Model Predictive Control for a Mecanum-wheeled Robot Navigating among Obstacles [⋆]

Iñigo Moreno-Caireta [*] **Enric Celaya** [**] **Lluís Ros** [**]

[*] *Facultat de Matemàtiques i Estadística (FME), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain. (e-mail: inigo.moreno@upc.edu).*
[**] *Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain. (e-mails: celaya@iri.upc.edu, ros@iri.upc.edu)*

Abstract: Mecanum-wheeled robots have been thoroughly used to automate tasks in many different applications. However, they are usually controlled by neglecting their dynamics and relying only on their kinematic model. In this paper, we model the behaviour of such robots by taking into account both their equations of motion and the electrodynamic response of their actuators, including dry and viscous friction at their shafts. This allows us to design a model predictive controller aimed to minimise the energy consumed by the robot. The controller also satisfies a number of non-linear inequalities modelling motor voltage limits and obstacle avoidance constraints. The result is an agile controller that can quickly adapt to changes in the environment, while generating fast and energy-efficient manoeuvres towards the goal.

Keywords: Dynamic modelling of wheeled robots, model predictive control, motion control, mobile robot, trajectory and path planning, optimization-based control, obstacle avoidance, energy efficiency, Mecanum wheels, directional sliding wheels.

1. INTRODUCTION

Today, many ground robots and transport systems use traditional wheels to displace themselves (Campion and Chung, 2008). Such wheels are limited to move in a single direction of the plane, as their sliding along the orthogonal direction is prevented by friction. This is beneficial in some cases. For example, in car-like vehicles that go at high speeds it is important to avoid lateral motions for security reasons, and to make the manoeuvrability more intuitive. When moving at lower speeds, however, traditional wheels are less attractive, as they impede holonomic motions in the plane, forcing the vehicle to use multiple manoeuvres even to perform local movements (as when parking a car-like robot in a tight spot). Mecanum wheels solve this problem by mounting rollers on the wheels (Fig. 1, top). With such rollers, the ground force still acts in one direction, but a second direction is available for sliding. By mounting three or more of such wheels, the robot chassis is then free to follow any motion in the plane (Agulló et al., 1987). The wheels can even be fixed to the chassis, which simplifies the robot design substantially. A downside, however, is that mecanum wheeled robots can be inefficient in terms of energy consumption, as the wheels may produce ground opposing forces with a null effect on motion (Fig. 1, bottom). A careful control of the motor torques or voltages is thus needed to minimise this effect.

Due to their freedom of movement and simple design, mecanum-wheeled robots are increasingly used in different contexts, as in intelligent storage facilities, assembly lines of trains or planes, or hospitals, among others (Adăscăliţei and Doroftei, 2011). In such contexts, the robot is required to move to a goal location autonomously without colliding with the environment. The obstacles may be fixed or moving, or have unpredictable behaviour, as when human workers or other robots enter the workspace unexpectedly. The goal of this paper is to develop a motion controller to help fulfilling the previous task. Given a goal state for the robot, specified by a position and a velocity to be attained, the controller computes a sequence of feasible control actions aiming to bring the robot towards such state. Our controller (1) is based on a thorough dynamic model including the equations of motion of the robot and the electrodynamic effects and frictions of the motors, (2) respects motor voltage limits, (3) avoids the collisions with obstacles, and (4) attempts to minimise the energy consumption so as to reduce ground opposing forces to the largest possible extent.

While motion controllers for mecanum-wheeled robots have been given in the past, most of them rely only on the kinematic model of the robot (Indiveri, 2009; Lynch and Park, 2017). That is, they neglect inertia and motor torque limitations in the hope that the commanded velocities will accurately be followed. A kinematic controller may suffice if the robot is light, or if its motors are powerful, but if we have a heavy-weight robot that needs some time to accelerate, or less powerful actuators, a proper control will only be achieved if, as we propose, the full robot dynamics is taken into account. Moreover, since kinematic models only involve position and velocity coordinates,

[⋆] Work partially funded by the Spanish Ministry of Science, Innovation, and Universities under project DPI2017-88282-P.

