Mobile robot position control in environment with static obstacles

Konstantinos Giannousakis¹, Anthony Tzes²

¹Electrical & Computer Engineering Department, University of Patras, Rio 26500, Greece
²Electrical & Computer Engineering Program, New York University Abu Dhabi, Abu Dhabi, P.O. Box 129188, United Arab Emirates
E-mail: giannousakis@ece.upatras.gr

Abstract: This study addresses the problem of efficiently navigating a differential drive mobile robot to a target pose in a region with obstacles, without explicitly generating a trajectory. The robot is assumed to be equipped with an omnidirectional range sensor, while the region may or may not be a priori known. Given the known obstacles in each iteration of the controller, the shortest path connecting the robot and the target point provides a raw desired movement direction. Considering the unobstructed area in that direction, the size of the robot and the obstacle contours in its visibility range, the reference direction is determined. Finally, respecting the velocity and acceleration constraints of the robot, the angular velocity is properly selected to rotate the robot towards the reference direction, while the linear velocity is chosen to efficiently minimise the distance to the final target, as well as to avoid collisions. After the robot has reached the target, the controller switches to orientation mode in order to fix the orientation. Experimental studies demonstrate the effectiveness of the algorithm.

1 Introduction

Mobile robots have proven to be a reliable choice for a variety of applications. Besides exploration [1–3], area coverage [4] and surveillance [5], industries are now using mobile robots to automate the management of their warehouse inventory [6, 7]. Whatever the task, the main problem always falls to efficiently move a robot from one location to another, avoiding all the obstacles in its path. This justifies mobile robot navigation to be a field of continuously increasing interest [8–12].

Over the years, various methods have been developed to deal with the problem of autonomous robot navigation. Khatib [13] introduced the artificial potential field method, where the robot movement is based on artificial forces from the target and the obstacles. This method has been very popular due to the simplicity of its core idea [14–16], but has the drawback that it suffers from local minima. Another well-known method is the vector field histogram [17], where the area of interest is represented as a 2D grid of obstacle probability. This method has proven to be efficient in real-time applications as it utilises directly the range sensor data, but is prone to local minima and it does not take the robot dynamics into account. Besides that, it may also fall short in cases of narrow areas, such as doors or corridors. On the other hand, a method that uses the robot dynamics in real-time applications is the dynamic window approach [18]. Here the velocity space of a synchro-drive robot is searched directly for admissible control inputs, ensuring that the robot will be able to stop before hitting an obstacle. This search is limited to the dynamic window of achievable values, considering the acceleration capabilities of the robot and the time interval of the controller. The main drawback of the dynamic window approach is that convergence is not guaranteed as it is susceptible to local minima. Improvements on this method in order to address this drawback have been suggested by other authors, such as the ones in [19, 20].

In this paper, we will focus on the motion planning for a smooth obstacle avoidance, without imposing unnecessary velocity constraints. The presented method assumes a differential drive robot with constrained velocity and acceleration. A partial or whole region is required for the first, path planning step, where the shortest path [21] connecting the robot to the target hints the movement direction. In our experiments we will assume a known region, but the algorithm is intended to work also in exploration cases, where the shortest path is extracted given the explored region. If the path suggests a full turn after encountering a dead end, the motion planning will successfully handle it. The next steps incorporate some core ideas of the above methods and require only the obstacle boundaries in the visible field. First of all, an approach similar to the potential field forces has been taken, in order for the visible obstacles to directly affect the movement direction. Besides that, in a similar manner as in the dynamic window approach, the linear velocity of the robot is constrained to prevent potential collisions.

The movement direction acquired from the shortest path, which is the theoretically optimal, is the base to determine the reference direction for the robot. Initially, the direction of the shortest path is modified to point to the furthest unobstructed location considering the size of the robot, which will be referred to as the effective direction. Subsequently, the obstacles in the visible field are scanned in order to produce the actual reference direction. For each point of the visible obstacles, an admissible direction is determined, pointing away from obstacle and closer to the effective direction. These are then weighted by an importance factor, based on the distance of the respective point from the robot and the correlation of its visibility angle with the effective direction.

Having the reference direction and assuming a differential drive robot, the final step is to determine its velocity inputs. Firstly, we define an estimated collision distance function for the obstacles in the visible space. Then we determine the sign of the linear velocity depending on which direction the distance from target is minimised. Given the minimum collision distance based on that direction, the orientation deviation and the acceleration capabilities of the robot, the linear velocity is selected to ensure no collision and smooth movement, while preserving high values. Finally, the angular velocity endeavours to minimise the orientation deviation, but not surpass it. This is achieved by properly utilising the acceleration capabilities of the robot and the time interval of the controller.

The paper is structured in the following manner. In Section 2, the model of the robot is presented and some key elements are defined. In Section 3, the movement direction towards the target is determined, while in Sections 4 and 5 the actual control inputs for the robot are formulated. Additionally, Section 6 states when and how the orientation mode takes effect. Finally, some experimental
results are presented in Section 7, followed by some concluding remarks in Section 8.

