implies that the robot wheels have a pure rolling motion without any slipping. This constraint is described by the following equations

$$\dot{x}\cos\theta + \dot{y}\sin\theta + L\dot{\theta} = \omega_r R_w, \tag{3}$$

$$\dot{x}\cos\theta + \dot{y}sin\theta - L\dot{\theta} = \omega_l R_w. \tag{4}$$

The no lateral slip constraint implies that the robot's center point velocity is only in the direction of the axis of symmetry and its lateral component is zero. It is given by

$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0. \tag{5}$$

Without reference to forces and masses, robot kinematics implies a relationship between the position of the robot and its wheels, velocities, and the equations of motion. This section analyzes the mathematical kinematic relationship related to a differentially driven vehicle. The robot configuration is illustrated in **Figure 3**.

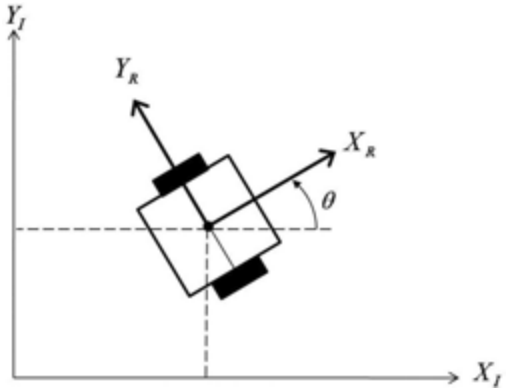

Figure 3. Kinematics of a two-wheel robot.

Let the rotational velocities of the left and right wheel be ω_L and $\omega_{R,}$ respectively, and R_w be the wheel radius then. Then, assuming no wheel slippage, the translational velocities of the wheels are given by

$$v_l = \omega_l R_w, \tag{6}$$

$$v_r = \omega_r R_w. \tag{7}$$

Let the robot forward velocity in the local frame be v, the angular velocity about its Instantaneous Center of Rotation (ICR) axis be ω, and let L be half the distance between the wheels, as shown in **Figure 4**. Then the forward and angular velocities of the robot can be derived from the wheels velocities as follows:

$$\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{-1}{2L} & \frac{1}{2L} \end{bmatrix} \begin{bmatrix} v_l \\ v_r \end{bmatrix}. \tag{8}$$

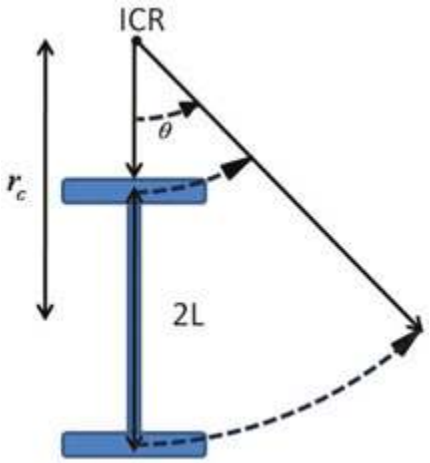

Figure 4. Instantaneous turning radius.

Let θ be the robot orientation with respect to the global x-axis, then the robot velocity vector in the global frame is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}. \tag{9}$$

Figure 4 shows the instantaneous turning radius r_c that can be evaluated by

$$r_c = L\left(\frac{v_r + v_l}{v_r - v_l}\right). \tag{10}$$

http://dx.doi.org/10.5772/64227

2.4. Odometry

A robot's global frame position is measured via the dead reckoning method. This method integrates incremental movements measured through wheel encoders and a compass to estimate position, given a known initial start location. The compass delivers the robot's orientation, θ. The incremental movements are measured through wheel encoders and a compass. The robot orientation is θ, while the angular velocities of the left and right wheels ω_l and ω_r, respectively, estimate the two-dimensional position (x, y) via encoders. Encoders utilize encoder pulses to deliver accurate arrival times through the measurement of angular velocities. For encoder resolution, p, and elapsed time, Δt, the angular velocities of the wheels is then defined as

$$\omega_{r,l} = \frac{2\pi}{p\Delta t}. \tag{11}$$

Next, the robot kinematic Eqs (6–9) are used to find the robot velocities $\dot{x}$ and $\dot{y}$. Let T denote a fixed sampling time. Then, the robot position (x, y) in the global frame is found by performing trapezoidal integration

$$x = x_{\text{old}} + \frac{T}{2}(\dot{x} + \dot{x}_{\text{old}}), \tag{12}$$

$$y = y_{\text{old}} + \frac{T}{2}(\dot{y} + \dot{y}_{\text{old}}). \tag{13}$$

Since the position estimation involves a numerical integration of the measurements, there will be an error accumulation over time. As a result, a meaningful estimate of the position cannot be attained with the dead reckoning method.

Systematic and nonsystematic errors are usually encountered with the dead reckoning method. Systematic errors arise due to the misalignment and the unequal diameter of both wheels, while nonsystematic errors may occur as a result of wheel slippage incidents or nonhomogenous environment with uneven floors.

3. Collision avoidance algorithm

In this section, the proposed collision avoidance algorithm is developed using the following parameters. The configuration $q(t_i)$ denotes the position and orientation of the robot in the global frame, while $\mathcal{P}\left(q\left(t_i\right)\right)$ represents only the sensed portion of the environment. This measured portion of the environment is used to construct the robot's workspace polar map,

which allows the set of obstacles $\mathcal{O}\left(q\left(t_i\right)\right)$ to be identified and the configuration space $\mathcal{C}$ to be computed. Next, the operation of the robot is performed in the C-space to simplify the motion planning and navigation. The motion of the two-dimensional robot in the global frame can be simplified to that of a one point (robot reference point) in the C-space. The optimum steering angle γ_{desired} is selected by identifying all the workspace obstacles and classifying the available gaps that can be accessed by the robot. The nonholonomic constraints are taken into account by computing the required radius of curvature r_c such that $Q_{t_i,T}$ which is the set of configurations of the trajectory followed from $q(t_i)$ to $q(t_i + T)$ does not intersect with any obstacle, $\mathcal{A}(Q_{t_i,T}) \bigcap \mathcal{O}(q(t_i)) = \emptyset$. This is achieved by restricting the radius of curvature to an adaptive upper bound. Finally, the robot executes the control action $u_i = (\gamma_{\text{desired}}, r_c)$. The process is repeated until the robot converges to the target position q_{target}. The algorithm is pictorially illustrated in **Figure 5**.