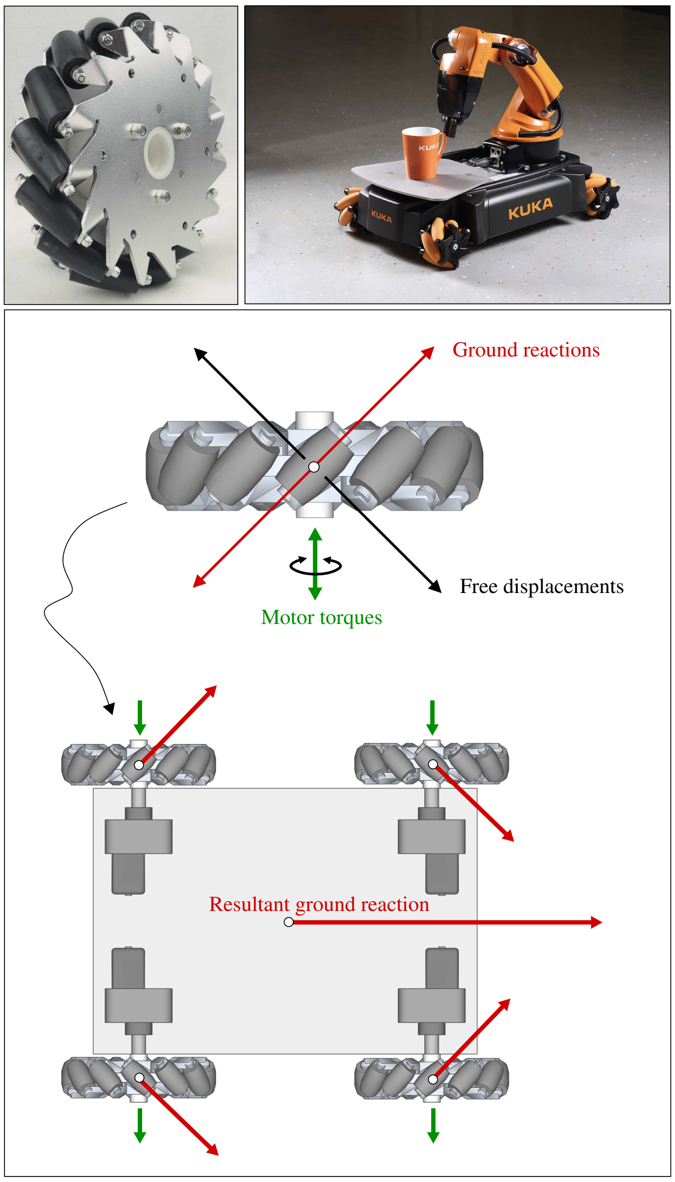


Fig. 1. Top: A mecanum wheel and its application to a mobile manipulation platform (picture courtesy of Kuka AG). Bottom: Possible directions for the motor torques, ground reactions, and free displacements in a wheel when seen from the ground. In a forward move due to the green torques, the four ground reactions in red have opposing lateral components that make the motion inefficient in terms of energy consumption.

and not motor torques or voltages, such models do not allow energy minimisation during the control loop. Also, while in many robots a geometric path is initially planned and later tracked using some control law (Liu et al., 2008; Kuenemund et al., 2017), this approach may be unsuitable if obstacles arise or move unexpectedly, making the planned path invalid at some point. To minimise these situations, we design a model predictive controller that adds flexibility to the system. If sudden changes occur in the environment, or if the robot model has small inaccuracies, or its state suffers a perturbation, the control system will mitigate these issues by recomputing a new action plan to bring the robot to the goal.

The rest of the paper is organised as follows. Section 2 is devoted to obtain the dynamic model of the robot taking into account the nonholonomic constraints existing between the turning speeds of the wheels. The electromechanical behaviour of the motors is also accounted for so that the control signals are given by motor voltages instead of torques, as the latter are not directly controllable in our case. Section 3 then develops a model predictive controller based on this model, introduces the torque and workspace constraints of the robot, and defines the terminal and running cost functions. The performance of the controller is illustrated in Section 4 under various situations including fixed or moving obstacles to be avoided. The paper conclusions and several points for further attention are finally given in Section 5.