2 Problem formulation

Let us consider a non-holonomic mobile robot with differential drive, as presented in Fig. 1. The kinodynamic equations are given as follows:

\[
\begin{align*}
\dot{x}(t) &= \cos \theta(t) v(t) \\
\dot{y}(t) &= \sin \theta(t) v(t) \\
\dot{\theta}(t) &= \frac{v(t)}{a(m)}
\end{align*}
\] (1)

where \( t \in \mathbb{R}_+ \) is the time instance, \( x(t), y(t), \theta(t) \in \mathbb{R} \) are the coordinates and the orientation of the robot, while \( v(t), w(t) \in \mathbb{R} \) are its linear and angular velocities, respectively. We define the kinematic constraints of the robot in the following manner:

\[
v_{\text{min}} \leq v(t) \leq v_{\text{max}}, \quad |\dot{\theta}(t)| \leq a_{\text{max}},
\]

where \( v_{\text{min}} \in \mathbb{R}_+, v_{\text{max}} \in \mathbb{R}_+ \) and \( a_{\text{max}} \in \mathbb{R}_+ \) are the linear and angular velocity bounds, \( a(t), \alpha(t) \in \mathbb{R} \) are the linear and angular accelerations, while \( a_d, a_\theta \in \mathbb{R}_+ \) and \( a_{\Delta x}, a_{\Delta \theta} \in \mathbb{R}_+ \) are the linear and angular deceleration and acceleration limits. The time interval of the controller is \( T_r \in \mathbb{R}_+ \), and at each iteration \( k \in \mathbb{N} \) we have access to the full measured state vector of the robot \( x(kT_r) = [x_m, y_m, \theta_m, v_m, a_m]^T \). For simplicity, the subscript \( k \) will be omitted from the variable names, because all of them are local to the current time instance. Additionally, concerning the dimensions of the robot, we assume that it has a circular shape of radius \( R \in \mathbb{R}_+ \) around its rotational centre.

Except from the aforementioned constraints, the differential drive robot has an additional constraint between its linear and angular velocities. If \( v_L, v_R \in \mathbb{R} \) are its right and left wheel velocities and \( L \in \mathbb{R}_+ \) the wheel-to-wheel distance, then \( v = (v_L + v_R)/2 \), while \( \omega = (v_R - v_L)/L \). It is obvious that when the linear velocity has its maximum absolute value, then the angular velocity is zero and vice versa. If we define the wheel velocity limit imposed by each motor as \( v_{\text{lim}} \in \mathbb{R}_+ \), we deduce that the mechanical constraints for the linear and angular velocities are, respectively, \( v_{\text{lim}} \) and \( a_{\text{lim}} = 2v_{\text{lim}}/L \). Now given a value \( v \in [-v_{\text{lim}}, v_{\text{lim}}] \) for the linear velocity, the effective mechanical constraint for the angular velocity can be obtained by

\[
\alpha_{\text{eff}} = a_{\text{lim}} \left( 1 - \frac{v}{v_{\text{lim}}} \right).
\]

It is very common for the kinematic constraints to differ from the mechanical, but of course \( |v_{\text{lim}}|, v_{\text{max}} \leq v_{\text{lim}} \) and \( w_{\text{max}} \leq w_{\text{lim}} \) need to be satisfied.

Let \( \Omega \subset \mathbb{R}^2 \) be the region of interest. In most reality cases, a map of the region is incrementally built by a SLAM algorithm [22], using an omnidirectional range sensor of radius \( r_0 > 2R \) equipped on the robot. We define the mapped region as \( \Omega^c \subset \mathbb{R}^2 \), where the entire unexplored area is presumed to be free. In our case, we assume that the problem of localisation and the acquisition of the region are already solved, while in the experiments we assume a known region. The measured position of the robot will be referred as \( p_m = [x_m, y_m]^T \in \Omega^c \), while the localisation error is assumed bounded by a disc of radius \( r_m \in \mathbb{R}_+ \).

Let \( p_i \in \Omega \) with \( \theta_i \in (-\pi, \pi) \) be the target pose for the robot. We define the shortest path from \( p_m \) to \( p_i \) in region \( \Omega^c \), as the ordered set of points \( P(p_m, p_i) = [p_{m}, p_1, \ldots, p_{i}] \), where \( n \geq 1 \), \( p_0 = P(p_m, p_1) = p_m \) and \( p_i = P(p_m, p_i) = p_i \). It is assumed that this path is wide enough to be traversed by the robot. Additionally, we define the visible obstacles with respect to the visibility angle \( \theta_i \in (-\pi, \pi) \) as \( V(\theta) \in \partial \Omega \) and their distance from the robot centre as \( d(\theta) \in \mathbb{R}_+ \cup [\infty] \), where \( d(\cdot) \to \infty \) for obstacles further than \( r_i \).