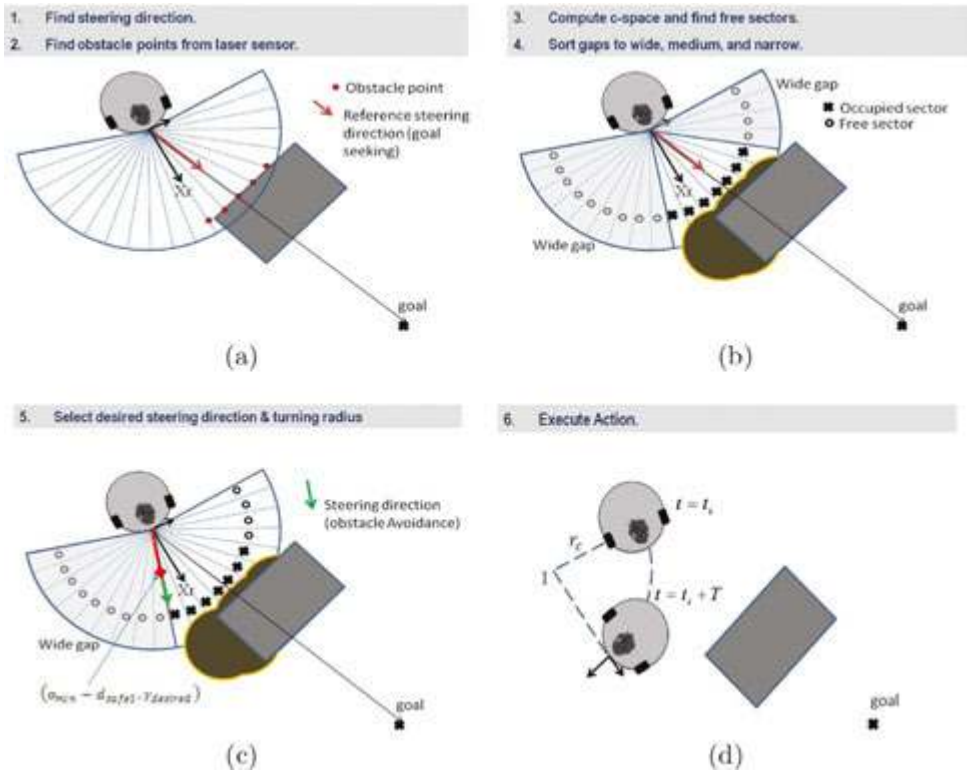

Figure 5. Reactive navigation algorithm in action.

http://dx.doi.org/10.5772/64227

3.1. Identification of reference steering angle

The steering angle with which the robot takes in the absence of obstacles is referred to as the reference steering angle, γ_{ref}. It is an intermediate variable that will later help us find the desired steering angle γ_{desired}, which is derived as follows. Let the robot configuration shown in **Figure 6** be:

$$q_r = (x_r, y_r, \theta_r), \tag{14}$$

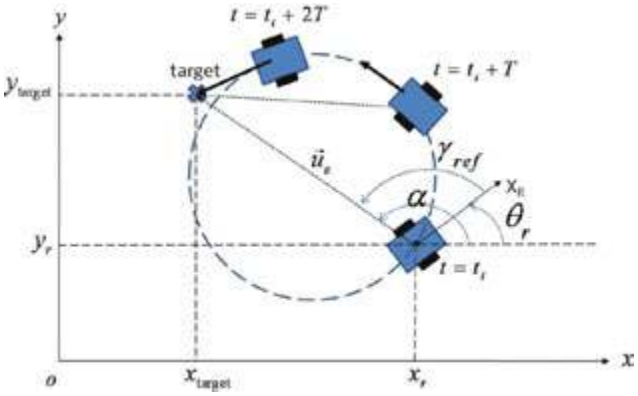

Figure 6. The robot's trajectory to q_{target} in the absence of obstacles.

where (x_r, y_r) is the position of the robot in the $x - y$ plane, and $\theta_r \in [0, 2\pi)$ is the robot's orientation with respect to the x-axis.

Let the target configuration be

$$q_{\text{target}} = (x_{\text{target}}, y_{\text{target}}, \theta_{\text{target}}). \tag{15}$$

Let $\vec{u}_e$ be the vector connecting the robot reference point to the target location. The phase angle of $\vec{u}_e$ is given by

$$\alpha = \arctan \frac{y_{\text{target}} - y_r}{x_{\text{target}} - x_r}. \tag{16}$$

Orientation error is corrected for by turning the robot in an angle defined as follows: $\gamma_{ref} = \alpha - \theta_r$, $-\pi \le \gamma_{ref} \le \pi$. The range of γ_{ref} is chosen in such a way that the smaller turning angle is selected. The robot can turn clockwise (right) or counter clockwise (left). **Figure 6** illustrates

the trajectory taken by the robot to move to the target configuration. The robot finally achieves a straight line path once its x-axis is on line with the error vector.

3.2. Model of robot environment

The robot environment is modeled by constructing a polar map of the workspace in the local robot frame. At a given instant of time, the distances from the robot to all the surrounding obstacles are measured by a laser range finder and used to build the partial polar map. The laser range sensor is calibrated to scan the 200° front view of the robot in 20 sectors with a 10° angular resolution, as illustrated in **Figure 7**. The measured data is returned as a set of data points:

$$\mathcal{P}(q(t_i)) = \{p_1, p_2, ..., p_j, ..., p_{20}\}. \tag{17}$$

A point p_j is expressed by a pair (d_j, β_j) where d_j is the distance between the robot and the obstacle at sector j. β_j is the orientation of the j^{th} sector, S_j, with respect to the local x-axis. The subset of workspace obstacles seen at configuration $q(t_i)$ is identified by applying a threshold on d_j,

$$\mathcal{O}(q(t_i)) = \{p_j \in \mathcal{P}(q(t_i)) \mid d_j \leq R_{safe}\}. \tag{18}$$

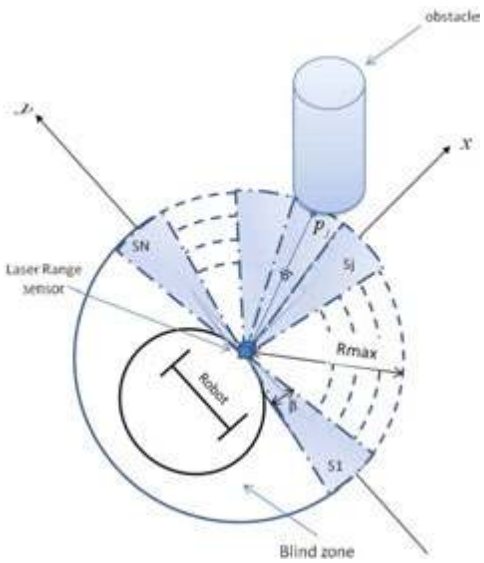

Figure 7. Polar map of the workspace.

http://dx.doi.org/10.5772/64227

The choice of the threshold R_{safe} plays an important role in the obstacle avoidance algorithm. If R_{safe} is large, then the obstacle avoidance will start too soon which results in a suboptimal path. Also, by selecting a large R_{safe}, the algorithm may fail to detect any gaps in the environment and, therefore, incorrectly report a trap situation. For example, the robot in **Figure 8** successfully detects a gap in the environment with R_{safe1} but fails to do so when using a large value R_{safe2}.

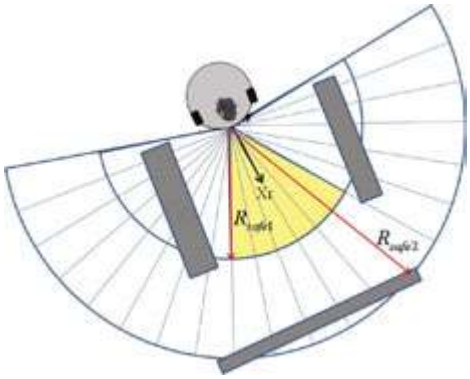

Figure 8. The effect of using a large value for R_{safe}.

The detection range threshold R_{safe} is allowed to take different values depending on the situation encountered:

$$R_{safe} = \begin{cases} 0.1\,m & \text{if robot is close to target configuration;} \\ 0.5\,m & \text{otherwise.} \end{cases} \tag{19}$$

The robot is considered close to the target configuration if:

$$(x_{target} - x_r)^2 + (y_{target} - y_r)^2 \leq \varepsilon \tag{20}$$

where ε is the target threshold.