2. DYNAMIC MODEL

The dynamic model of a mecanum-wheeled robot was developed by Agulló et al. (1989) using the principle of virtual work. However, Agulló et al. (1989) employ velocity coordinates that are associated with pseudo-coordinates, so their model requires additional manipulations to leave it in a form suitable for control design. To directly obtain a model in the pose coordinates of the chassis, and their time derivatives, we here apply Lagrange's equation with multipliers, which results in less involved derivations. Also in contrast to (Agulló et al., 1989), and to related works like (Wampfler et al., 1989), we extend the model to account for the electrodynamic effects of the motors, including the dry and viscous friction in their axes. In what follows, we shall assume that (1) the robot moves on flat terrain; (2) its wheel rollers have negligible rotational inertia; and (3) the rollers establish non-slipping contacts with the ground.

2.1 Kinematic constraints

We consider the symmetric robot geometry shown in Fig. 2. The vehicle pose is given by the (x, y) coordinates of the vehicle's centre of mass G and the orientation angle ψ of its chassis, both relative to an absolute frame $\{O, X, Y, Z\}$ (not drawn for simplicity). We number the wheels from 1 to 4 and let φ_k denote the rotation angle of the k-th wheel. We also define the chassis configuration by $\boldsymbol{q}_r = [x, y, \psi]^\mathsf{T}$ and the wheels configuration by $\boldsymbol{q}_w = [\varphi_1, \ldots, \varphi_4]^\mathsf{T}$. The robot configuration is then given by $\boldsymbol{q} = [\boldsymbol{q}_r^\mathsf{T}, \boldsymbol{q}_w^\mathsf{T}]^\mathsf{T}$. Since the rollers do not slip, however, the values of $\boldsymbol{q}_r$ and $\boldsymbol{q}_w$ are not independent. They are coupled by the differential constraint

$$\dot{\boldsymbol{q}}_w = \mathbf{R}\mathbf{R}_\psi^\mathsf{T}\dot{\boldsymbol{q}}_r, \tag{1}$$

where

$$\mathbf{R} = \frac{1}{r}\begin{bmatrix} 1 & -1 & -(L+l) \\ 1 & 1 & L+l \\ 1 & 1 & -(L+l) \\ 1 & -1 & L+l \end{bmatrix},$$

and $\mathbf{R}_\psi$ is the 3×3 matrix that encodes a rotation of angle ψ about the Z axis. Equation (1) can be obtained by noting that the velocity of the centre point C_i of each wheel can be computed in terms of $\dot{\boldsymbol{q}}_r$, if C_i is seen as

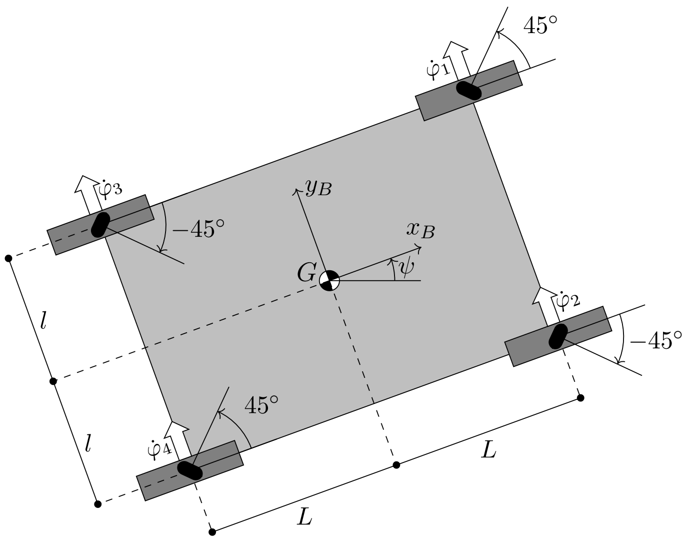


Fig. 2. Geometry of a mecanum-wheeled robot.

a chassis point, or in terms of $\dot{\boldsymbol{q}}_w$, if it is seen as a wheel point, and equating the resulting expressions (Agulló et al., 1987).