We define as \( R_c = R + r_m \) the minimum allowed distance from the obstacles and as \( l_e \in (r_m, R) \), the extra safe distance, which the robot will try to preserve when possible.

3 Movement direction

In order for the robot to reach the final target, the shortest path \( P(p_m, p_i) \) is taken as a reference. Specifically, the furthest visible point of the path, \( p_1 \), is chosen as an initial guess for the movement direction. This direction is then modified to the closest unobstructed direction with respect to the robot size, leading to the effective target. Based on that, the visible obstacles are scanned to produce the reference direction.

3.1 Effective target

To determine the effective target we need to search for the effective obstacle direction. Firstly, we have to define the permissible search directions from the right, clockwise, and from the left, counterclockwise, of \( p_i \), which are, respectively

\[
\theta_{cw} = (\theta_i - \pi/2, \theta_i), \quad \theta_{cw} = (\theta_i, \theta_i + \pi/2),
\]

with \( \theta_i = \angle(p_i - p_m) \), as shown in Fig. 2.
to compute the actual enforce a movement direction away from it and closer to the target. Firstly, we define the tangent vector to the visible direction. Specifically, the idea here is that each visible obstacle point tries to these directions will be referred as the effective target is in most cases an admissible location, by the obstacles and their properly weighted combination will result in the reference direction.

There are some prerequisites before getting to the effective direction. Firstly, we define the tangent vector to the visible obstacles, even on the discontinuity locations, as follows:

$$\Delta \theta_b(d) = \begin{cases} \theta - \theta_1 - 2\arcsin \frac{R_s + l_s}{2d(\theta)} & \text{if } \theta \in \Theta_{cw} \\ \theta + 2\arcsin \frac{R_s + l_s}{2d(\theta)} & \text{if } \theta \in \Theta_{cw} \end{cases}$$

which adds an offset $R_s + l_s$ from respective obstacle point. Starting the search from $\theta_e$, the direction of the first effective obstacle point, $\theta_b$ and the effective target, $p_e = p_m + d_e \hat{e}_e$, with $d_e, e_e = 1$ $\Delta \theta_e$, can be found in Algorithm 1 (Fig. 3).

So, the effective target in is most cases an admissible location, reachable from the current position of the robot. Setting $d_e = \min (r_e, \|p_t - p_m\|)$ as an initial value and requiring $d(\theta) \leq d_e$ for the effective obstacles, ensures that, if $p_e$ is visible, thus $p_e = p_m$, it will be selected as the effective target even if it is not really reachable due to the obstacles behind it. This is intended and the robot will get close enough, but will not collide due to the imposed limits on the velocity.

3.2 Reference direction

Having the effective target, the visible obstacle contours are used to compute the actual reference direction of the controller. Specifically, the idea here is that each visible obstacle point tries to enforce a movement direction away from it and closer to the target. These directions will be referred as the effective directions imposed by the obstacles and their properly weighted combination will result in the reference direction.

The selection areas are depicted in Fig. 4 for both cases of $\hat{e}_e$. Also, Fig. 5 shows all these directions in an example, while Fig. 6 demonstrates the selection of the effective direction for various cases.

The importance of each obstacle point is measured by two factors. The first is its distance from the robot. Closer obstacles take priority as their avoidance is more urgent. The second is the
visibility direction of the obstacle. Obstacles at directions closer to \( \theta_e \) take priority, because they are the potential threats, while obstacles in the opposite direction are of no concern. As a result, the combined direction is computed like so

\[
E_e = \int_{-\pi}^{\pi} e_e(\theta) \left( \frac{1}{d(\theta) - R_s} - \frac{1}{l_s} \right) d\theta.
\]

(4)

Except that, the robot should try to maintain a safe distance \( l_s \) when possible. To achieve that we add an extra weight for directions away from the obstacles which are closer than \( l_s \). The visibility angles of these obstacles are as follows:

\[
\Theta_s = \{ \theta \in (-\pi, \pi) : d(\theta) < l_s \}
\]

and the direction we choose for these is

\[
\hat{e}_s(\theta) = \frac{3 \hat{e}_n(\theta) + \hat{e}_b(\theta)}{\sqrt{10}}.
\]

Therefore, the weighed direction for all these points is

\[
E_s = \int_{\Theta_s} \hat{e}_s(\theta) \left( \frac{1}{d(\theta) - R_s} - \frac{1}{l_s} \right) d\theta.
\]

(5)

Finally, as reference direction is chosen \( \theta_r = \angle(E_e + E_s) \). Consequently, the deviation from the current orientation of the robot, the desired rotation angle, is determined as follows:

\[
\Delta \theta_e = \theta_r - \theta_m.
\]

(6)

This value also signifies the sign of the linear velocity, which will follow the sign of \( \cos \Delta \theta_e \).

4 Linear velocity

Now that the reference direction is determined, all that remains is to select the velocity inputs for the robot. We start with the linear velocity, as the limit of the angular will depend on that. The sign of the linear velocity aims to minimise the robot to target distance, while its maximum absolute value is limited based on three things; the collision distance, the target distance and the expected rotation angle after reaching the effective target.