3.3. Evaluation of configuration space

The 2D robot with radius R computes the C_{obst} for a given set of workspace obstacles, $\mathcal{O}$. Assume for a moment that a single obstacle exists, $\mathcal{O} = \{p_j\}$. As illustrated in **Figure 9**, $\mathcal{C}_{obst}$ is found through tracing the robot's configuration as it slides around p_j. Hence, the following relations can be written for Circle C_j enclosing $\mathcal{C}_{obst}$ with Radius R and center $I_j = (I_{j,x}, I_{j,y})$:

$$\mathcal{C}_{\text{obst}} = \{q \in \mathcal{C} \mid (x - I_{j,x})^2 + (y - I_{j,y})^2 \le R^2\}, \tag{21}$$

$$I_{j,x} = R + d_j \cos\beta_j, \tag{22}$$

$$I_{j,y} = d_j \sin\beta_j, -100^\circ \le \beta_j \le 100^\circ. \tag{23}$$

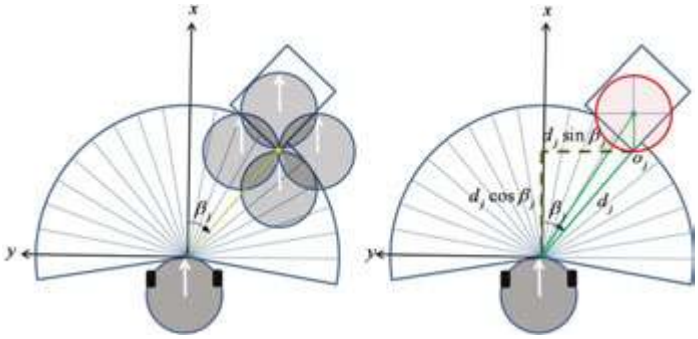

Figure 9. C-space algorithm.

Next, we find the radial distance L_i which is the radial distance between the robot and the boundary of $\mathcal{C}_j$ at angle β_i. The equation of C_j in polar coordinates is

$$\rho_j = \sqrt{I_{j,x}^2 + I_{j,y}^2}, \qquad \phi_j = atan2(\frac{I_{j,y}}{I_{j,x}}), \tag{24}$$

$$L_i^2 + \rho_j^2 - 2\rho_j L_i \cos(\beta_i - \phi_j) = R^2. \tag{25}$$

Equation 24 can be solved for L_i, giving:

$$L_i = \min\{\rho_j \cos(\beta_i - \phi_j) \pm \sqrt{R^2 - \rho_j^2 \sin^2(\beta_i - \phi_j)}\}, \qquad \alpha_{\min} \le \beta_i \le \alpha_{\max}. \tag{26}$$

Equation 26 has a real value if $\alpha_{\min}$ and $\alpha_{\max}$ are selected as:

$$\alpha_{\min} = \min\{\phi_j \pm \sin\frac{R}{\rho_j}\}, \qquad \alpha_{\max} = \max\{\phi_j \pm \sin\frac{R}{\rho_j}\} \tag{27}$$

http://dx.doi.org/10.5772/64227

The above analysis is for the case when $\mathcal{O}$ contains a single obstacle point. In the common case where $\mathcal{O}$ consists of m obstacle points, C_{obst} is found by:

$$\mathcal{C}_{obst} = \bigcup_{1 \le j \le m} \mathcal{C}_j. \tag{28}$$

The exact robots radius was utilized to enlarge the obstacle points. However, control errors arise within the algorithm even when using accurate robot dimensions. Therefore, the radius is modified to $R_s = R + d_{\text{safe}}$. This serves as a space buffer that adds a safety margin. In our implementation d_{safe} is chosen to be 20% of the robot radius.

3.4. Selection of desired steering angle

The sectors in $\mathcal{C}$ are classified as free or occupied. The j^{th} sector S_j is occupied if $L_j \le R_{\text{safe}}$; otherwise, it is free. Adjacent free sectors are grouped together to form gaps. Let N_{free} denote the number of sectors forming a gap. The gaps are classified as follows:

$$gap = \begin{cases} wide & \text{if } N_{\text{free}} > 3, \\ medium & \text{if } N_{\text{free}} = 3, \\ narrow & \text{if } N_{\text{free}} < 3. \end{cases} \tag{29}$$

The desired steering angle is set as the angle of the gap edge with minimum cost. To ensure the selection of the widest possible gap, the search at the beginning is performed over the free wide gaps. if no solution exists within this category, the algorithm searches for a gap in the medium category. The algorithm searches within the narrow gaps, only if the latter two categories did not contain any solution.

$$\text{Cost}(\beta_j) = c_1(\gamma_{\text{ref}} - \beta_j) + c_2\beta_j. \tag{30}$$

The equation is explained as follows: the term $c_1(\gamma_{\text{ref}} - \beta_j)$ refers to the closeness of the goal location to the desired steering direction. The second term, $c_2\beta_j$, indicates how close the current robot heading is to the current steering direction. The coefficients are chose to be $c_1 = 0.3$ and $c_2 = 0.7$, as with this, more weight is given to the steering angles resulting in a smoother trajectory.

In **Figure 10**, the oscillatory trajectory of the robot is exemplified. At the initial start time t_0, the robot's polar map has gaps G_1 and G_2. At this time, the robots steers toward G_2 as it closer to the q_{target}. After an elapsed time T, the robot no longer has G_2 within its range as it achieves a better view, steering the robot toward G_1. However, this action brings the robots back to its initial state at t_0 where both gaps are visible. With this repetitive motion, the robot gets trapped in an infinite loop of repetitive actions. One way to solve this problem is to adjust the steering

angle adaptively and smooth the trajectory while avoiding the trap situations as described later in the chapter.

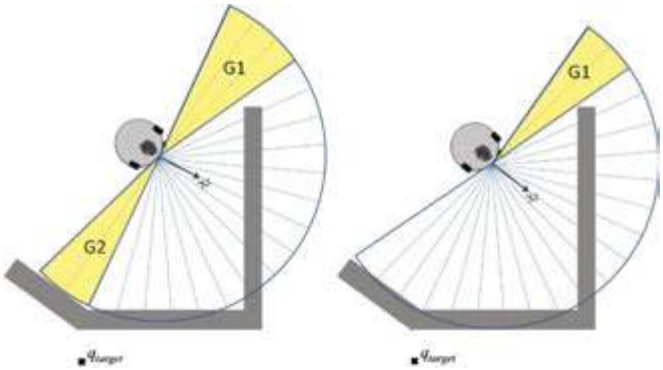

Figure 10. Trajectory oscillation scenario due to a trapped situation.

The algorithm used to select the desired steering angle is summarized as follows:

if $\gamma_{\text{ref}} \in \mathcal{G}$ **then**

$$\gamma_{\text{desired}} = \gamma_{\text{ref}}$$

elseif $\mathcal{G}_{\text{wide}} \neq \phi$ **then**

$$\gamma_{\text{desired}} = \arg\min_{\beta_j \in \mathcal{G}_{\text{wide}}} Cost(\beta_j).$$

else if $\mathcal{G}_{\text{medium}} \neq \phi$ **then**

$$\gamma_{\text{desired}} = \arg\min_{\beta_j \in \mathcal{G}_{\text{medium}}} Cost(\beta_j).$$

else if $\mathcal{G}_{\text{narrow}} \neq \phi$ **then**

$$\gamma_{\text{desired}} = \arg\min_{\beta_j \in \mathcal{G}_{\text{narrow}}} Cost(\beta_j).$$

http://dx.doi.org/10.5772/64227

else

Turn 180° around.

end if

end if

3.5. Identification of adaptive radius of curvature

The robot follows a circular arc with constant wheel velocities instead of the desired steering angle due to the nonholonomic kinematic constraint. A collision could take place when going from the initial to the final path configuration as it may be interested with $\mathcal{C}_{obst}\left(q\left(t_i\right)\right)$. An example of a collision is demonstrated in **Figure 11**, where the robot had steered with a relatively large radius.