2.2 Lagrange's equation with multipliers

Note that equation (1) can be written as

$$\mathbf{C}\,\dot{\boldsymbol{q}} = \mathbf{0}. \tag{2}$$

where $\mathbf{C} = [-\boldsymbol{R}\boldsymbol{R}_\psi^\mathsf{T} \quad \mathbf{I}_4]$ and $\mathbf{I}_4$ is the 4×4 identity matrix. Equation (2) constitutes a nonholonomic constraint on the configuration variables, as it cannot be integrated to depend solely on $\boldsymbol{q}$. On systems subject to such kind of constraints, the dynamic model can be formulated by using Lagrange's equation with multipliers, which has the form

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial T}{\partial \boldsymbol{q}} + \frac{\partial U}{\partial \boldsymbol{q}} + \mathbf{C}^\mathsf{T}\boldsymbol{\lambda} = \boldsymbol{F}. \tag{3}$$

In this equation, $T = T(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $U = U(\boldsymbol{q})$ provide the kinetic and potential energies of the system, $\boldsymbol{F}$ is the generalised force of actuation, and $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers. It is easy to see that, for a mecanum-wheeled robot,

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}m\dot{y}^2 + \frac{1}{2}I_z\dot{\psi}^2 + \sum_{k=1}^{4}\frac{1}{2}I_k\dot{\varphi}_k^2, \tag{4}$$

where m is the total mass of the system, I_z is the moment of inertia of the vehicle around the Z-axis at G assuming each wheel has its mass concentrated at its centre point C_i, and I_k is the moment of inertia of the k-th wheel around its spin axis. Equation (4) can be compactly written as

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2}\,\dot{\boldsymbol{q}}^\mathsf{T}\,\mathbf{M}\,\dot{\boldsymbol{q}} \tag{5}$$

where

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_r & \\ & \mathbf{M}_w \end{bmatrix}, \quad \mathbf{M}_r = \begin{bmatrix} m & & \\ & m & \\ & & I_z \end{bmatrix}, \quad \mathbf{M}_w = \begin{bmatrix} I_1 & & & \\ & I_2 & & \\ & & I_3 & \\ & & & I_4 \end{bmatrix}.$$

Note that, since the robot moves on flat terrain, the term $\partial U/\partial\boldsymbol{q}$ is zero in equation (3). Moreover, for the chosen $\boldsymbol{q}$ coordinates we have

$$\boldsymbol{F} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau} \end{bmatrix}, \tag{6}$$

where $\mathbf{0}$ is a column vector of 3 zeros, $\boldsymbol{\tau} = [\tau_1, \ldots, \tau_4]^\mathsf{T}$, and τ_k is the torque exerted by the k-th motor on the vehicle chassis.

By evaluating the partial derivatives in equation (3) we obtain

$$\mathbf{M}\,\ddot{\boldsymbol{q}} + \mathbf{C}^\mathsf{T}\boldsymbol{\lambda} = \boldsymbol{F}. \tag{7}$$

For a given $\boldsymbol{\tau}$, equation (7) is a system of 7 equations in 11 unknowns (the $7 + 4$ components of $\ddot{\boldsymbol{q}}$ and $\boldsymbol{\lambda}$). To be able to solve for $\ddot{\boldsymbol{q}}$ and $\boldsymbol{\lambda}$, therefore, we have to complement equation (7) with the time derivative of equation (2),

$$\dot{\mathbf{C}}\,\dot{\boldsymbol{q}} + \mathbf{C}\,\ddot{\boldsymbol{q}} = \mathbf{0}. \tag{8}$$

By using equations (7) and (8) simultaneously it would now be possible to write $\ddot{\boldsymbol{q}}$ as a function of $\boldsymbol{\tau}$, which would result in a model of the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\tau})$, with $\boldsymbol{x} = [\boldsymbol{q}^\mathsf{T}, \dot{\boldsymbol{q}}^\mathsf{T}]^\mathsf{T}$. We could now design our controller using such a model, but the optimisation problem to be solved in Section 3 would be of considerable size, as $\boldsymbol{x}$ would involve 14 state variables. On the other hand, since we only wish to control the positions and velocities of the vehicle chassis, a model involving only $\boldsymbol{q}_r$ and $\dot{\boldsymbol{q}}_r$ will be sufficient to our needs. Such a model can be derived as follows.