Firstly, we will define the collision distance for the visible obstacles. The collision distance is computed with respect to the
This line can be defined as deceleration in the current direction of the robot, \( \theta_n \), and expresses the proximity of the obstacle points, even if they do not directly obstruct the movement at that direction. To express that, the distance of the obstacle points from the line through \( p_n \) and direction \( \theta_n \) is needed. This line can be defined as \( p = p_n + \lambda \hat{\theta}_m \), with \( \hat{\theta}_m = \frac{1}{L} \theta_m \) and \( L \in \mathbb{R} \). Therefore, the distance from an obstacle point \( V(\theta) \) is

\[
l_0(\theta) = \left\| (p - V(\theta)) - ((p - V(\theta)) \cdot \hat{\theta}_m) \hat{\theta}_m \right\|.
\]  

(7)

When \( l_0(\theta) \leq R_s \), the collision distance is selected as the distance that can be traversed by the robot in direction \( \theta_n \) before touching the obstacles. In this case, the distance is

\[
d_c(\theta) = \sqrt{d(\hat{\theta}_m - \hat{\theta}(\theta))}, \quad \text{if} \ l_0(\theta) \leq R_s
\]

(8)

as shown in Fig. 7a, with \( l_s(\theta) = \sqrt{R_s - l_0(\theta)} \). In order to extend it for all the visible obstacles, we choose

\[
l(\theta) = \begin{cases} \sqrt{R_s - l_0(\theta)}, & \text{when } l_0(\theta) \leq R_s \\ \frac{v_{\text{max}}}{2a} \sin \left( \frac{\Delta \theta}{2} \right) - R_s, & \text{when } l_0(\theta) < R_s. \end{cases}
\]

(9)

The selection for the case of \( l_0(\theta) < R_s \), which ensures continuity as the robot moves and \( \theta_n \) changes, expresses the proximity of the indirect obstacles in terms of the capabilities of the robot to decelerate, considering the time interval of the controller. Finally, the minimum values from the front and from the rear of the robot are, respectively

\[
d_{c, f} = \min_{\theta \in \Theta_f} d_c(\theta) \quad \text{(10)}
\]

\[
d_{c, r} = \min_{\theta \in \Theta_r} d_c(\theta),
\]

(11)

with \( \Theta_f = [\theta_n - \pi/2, \theta_n + \pi/2] \) and \( \Theta_r = [\theta_n + \pi/2, \theta_n + \pi] \).

Besides the collision avoidance, the robot may need to decelerate before taking a turn around an obstacle or while reaching the target. To express that need for deceleration, we introduce the expected distance, \( d_e \), which incorporates the distance \( d_{c, e} \) along with the expected rotation angle while considering the capabilities of the robot. The expected rotation angle is the supplement of the angle formed by \( p_{c, p} \), \( p_{c, p} \) and the next point of the path. By choosing \( p' = P(s_{l_{c, e}}, p) \), as the next point of the path, we define the expected rotation angle as follows:

\[
\Delta \theta_e = \pi - (p_{c, p} \cdot p). \quad \text{(12)}
\]

There are two requirements that need to be met while selecting the expected distance. Firstly, when \( |\Delta \theta_e| \geq \pi/2 \), it should force a full deceleration in \( d_{c, e} \), hence \( d_e = d_{c, e} \). Secondly, it should express no need for deceleration when \( |\Delta \theta_e| < \pi/2 \). The latter can be achieved by adding enough room for deceleration, thus selecting \( d_e = d_{c, e} + (3/2) |v_{\text{max}}/a_d| \). Lastly, considering the relation between the durations of a full deceleration, \( v_{\text{max}}/a_d \), and for a 90° rotation, \( \sqrt{(\pi/2)(2/a_d)} \), we define the expected distance as follows:

\[
d_e = \begin{cases} + \min(d_{c, f}, d_{c, r}) & \text{when } \cos \Delta \theta \geq 0 \\ - \min(d_{c, f}, d_{c, r}) & \text{when } \cos \Delta \theta < 0. \end{cases}
\]

(13)

Finally, we have to compute the velocity limit so that the robot can be able to decelerate to zero at the aforementioned distances. The minimum signed distance at the movement direction is determined as follows:

\[
d_v = \frac{3v_{\text{max}}}{2a_d}. \quad \text{(14)}
\]

Depending on the values of these distances, the following cases are formulated:

1. Case of \( d_{v, d} \geq 0 \) and \( |d_v| < |d| \) or \( v_{\text{max}} |T_s| \). In this case, the robot needs to decelerate. For a simple deceleration we need to solve the following for \( v_{\text{i}} \):

\[
|d_v| = v_{\text{max}} |T_s| = \frac{a_d}{2} |\Delta t^2 + \frac{3}{2} v_{\text{i}}^2 | \quad \text{with } \Delta t = (|v_{\text{max}}| - v_{\text{i}})/a_d \text{ being the time interval to reach } v_{\text{i}}.
\]