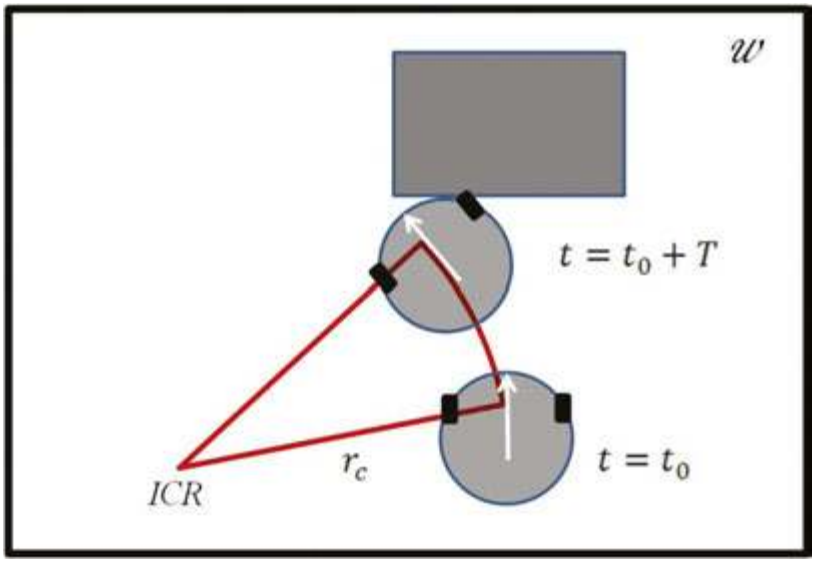

Figure 11. The robot collides with an obstacle because it uses a large turning radius.

The sector along the local x-axis is labeled as S_0 and the sector along the desired steering angle is labeled as S_{desired}. Next, define L_j^m as the distance between the robot front reference point and the obstacle point o_j as shown in **Figure 12**. This distance is evaluated in terms of the variable L_j as follows:

$$L_j^m = \sqrt{(L_j \cos\beta_j + a)^2 + (L_j \sin\beta_j)^2}, \tag{35}$$

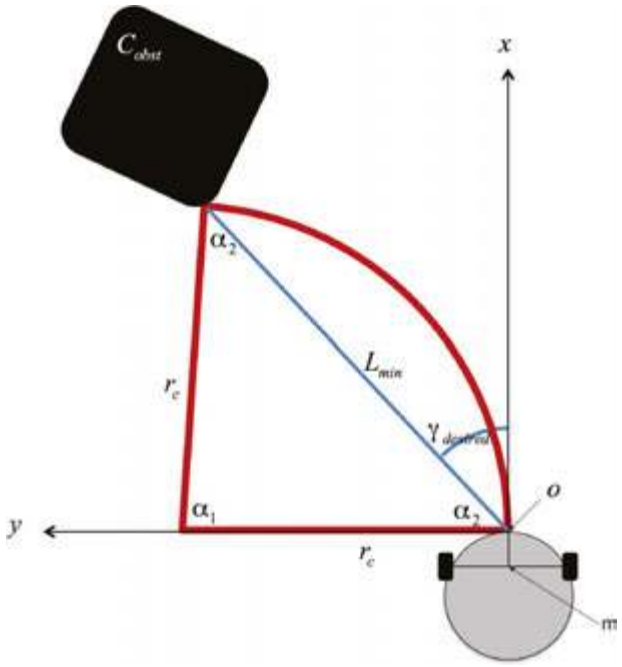

Figure 12. Turning radius selection.

where a is the actual distance separating the middle point m of the wheels axis from the robot front reference point. $L_{\min}$ is then defined as the minimum distance to the nearest obstacle point situated between S_{desired} and S_0. The optimum turning radius r_c is selected so that the robot trajectory goes through ($L_{\min}$, γ_{desired}) as shown in **Figure 12**. The turning radius is derived as follows. Consider the isosceles triangle where the two equal sides have length r_c and the remaining side has length $L_{\min}$. From the law of cosines,

$$L_{\min}^2 = 2r_c^2 - 2r_c^2 \cos\alpha_1. \tag{36}$$

α_1 can be found as

$$\alpha_1 + 2\alpha_2 = 180, \tag{37}$$

$$\alpha_2 = 90 - \gamma_{\text{desired}}, \tag{38}$$

$$\Rightarrow \alpha_1 = 2\gamma_{\text{desired}}. \tag{39}$$

http://dx.doi.org/10.5772/64227

Using the double angle formula and equation 32, we can find r_c as

$$r_c = \frac{L_{\min}}{2\sin(\gamma_{\text{desired}})}. \tag{40}$$

A safety margin is introduced by reducing the turning radius so that the robot passes through the point ($L_{\min} - d_{\text{safe2}}$, γ_{desired}) instead. Also, the turning radius r_c is forced to saturate if it is greater than a threshold value r_{large}. In our implementation, d_{safe2} is selected to be $1.2R$ and $r_{\text{large}} = 0.5\text{m}$.

4. Experimental results

4.1. Mobile robot platform

A prototype robot platform was designed and built to validate the proposed algorithms. The platform has a differential drive mechanism and is designed to operate indoors on flat solid surfaces. Forward, backward, and steered motion is generated by controlling the right and left wheel velocities based on the differential steering concept. The platform control system includes a single board computer and a microcontroller, thus providing a dependable and strong computing environment. The platform comprises also a large range of sensors including ultrasonic sensors and a laser range sensor for obstacle detection, as well as a compass and encoders for localization. **Figure 13** shows a front view picture of the mobile robot platform.

The obstacle avoidance algorithms described earlier are tested on the mobile robot platform in different environment settings. The testing is conducted indoors in a lab environment where the lab furniture is to be avoided. The obstacles are arranged in five different scenarios that vary in difficulty. For all scenarios, the sample time is $T = 1$ s, the robot initial configuration is (0, 0, − 90°) while the target x–y location is (1.6 m, − 1.5 m). Hence, the initial error in position is 2.1932 m.

4.2. Environment setting 1

The robot's initial configuration is connected to the target configuration through a direct path indicated by a straight line as observed in **Figure 14a**. The trajectory, of length 2.2711 m, is depicted in **Figure 14b**. In **Figure 14c**, the robot's velocities are presented and are smooth in the global frame. **Figure 14d** exemplifies the control action. An infinite radius of 0.5 is set for keeping the plot in rage as the robot would steer in an infinite radius when moving in a straight line.

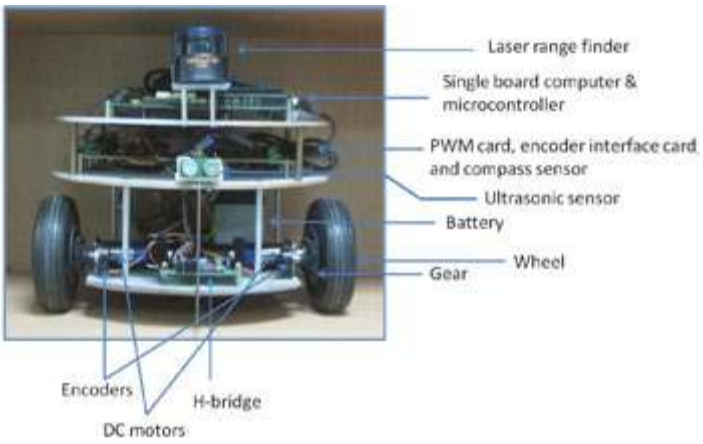

Figure 13. Mobile robot platform.