By employing the block definitions of $\mathbf{M}$, $\mathbf{C}$, and $\boldsymbol{F}$, it can be checked that equations (7) and (8) can be expanded to

$$\mathbf{M}_r\ddot{\boldsymbol{q}}_r - \boldsymbol{R}_\psi\boldsymbol{R}^\mathsf{T}\boldsymbol{\lambda} = \mathbf{0} \tag{9}$$

$$\mathbf{M}_w\ddot{\boldsymbol{q}}_w + \boldsymbol{\lambda} = \boldsymbol{\tau} \tag{10}$$

and

$$-\boldsymbol{R}\dot{\boldsymbol{R}}_\psi^\mathsf{T}\dot{\boldsymbol{q}}_r - \boldsymbol{R}\boldsymbol{R}_\psi^\mathsf{T}\ddot{\boldsymbol{q}}_r + \ddot{\boldsymbol{q}}_w = \mathbf{0} \tag{11}$$

respectively. If we then isolate $\ddot{\boldsymbol{q}}_w$ from equation (11), and substitute the result into equation (10), we get

$$\boldsymbol{\lambda} = \boldsymbol{\tau} - \mathbf{M}_w\boldsymbol{R}\dot{\boldsymbol{R}}_\psi^\mathsf{T}\dot{\boldsymbol{q}}_r - \mathbf{M}_w\boldsymbol{R}\boldsymbol{R}_\psi^\mathsf{T}\ddot{\boldsymbol{q}}_r. \tag{12}$$

Inserting this value for $\boldsymbol{\lambda}$ into equation (9) and rearranging the terms we arrive at

$$\mathbf{H}\,\ddot{\boldsymbol{q}}_r + \mathbf{K}\,\dot{\boldsymbol{q}}_r = \boldsymbol{R}_\psi\boldsymbol{R}^\mathsf{T}\boldsymbol{\tau}, \tag{13}$$

where

$$\mathbf{H} = \mathbf{M}_r + \boldsymbol{R}_\psi\boldsymbol{R}^\mathsf{T}\mathbf{M}_w\boldsymbol{R}\boldsymbol{R}_\psi^\mathsf{T}, \tag{14}$$

$$\mathbf{K} = \boldsymbol{R}_\psi\boldsymbol{R}^\mathsf{T}\mathbf{M}_w\boldsymbol{R}\dot{\boldsymbol{R}}_\psi^\mathsf{T}, \tag{15}$$

or, using the vehicle parameters,

$$\mathbf{H} = \begin{bmatrix} m + \frac{4\,I_w}{r^2} & 0 & 0 \\ 0 & m + \frac{4\,I_w}{r^2} & 0 \\ 0 & 0 & I_z + \frac{4\,I_w\,(L+l)^2}{r^2} \end{bmatrix}, \tag{16}$$

$$\mathbf{K} = \begin{bmatrix} 0 & \frac{4\,I_w\,\dot{\psi}}{r^2} & 0 \\ -\frac{4\,I_w\,\dot{\psi}}{r^2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{17}$$

Using the trivial equation $\dot{\boldsymbol{q}}_r = \dot{\boldsymbol{q}}_r$ in conjunction with equation (13) we now can write

$$\begin{bmatrix} \dot{\boldsymbol{q}}_r \\ \ddot{\boldsymbol{q}}_r \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}}_r \\ \mathbf{H}^{-1}(\boldsymbol{R}_\psi\boldsymbol{R}^\mathsf{T}\boldsymbol{\tau} - \mathbf{K}\dot{\boldsymbol{q}}_r) \end{bmatrix}. \tag{18}$$

Finally, if we redefine the state variable as $\boldsymbol{x} = [\boldsymbol{q}_r^\mathsf{T}, \dot{\boldsymbol{q}}_r^\mathsf{T}]^\mathsf{T}$ and let $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\tau})$ denote the right-hand side of equation (18), we obtain

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\tau}). \tag{19}$$

which gives the model in the first-order form typically assumed in control systems design.

2.3 Accounting for the motor dynamics

Suppose our robot is actuated by DC motors, which is often the case in practice. If such motors admit reference torque signals, and are powerful enough to accurately follow such signals, then the model in equation (19) may be sufficient to design a proper controller. Here, however, we shall assume that the motors are less potent, or that they only admit voltages as inputs, so we will need to transform the model into one of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \tag{20}$$

where $\boldsymbol{u} = [v_1, \dots, v_4]^\mathsf{T}$, being v_k the voltage input to the k-th motor. A clear advantage of this model over a purely kinematic one is that, since equation (20) relates motor voltages to state rates, it allows an easy formulation of a cost term penalising energy consumption. This point will be exploited in Section 3.2 below.