- If \( \exists T_s \text{ and } \Delta t \geq T_s \): Then we select the positive root.
- If \( \exists T_s \text{ and } \Delta t < T_s \): Then the control input will not be able to change on time for a full deceleration. To fix that we introduce a delay by including a part with constant velocity, thus the polynomial to solve becomes

\[
|d_v| = v_{\text{max}} |T_s| = \frac{a_d}{2} |\Delta t^2 + v(T_s - \Delta t) + \frac{3}{2} v_{\text{i}}^2 | \quad \text{of which we chose the largest root.}
\]

- If \( \exists T_s \): Then the robot needs to stop the fastest it can, so we select

\[
v_{\text{i}} = 0.
\]
ii. Case of \( dd_x \geq 0 \) and \( |d| \geq |d_x| + |v_m|T_s \): In this case, the robot has more room to move and can, possibly, even accelerate. To obtain the limiting value, we solve the following polynomial:

\[
|d| = |v_m|\Delta_t + \frac{a_d}{2} \Delta t^2 + \frac{3}{2} \frac{\sqrt{v}}{a_d},
\]

with \( \Delta t = (v_i - v_m)/a_d \) being the time interval to reach \( v_i \).

- If \( \Delta t \geq T_s \): Then we choose the largest root.
- If \( \Delta t < T_s \): Then we add again the constant velocity delay and solve for the largest root:

\[
|d| = |v_m|\Delta_t + \frac{a_d}{2} \Delta t^2 + v(T_s - \Delta t) + \frac{3}{2} \frac{\sqrt{v}}{a_d}.
\]

iii. Case of \( dd_x < 0 \): Here, the linear velocity has to change sign. Based on that, we need to take into account a full deceleration from the current velocity, an acceleration to the to-be-computed velocity limit and then a deceleration again. This can be expressed as follows:

\[
|d| = -\frac{3}{2} \frac{\sqrt{v}}{a_d} + \frac{a_d}{2} \Delta t^2 + \frac{3}{2} \frac{\sqrt{v}}{a_d},
\]

where \( \Delta t = v_i/a_d \) and the time interval until \( v_i \) is \( \Delta t = (v_i/v_m) + (v_i/a_d) \).

- If \( \Delta t \geq T_s \): Then we choose the largest root.
- If \( \Delta t < T_s \): Then we choose the smallest root of

\[
|d| = -\frac{3}{2} \frac{\sqrt{v}}{a_d} + \frac{a_d}{2} \Delta t^2 + v(T_s - \Delta t) + \frac{3}{2} \frac{\sqrt{v}}{a_d}.
\]

The above velocity limit ensures deceleration at some distance but does not necessarily respect the constraints. To choose an appropriate value for the velocity, we have

\[
\begin{align*}
v_{l,t} &= \min (v_i, v_m), \\
v_i &= \max (v_{lin}, \min (v_{max}, v_i \cos \Delta \theta_i)).
\end{align*}
\]

Multiplying with \( \cos \Delta \theta_i \) not only sets the sign of the velocity, but it also imposes an extra limit. This ensures that, depending on the desired rotation angle, the choice of the linear velocity does not compromise the requirements for angular velocity.

### 5 Angular velocity

The angular velocity is chosen to minimise the deviation from the reference direction, \( \Delta \theta \). Same way as with the linear velocity, the minimum signed rotation angle due to the current angular velocity is

\[
\Delta \theta = \frac{3}{2} \frac{v_m}{a_d} \sqrt{|v_m|}.
\]

Again, depending on the values of \( \Delta \theta_l \) and \( \Delta \theta_r \), we have the following cases.

i. Case of \( |\Delta \theta_l| \geq 0 \) and \( |\Delta \theta_r| < |\Delta \theta_l| + |v_m|T_s \): In this case, the robot needs to decelerate its rotation. The polynomial to solve for the limiting value of the angular velocity, \( v_\theta \), is as follows:

\[
|\Delta \theta| = |v_m|\Delta_t - \frac{a_d}{2} \Delta t^2 + \frac{3}{2} \frac{\sqrt{\omega}}{a_d},
\]

where \( \Delta t = (|v_m| - v)/a_d \) is the time interval until \( v_\theta \).

ii. Case of \( d\theta \Delta t \geq 0 \) and \( |\Delta \theta_l| \geq |\Delta \theta_r| + |v_m|T_s \): Here, the robot needs to rotate even further than \( \Delta \theta_r \), so it may even be possible to accelerate. In order to obtain the limiting value for the angular velocity, we solve the following polynomial:

\[
\Delta \theta = v_m\Delta t + \frac{a_d}{2} \Delta t^2 + \omega(T_s - \Delta t) + \frac{3}{2} \frac{\omega^2}{a_d},
\]

with \( \Delta t = (v - v_m)/a_d \) being the time interval to reach \( v_\theta \).