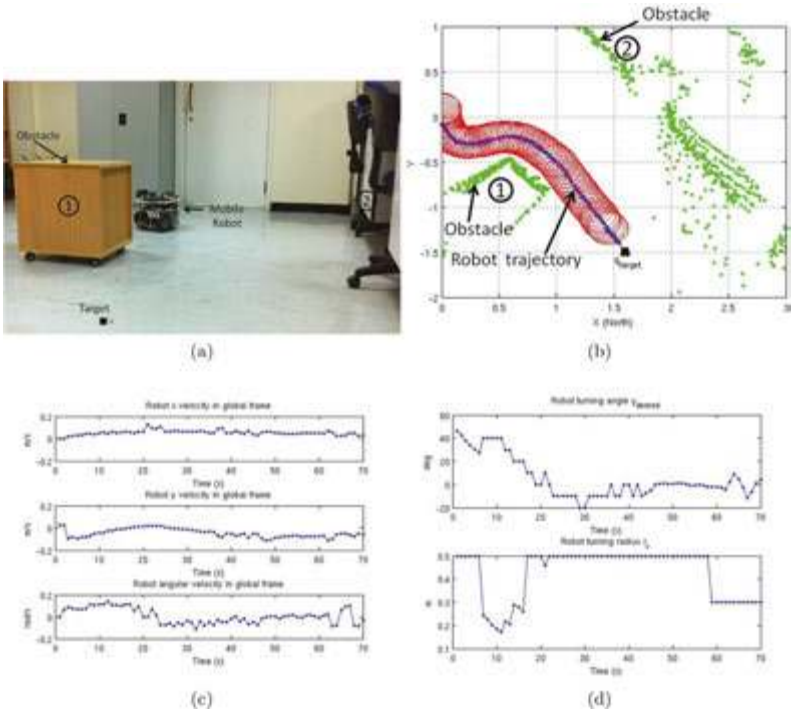

Figure 14. Experimental results. (a) Robot initial position, target position, and the surrounding obstacles; (b) The obstacle points in green, the area occupied by the robot at each instance in time in red, and the reference point trajectory in blue; (c) The robot velocities; (d) The robot control action.

http://dx.doi.org/10.5772/64227

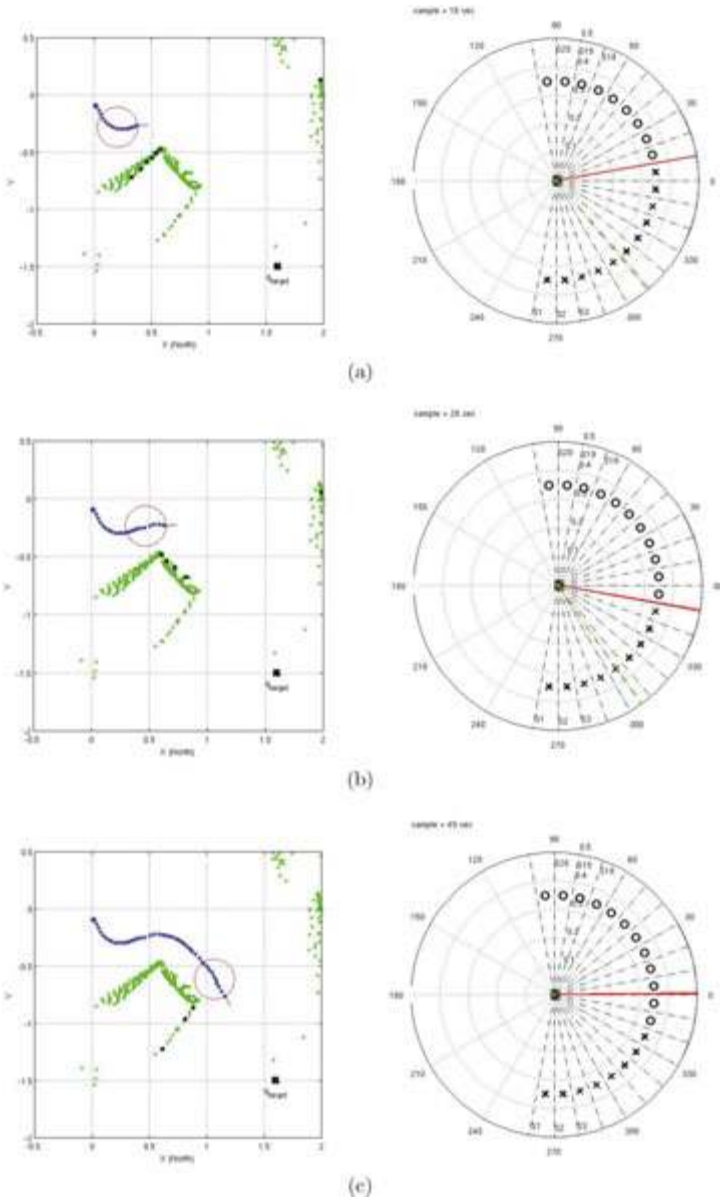

Figure 15. Illustration of the trajectory control algorithm at different time intervals. The obstacles surrounding the robot at a given instant of time are shown as black dots in the Cartesian coordinate frame. A rough estimate of the obstacle contour is defined by the solid yellow line. The classified sectors are shown in the polar histogram. The desired steering angle is indicated by a dashed green line, while the reference steering angle is indicated by a solid red line.

Critical time samples with some intermediate values are shown in **Figure 15**. At a sample of 12 s, the robot is only capable of viewing the square obstacles front side. This classifies the front gaps as occupied. The reference steering angle is classified to be in an occupied sector as it is of −20° value. Hence, the reference angle γ_{desired} with a value 30° is selected as the next best alternative, and the corresponding turning radius is approximately 0.205 m as shown in **Figure 14d**. **Figure 15b** illustrates the robot at a sample time of 26 s. At this sample, the robot can only observe the obstacles' right side and has a γ_{ref} of −54° and a γ_{desired} of −10°. Additionally, the robot takes a turn with a 0.5 m radius. The sample at 49 s is seen in **Figure 15c**. γ_{desired} is simply equated to γ_{ref} as it resides in a free sector. This moves the robot straight to the target. This scenario course was completed within 70 s.

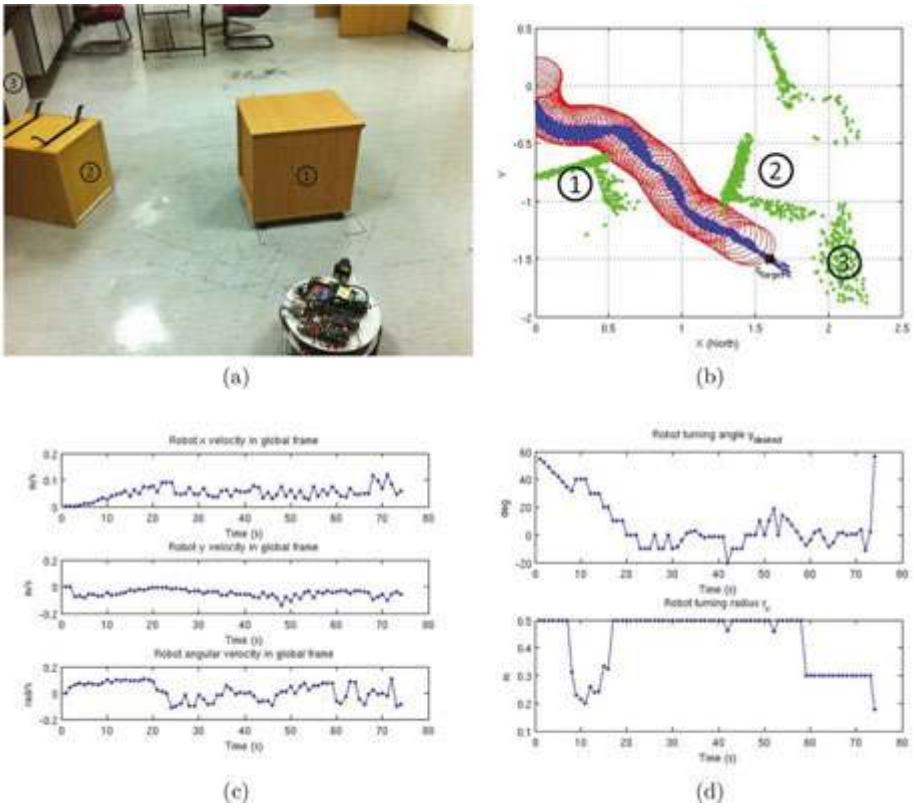

Figure 16. Experiment 2 results. (a) The robot initial position, target position, and the surrounding obstacles; (b) The obstacle points in green, the area occupied by the robot at each instance in time in red, and the reference point trajectory in blue. (c) The robot velocities; (d) The robot control vector.

http://dx.doi.org/10.5772/64227

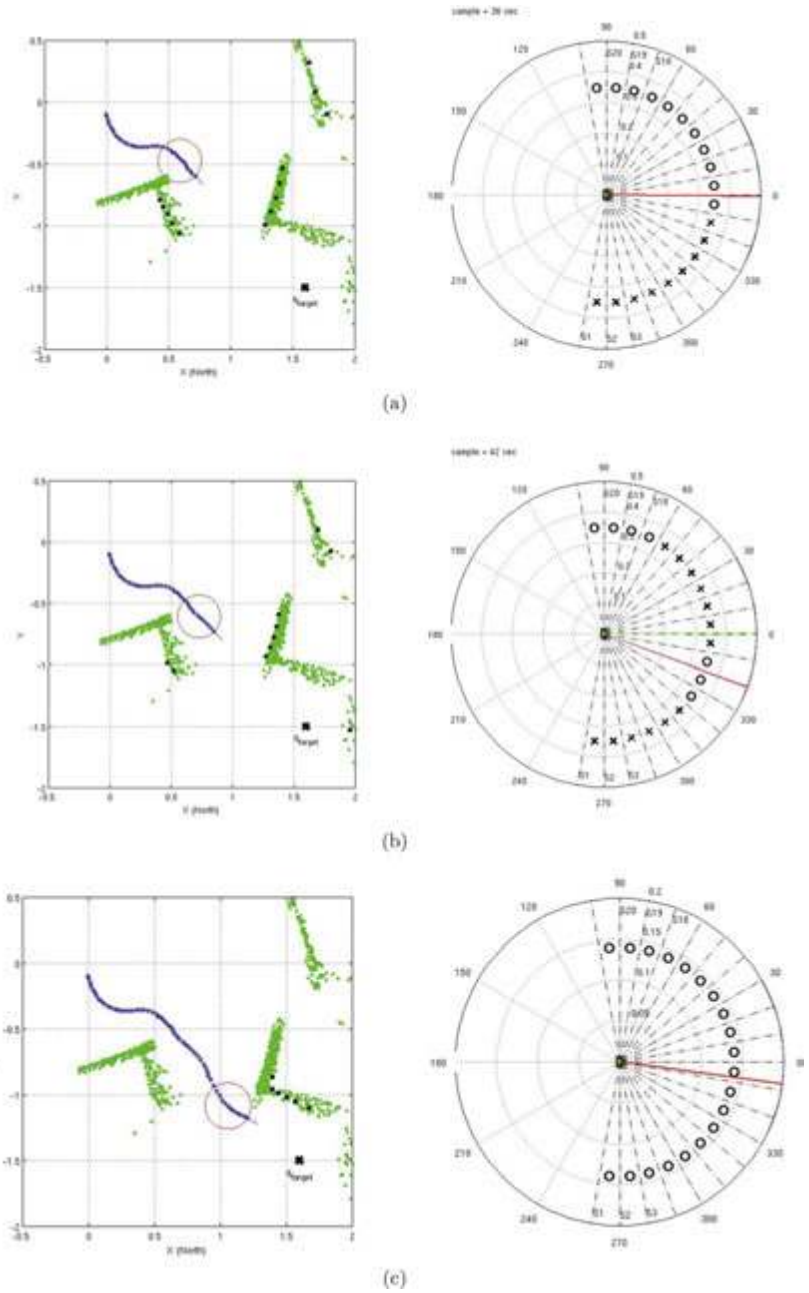

Figure 17. (a) Shows the robot entering the passage; (b) shows the robot inside the passage; (c) shows the robot exiting the passage.

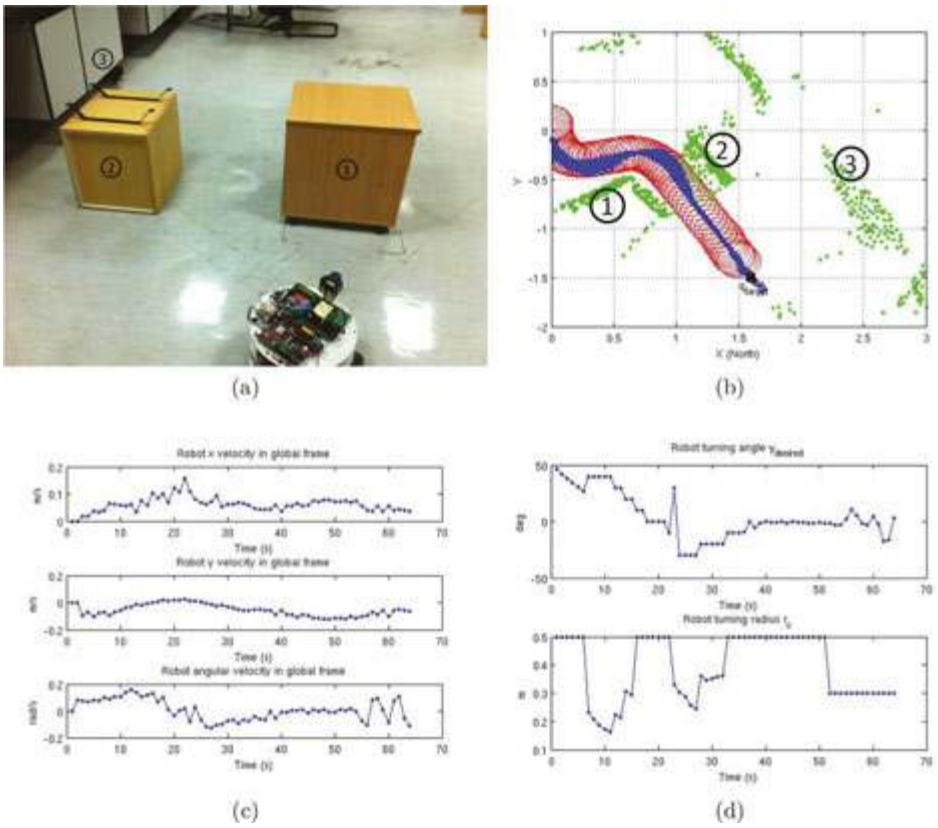

Figure 18. Experiment 3 results. (a) The robot testing environment; (b) the robot trajectory; (c) the robot velocities; (d) the robot control vector.

4.3. Environment setting 2

A scenario of wide entrance but narrow exit was modeled as in **Figure 16a**. The robot was able to make it through the passage within 74 s and with a 2.2539 m trajectory length, shown in **Figure 16b** below. Figue 16c and 16d depict the robot velocities and control actions, respectively. **Figure 17** illustrates with polar histograms for when the robot first enters the passageway, moves within it and then exits.