- If \( \Delta t \geq T_s \): Then we choose the largest root.
- If \( \Delta t < T_s \): Then we add again the constant velocity delay and solve for its largest root:

\[
\Delta \theta = v_m\Delta t + \frac{a_d}{2} \Delta t^2 + v(T_s - \Delta t) + \frac{3}{2} \frac{\omega}{a_d}.
\]

iii. Case of \( \Delta \theta \Delta t < 0 \): In this case, the angular velocity has to change sign, as depicted in Fig. 8b. Same way with the linear velocity, we take into account a full deceleration from the current velocity, an acceleration and then a deceleration. Thus, we solve

\[
\Delta \theta = v_m\Delta t + \omega(T_s - \Delta t) + \frac{a_d}{2} \Delta t^2 + \frac{3}{2} \frac{\omega}{a_d}.
\]

Finally, we need to take into consideration the hard constraints of the angular velocity. The upper limit comes from (3) based on the robot limits and the selected reference linear velocity.
Moreover, the sign is the same with $\Delta \theta$. Thus, the angular velocity is selected as follows:

$$\omega_t = \text{sgn}(\Delta \theta) \ min(\omega_l, \omega_{max}, \omega_r, \omega_o).$$ (18)

6 Orientation controller

The controller needs to switch to orientation mode when the robot has reached the target. The target is assumed to be reached when $d_{t,e} < 2\pi r$. In that case, we simply set $d_t = 0$ and $\theta_t = \theta$ and follow the same procedure for the selection of the linear and angular velocity.

7 Experimental results

The mobile robot in our experiments is the AmigoBot from ActiveMedia, shown in Fig. 9. The robot can be bounded in a circle of radius $R = 0.17$m around its rotation centre and its wheel-to-wheel distance is $L = 0.279$m. Its mechanical limits for the linear and angular velocities are $v_{lin} = 1 \text{ m/s}$ and $\omega_{lin} = 180 \text{ deg/s}$, respectively, while the mechanical limits for the decelerations and accelerations are $\omega_d = \omega_a = 0.5 \text{ m/s}^2$ for the angular, and $a_{d,lin} = a_{o,lin} = 180 \text{ deg/s}^2$ for the angular. In our case studies, we consider as normal active bounds the $v_{lin} = -0.5 \text{ m/s}$, $v_{max} = 0.5 \text{ m/s}$ and $a_{lin} = 50 \text{ deg/s}^2$ for the velocities, $a_d = a_o = 0.3 \text{ m/s}^2$, $a_d = 100 \text{ deg/s}^2$ and $a_o = 50 \text{ deg/s}^2$ for the decelerations and accelerations.

The localisation is based on a combination of camera and encoder measurements. The augmented reality library ArUco [23] is used in combination with two cameras on the ceiling as the main way of retrieving the position and orientation of the robot in the known region of interest. For completeness and reproducibility, the polygon vertices of our region are given in Table 1. Moreover, the robot is equipped with encoders, used by the robot to calculate its wheel velocities and its relative pose. This relative pose is used to update our localisation whenever the camera system is slow. The localisation error is bounded in a radius $r_u \approx 0.015 \text{ m}$ and the normal sampling time interval in our experiments is $T_s \approx 160 \text{ ms}$. The safe distance that the robot should maintain from the obstacles when possible, is selected with respect to the robot radius as $l_s = R/2$. Furthermore, as our region is known and we are not concerned with mapping it, we assume a large sensing radius $r_r = 2 \text{ m}$.

In our experiments, we present two comparison case studies. The first one compares the algorithm for the normal velocity and acceleration bounds, and for the maximum/mechanical bounds. The second compares the algorithm for the normal sampling time and for a relatively big one. For variety, the initial and final pose are different in each case study.

7.1 Case study on the velocity and acceleration bounds

This case study compares between normal and maximum velocity and acceleration bounds. The initial pose of the robot was $p_{in,0} \approx [1.2, 0.75, 0]$ m, with $\theta_{in,0} = 0^\circ$ and for the final target we chose $p_{f} = [2.0, 1.15, 0]$ m, with $\theta_{f} = 0^\circ$. The trajectory of the robot for the normal bounds case is shown in Fig. 10a and for the maximum bounds case in Fig. 11a, while Figs. 10b and 11b depict the actual motion of the robot. Additionally, Figs. 10c, d and 11c, d depict the linear and angular velocities, respectively, for both cases.

In both cases, the robot successfully reaches the target under a smooth trajectory. The traversed distance is $\sim 9.6 \text{ m}$, with $0.16 \text{ m/s}$ mean linear velocity in a duration of $\sim 49 \text{ s}$ for the first case and $0.2 \text{ m/s}$ mean linear velocity in a duration of $40 \text{ s}$ for the second. The maximum linear velocity was $0.383$ and $0.53 \text{ m/s}$ for the first and second case, respectively. We notice that, even though the maximum linear velocity in the second case is above the limit of the first’s, it is far from its maximum allowed value. This was anticipated and proves that the algorithm effectively adapts the capabilities of the robot with respect to the complexity of the region, eliminating any possibility of collision.