4.4. Environment setting 3

The difference in this scenario is that the narrow nature of the passage way is greater than that depicted in scenario 2, as shown in **Figure 18a**. As a result, the robots ability to pick up on and detect the narrow gaps that are only slightly larger than its size is tested. As shown in **Figure 18b**, the robot successfully makes it through the pathway and its corresponding

http://dx.doi.org/10.5772/64227

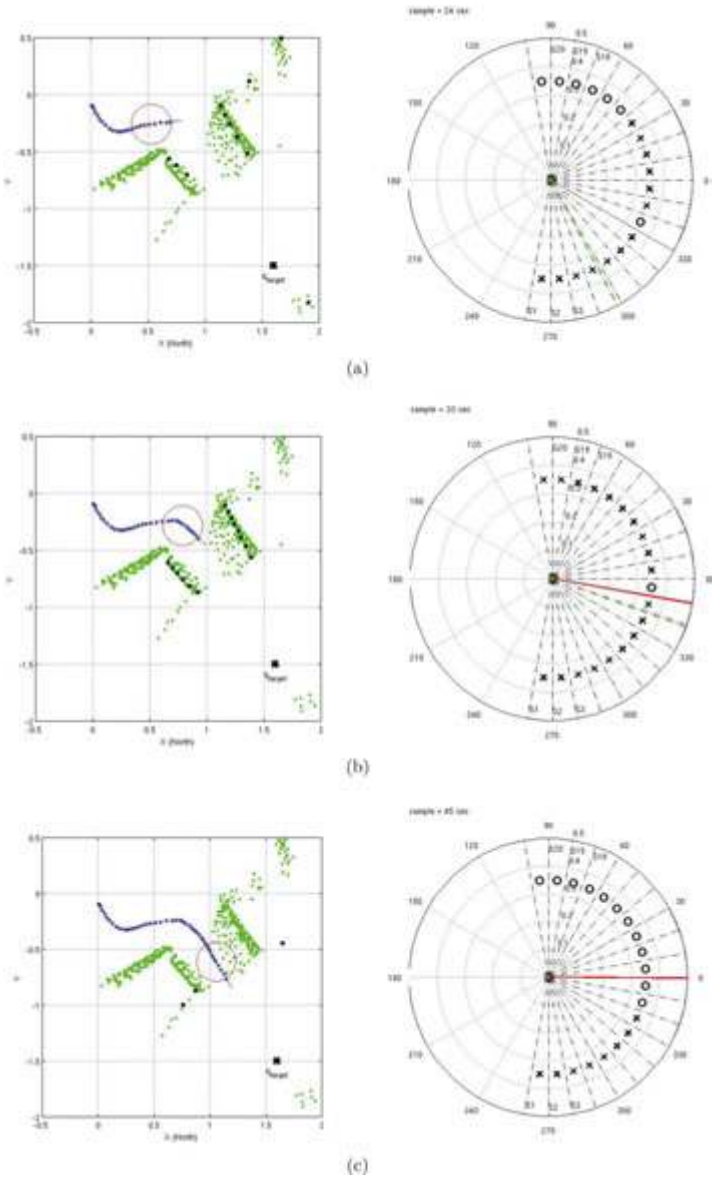

Figure 19. (a) Shows the robot entering the passage; (b) Shows the robot inside the passage; (c) Shows the robot exiting the passage.

velocities and control actions during the trajectory are illustrated in **Figure 18c** and **18d**, respectively. **Figure 19** provides details via histograms from when the robot enters to when it leaves the passageway 19.

4.5. Environment setting 4

The difference between this scenario, depicted in **Figure 20a**, and scenario 3 is the addition of an obstacle to block the exit from the passageway, forming a dead end for the robot. The trajectory taken by the robot is presented in **Figure 20b**, **20c**, and **20d** represent the robot's velocity and control actions through this passageway, respectively. In **Figure 20c**, fluctuations left and right can be seen for the turning angle, γ_{desired}. However, at $t = 31$ s the robot approaches the dead end and comes to realize that the narrow gap is in fact blocked. The robot thus steers left and now envisions the dead end as a gap, resulting in the robots attempt to steer toward it once more. This is illustrated in **Figure 21c**. After 179 s, the robot completes the mission of contouring the obstacles and overcoming the oscillations back and forth having travelled a total of 6.0243 m.

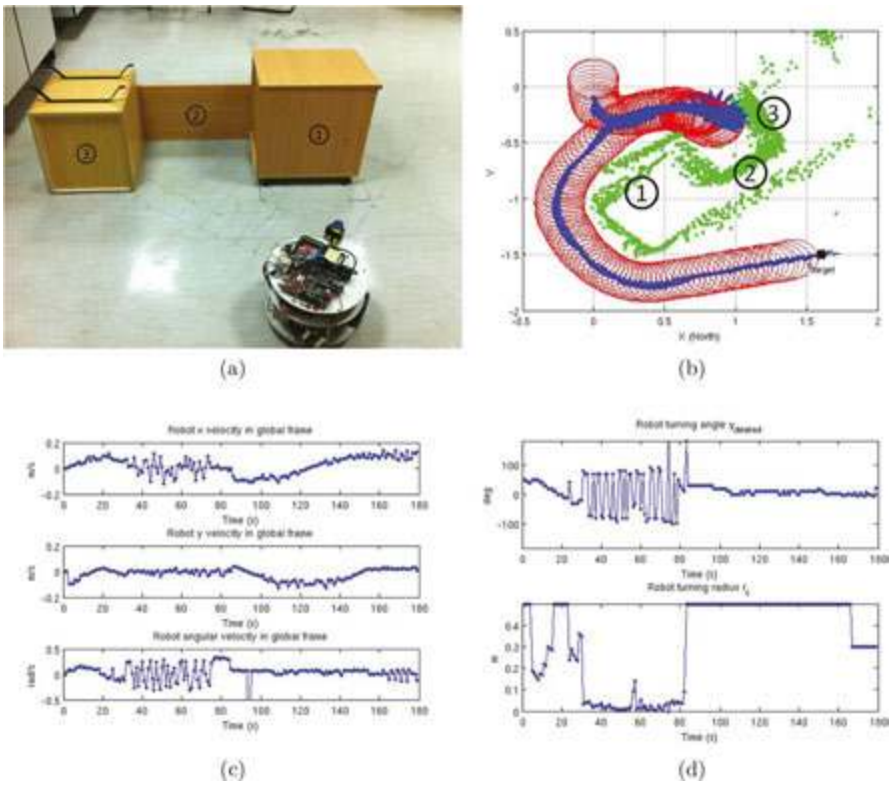

Figure 20. Experiment 4 results (a) The robot initial position, target position, and the surrounding obstacles; (b) The obstacle points in green, the area occupied by the robot at each instance in time in red, and the reference point trajectory in blue; (c) The robot velocities; (d) The robot control vector.