7.2 Case study on the sampling time interval

Here we compare between the normal sampling time interval and a larger one of $1 \text{ s}$. The robot was initially positioned at $p_{in} \approx [0, 0]$ m, with $\theta_{in} \approx -90^\circ$ and the target was chosen as $p_{f} = [1.2, 1.65, 0]$ m, with $\theta_{f} = 90^\circ$. The trajectories of the robot for the two cases are shown in Figs. 12a and 13a, respectively, whereas Figs. 12b and 13b depict the actual motion of the robot.

Table 1

| Region of experiment as a polygon with two parts |
|-----------------------------------------------|
| $x$ | $y$ |
| :--: | :--: |
| 1  | -0.700 | -0.600 |
| 2  | 0.900  | -0.600 |
| 3  | 0.900  | -0.150 |
| 4  | 2.300  | -0.150 |
| 5  | 2.300  | 1.350 |
| 6  | 2.650  | 1.350 |
| 7  | 2.650  | 1.300 |
| 8  | 2.350  | 1.300 |
| 9  | 2.350  | -0.150 |
| 10 | 2.550  | -0.150 |
| 11 | 2.550  | -0.200 |
| 12 | 0.950  | -0.200 |
| 13 | 0.950  | -0.600 |
| 14 | 2.950  | -0.600 |
| 15 | 2.950  | 1.900 |
| 16 | 0.950  | 1.900 |
| 17 | 0.950  | 1.500 |
| 18 | 1.900  | 0.800 |
| 19 | 1.880  | 0.770 |
| 20 | 0.950  | 1.450 |
| 21 | 0.950  | 0.300 |
| 22 | 0.900  | 0.300 |
| 23 | 0.300  | 1.900 |
| 24 | 0.300  | 1.250 |
| 25 | 0.500  | 1.250 |
| 26 | 0.500  | 1.200 |
| 27 | 0.250  | 1.200 |
| 28 | 0.250  | 1.900 |
| 29 | 0.250  | 1.900 |
| 30 | -0.700 | 1.900 |
| 31 | -0.700 | 0.900 |
| 32 | -0.200 | 0.900 |
| 33 | -0.200 | 1.450 |
| 34 | -0.150 | 1.450 |
| 35 | -0.150 | 0.850 |
| 36 | -0.700 | 0.850 |
| 37 | 0.400  | -0.200 |
| 38 | 0.400  | 0.400 |
| 39 | 0.450  | 0.400 |
| 40 | 0.450  | -0.200 |
Fig. 10 Experimental results of Section 7.1 for normal velocity bounds 
(a) Trajectory (—) from initial (●) to target pose (●), (b) Motion of robot at 0.5 fps, (c) Measured (—) and reference (— —) linear velocity, (d) Measured (—) and reference (— —) angular velocity.

Fig. 11 Experimental results of Section 7.1 for maximum velocity bounds 
(a) Trajectory (—) from initial (●) to target pose (●), (b) Motion of robot at 0.5 fps, (c) Measured (—) and reference (— —) linear velocity, (d) Measured (—) and reference (— —) angular velocity.
Fig. 12 Experimental results of Section 7.2 for normal sampling time interval
(a) Trajectory (---) from initial (●) to target pose (●), (b) Motion of robot at 0.5 fps, (c) Measured (---) and reference (— —) linear velocity, (d) Measured (——) and reference (— —) angular velocity

Fig. 13 Experimental results of Section 7.2 for large sampling time interval
(a) Trajectory (---) from initial (●) to target pose (●), (b) Motion of robot at 1 fps, (c) Measured (---) and reference (— —) linear velocity, (d) Measured (——) and reference (— —) angular velocity
Moreover, the linear and angular velocities are depicted in Figs. 12c, d and 13c, d, respectively, for both cases.

In this case, too, the robot smoothly reaches the target. The duration until the target for the normal sampling time was ~24 s, while for the larger sampling time it was 45 s. The larger sampling time enforces more constraints on the movement of the robot, thus leading to smaller velocity values. The other interesting thing to notice is the reverse movement of the robot at the start of the experiment, following the direction that minimises the shortest path.

8 Conclusions
This paper exploits the robot dynamics to efficiently navigate a differential drive robot from one pose to another, in a region with obstacles. The shortest path to the target is used as a reference, in order to ensure that the robot reaches the target in a relatively optimal way. Obstacle avoidance is performed, ensuring the best movement direction while considering the size of the robot. Additionally, the robot dynamics and the sampling time of the controller were utilised, so that the velocities of the robot are maximised while ensuring deceleration before hitting any obstacle. Finally, experimental studies demonstrate the effectiveness of the algorithm.

9 References
[1] Arvanitakis, I., Giannousakis, K., Tzes, A.: ‘Mobile robot navigation in unknown environment based on exploration principles’, 2016 IEEE Conf. on Control Applications (CCA), Buenos Aires, Argentina, 2016, pp. 493–498.
[2] Albina, K., Lee, S.G.: ‘Hybrid stochastic exploration using grey wolf optimizer and coordinated multi-robot exploration algorithms’, IEEE Access, 2019, 7, pp. 14246–14255.
[3] Arvanitakis, I., Tzes, A., Giannousakis, K.: ‘Synergistic exploration and navigation of mobile robots under pose uncertainty in unknown environments’, Int. J. Adv. Robot. Syst., 2018, 15, (1), p. 1729881417750785.
[4] Papatheodorou, S., Tzes, A., Giannousakis, K., et al.: ‘Distributed area coverage control with imprecise robot localization: simulation and experimental studies’, Int. J. Adv. Robot. Syst., 2018, 15, (5), p. 1729881418797494.
[5] Vieira, A.W., Drews, P.L.J., Campos, M.F.M.: ‘Spatial density patterns for efficient change detection in 3D environment for autonomous surveillance robots’, IEEE Trans. Autom. Sci. Eng., 2014, 11, pp. 766–774.
[6] Draganjac, I., Mikić, D., Kočavić, Z., et al.: ‘Decentralized control of multi-agv systems in autonomous warehousing applications’, IEEE Trans. Autom. Sci. Eng., 2016, 13, pp. 1433–1447.
[7] Reinke, C., Beinschob, P.: ‘Strategies for contour-based self-localization in large-scale modern warehouses’. 2013 IEEE 9th Int. Conf. on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, September 2013, pp. 223–227.
[8] Cheng, H., Chen, H., Liu, Y.: ‘Topological indoor localization and navigation for autonomous mobile robot’, IEEE Trans. Autom. Sci. Eng., 2015, 12, pp. 729–738.
[9] Seder, M., Baotić, M., Petrović, I.: ‘Receding horizon control for convergent navigation of a differential drive mobile robot’, IEEE Trans. Control Syst. Technol., 2017, 25, pp. 653–660.
[10] Costa, P.J., Moreira, N., Campos, D., et al.: ‘Localization and navigation of an omnidirectional mobile robot: the robot@factory case study’, IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, 2016, 11, pp. 1–9.
[11] Li, X., Li, X., Khyam, M.O., et al.: ‘Visual navigation method for indoor mobile robot based on extended bow model’, CAAI Trans. Intell. Technol., 2017, 2, (4), pp. 142–147.
[12] Ghosh, S., Panigrahi, P.K., Parhi, D.R.: ‘Analysis of fpa and ha meta-heuristic controllers for optimal path planning of mobile robot in cluttered environment’, IET Sci. Meas. Technol., 2017, 11, (7), pp. 817–828.
[13] Khatib, O.: ‘Real-time obstacle avoidance for manipulators and mobile robots’. Proc. 1985 IEEE Int. Conf. on Robotics and Automation, St. Louis, MO, USA, March 1985, vol. 2, pp. 500–505.
[14] Masoud, S.A., Masoud, A.A.: ‘Constrained motion control using vector potential fields’, IEEE Trans. Syst. Man Cybern. A. Syst. Humans, 2000, 30, pp. 251–272.
[15] Vadakkepat, P., Tan, K.C., Ming-Liang, W.: ‘Evolutionary artificial potential fields and their application in real time robot path planning’. Proc. of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00THS512), La Jolla, CA, USA, July 2000, vol. 1, pp. 256–263.
[16] Orozco-Ronas, U., Picos, K., Montiel, O.: ‘Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots’, IEEE Access, 2019, 7, pp. 156787–156803.
[17] Borenstein, J., Koren, Y.: ‘The vector field histogram-fast obstacle avoidance for mobile robots’, IEEE Trans. Robot. Autom., 1991, 7, pp. 278–288.
[18] Fox, D., Burgard, W., Thrun, S.: ‘The dynamic window approach to collision avoidance’, IEEE Robot. Autom. Mag., 1997, 4, pp. 23–33.
[19] Ogren, P., Leonard, N.E.: ‘A convergent dynamic window approach to obstacle avoidance’, IEEE Trans. Robot., 2005, 21, pp. 188–195.
[20] Brock, O., Khatib, O.: ‘High-speed navigation using the global dynamic window approach’. Proc. 1999 IEEE Int. Conf. on Robotics and Automation (Cat. No.99CH36301), Detroit, MI, USA, May 1999, vol. 1, pp. 341–346.
[21] Guibas, L.J., Hershberger, J.: ‘Optimal shortest path queries in a simple polygon’, J. Comput. Syst. Sci., 1989, 39, (2), pp. 126–152.
[22] Mur-Artal, R., Montiel, J.M.M., Tardós, J.D.: ‘Orb-slam: a versatile and accurate monocular slam system’, IEEE Trans. Robot., 2015, 31, pp. 1147–1163.
[23] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., et al.: ‘Automatic generation and detection of highly reliable fiducial markers under occlusion’, Pattern Recognit., 2014, 47, (6), pp. 2280–2292.