http://dx.doi.org/10.5772/64227

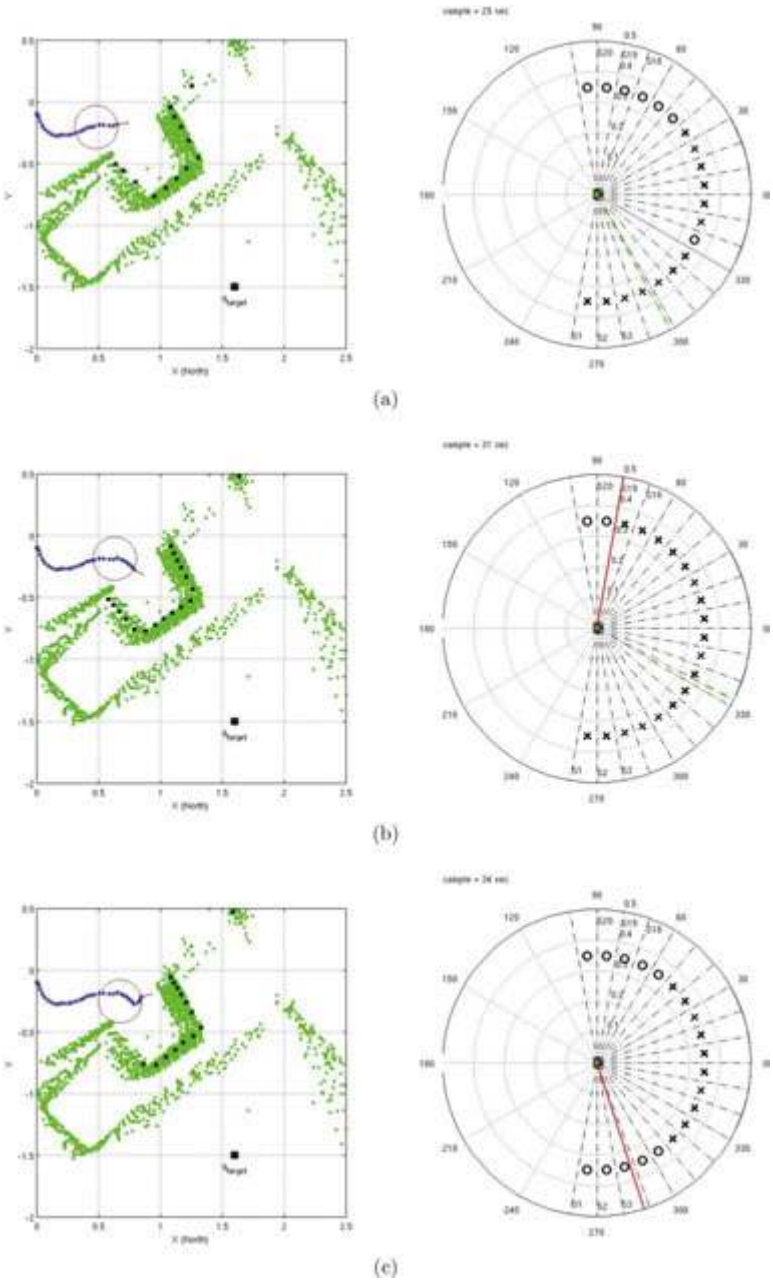

Figure 21. (a) Robot detects a passage; (b) robot discovers a dead end and attempt to turn away; (c) after the robot moves away, the dead end appears as a gap.

4.6. Environment setting 5

The three obstacles in scenario 5 are placed in such a way to form a narrow gap that is smaller than the robot size. This makes the robot's initial steering to fall into a blocked path as depicted in **Figure 22a**. The robot ends up at the target location due to its nature to persistently search for a gap, as illustrated in **Figure 22b**, the target location as shown in **Figure 22b**. The robot velocities and control actions are depicted in **Figure 22c** and **Figure 22d**, once more, depict the robot's respective velocities and control actions.

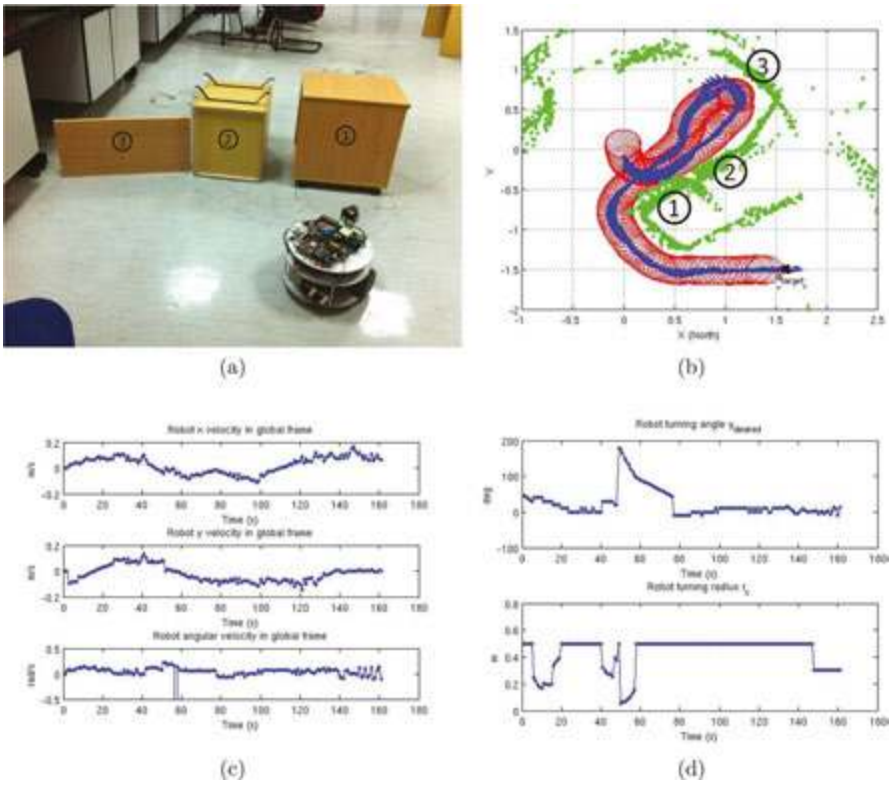

Figure 22. Experiment 5 results. (a) The robot testing environment; (b) The robot trajectory; (c) The robot velocities; (d) The robot control vector.

5. Conclusion

This chapter presents a reactive navigation algorithm for a wheeled mobile robot under nonholonomic constraints and in unknown environments. The mobile robot can travel safely and efficiently to a preset destination having no prior knowledge of the environment. The

shape and dimensions of the robot are all incorporated to produce the control algorithm that determines the set of all steering angles that result in no collisions. The selection of the steering angle depends on the one that is closest to the target and is identified as the widest gap. In addition, the algorithm takes into account the nonholonomic constraints of differentially steered robots by computing circular trajectories with adaptive radius of curvature. A mobile robot platform was built and used to assess and validate the performance of the algorithms over a variety of unstructured indoor environments. The results demonstrate that the navigation algorithm is capable of driving the robot safely through different obstacle arrangements and avoids successfully trap situations.

Author details

Mariam Al-Sagban* and Rached Dhaouadi

*Address all correspondence to: g00006931@aus.edu

American University of Sharjah, Sharjah, the United Arab Emirates

References

[1] M. Al-Sagban, "Autonomous robot navigation based on recurrent neural networks," Master's thesis, American University of Sharjah, 2012.

[2] *Robots attempt record breaking Pacific Ocean voyage*. http://www.bbc.co.uk/news/technology-15790088, 14, March 2012.

[3] *Honda shows smarter robot, helps in nuclear crisis*. http://www.taiwannews.com.tw/etn/news_content.php?id=1753308, 09, Nov 2011.

[4] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Scituate, MA, USA: Bradford Company, 2004.

[5] M. AlSagban and R. Dhaouadi, "Neural-based navigation of a differential-drive mobile robot," in *12th International Conference on Control Automation Robotics & Vision (ICARCV)*, 2012, pp. 353–358.

[6] W.H. Mark Spong and M. Vidyasagar, *Robot Modeling and Control*. New York, USA: Wiley, 2006.

[7] M.V. Mark W. Spong, Seth Hutchinson, *Principles of robot motion: Theory, algorithms, and implementation*. Cambridge, Mass. [u.a.]: MIT Press, 2005.

[8] J. Minguez, F. Lamiraux, and J.-P. Laumond, "Motion planning and obstacle avoidance," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin

Heidelberg, 2008, pp. 827–852, 10.1007/978-3-540-30301-5_36. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30301-5_36

[9] G. Campion and W. Chung, "Wheeled robots," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2008, pp. 391–410, 10.1007/978-3-540-30301-5_18. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30301-5_18

[10] O. Mohareri and R. Dhaouadi, "A neural network based adaptive tracking controller for nonholonomic wheeled mobile robots with unknown dynamics," in *Proc. of the ASME 2010 International Mechanical Engineering Congress & Exposition (IMECE2010)*, vol. 6, November 12–18, 2010.