To obtain equation (20) from equation (19), it suffices to show how each torque τ_k can be written as a function of v_k and $\dot{\varphi}_k$. This can be achieved by first writing v_k as a voltage drop along the motor armature; i.e.,

$$v_k = R\, i_k + L\,\frac{di_k}{dt} + K\, N\, \dot{\varphi}_k, \tag{21}$$

where R, i_k, and L are the resistance, current, and inductance of the armature circuit, K is the torque constant of the motor, and N is the gearbox ratio. On the other hand, the net motor torque at the output shaft of the gearbox is given by

$$\tau_k = \eta N K\, i_k - \tau_k^{\text{fric}} - \tau_k^{\text{inert}}, \tag{22}$$

where $\eta \in [0, 1]$ is the gearbox efficiency factor (which models the fact that the gearbox amplifies the motor torque by a bit less than N), and τ_k^{fric} and τ_k^{inert} are the friction and inertia torques of the motor reduced to its output shaft. Since both L and the inertia moment of the motor are often small, the terms $L\,\frac{di_k}{dt}$ and τ_k^{inert} can usually be neglected in the last two equations. If we then isolate i_k from equation (21) and substitute it into equation (22), we see that

$$\tau_k = \frac{\eta N K}{R}\, v_k - \frac{\eta N^2 K^2}{R}\dot{\varphi}_k - \tau_k^{\text{fric}}. \tag{23}$$

All parameters in equation (23) can be obtained from the datasheets of the motor in principle. However, τ_k^{fric} is a function of $\dot{\varphi}_k$ that, very often, has to be identified experimentally. Following common practice (see Kuenemund et al. (2016) for example), we shall assume that this function takes the form

$$\tau_k^{\text{fric}} = a\,\dot{\varphi}_k + b\,\text{sign}\left(\dot{\varphi}_k\right), \tag{24}$$

where the first and second terms model the viscous and Coulomb friction torques at the motor, and a and b are unknown constants a priori. To find a and b, one can apply a certain voltage v_k to the motor and wait for its rotor to reach a constant velocity $\dot{\varphi}_k$, which can be measured using the motor encoder. As $\dot{\varphi}_k$ is constant, there is no acceleration, and thus $\tau_k = 0$ in equation (22). We can then use equation (22) to obtain τ_k^{fric} for the known values v_k and $\dot{\varphi}_k$. Repeating the experiment for multiple values of v_k, we can draw an empiric plot of τ_k^{fric} versus $\dot{\varphi}_k$ and infer a and b using nonlinear regression. By substituting equation (24) into equation (23) we finally obtain τ_k as a function of v_k and $\dot{\varphi}_k$, and hence the model in equation (20).

3. CONTROLLER DESIGN

Suppose now that, at a given iteration of its control loop, our robot is at a current state $\boldsymbol{x}_c$, and we wish to find the action $\boldsymbol{u}_c$ that brings it as close as possible to a goal state $\boldsymbol{x}_g$. A model predictive controller finds $\boldsymbol{u}_c$ by solving a discrete-time optimal control problem (Maciejowski, 2002). To this end, the model in equation (20) is discretised into a state transition equation of the form $\boldsymbol{x}_{i+1} = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i)$ using a fixed time increment Δt. The trajectory to be planned is also discretised into n states $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$, so the future planned horizon is of $n\ \Delta t$ seconds. The optimal control problem can be posed as follows:

> Find the states $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$, and corresponding actions $\boldsymbol{u}_1, \dots, \boldsymbol{u}_n$, that minimise
>
> $$J = T(\boldsymbol{x}_n, \boldsymbol{x}_g) + \sum_{i=1}^{n} R_i(\boldsymbol{x}_i, \boldsymbol{u}_i) \tag{25}$$
>
> subject to the constraints
>
> $$\boldsymbol{x}_1 = \boldsymbol{x}_c, \tag{26}$$
> $$\boldsymbol{x}_{i+1} = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i), \tag{27}$$
> $$\mathbf{I}(\boldsymbol{x}_i, \boldsymbol{u}_i) \geq \mathbf{0}. \tag{28}$$

In this problem, $T(\boldsymbol{x}_n, \boldsymbol{x}_g)$ is a *terminal* cost used to penalise the mismatch between $\boldsymbol{x}_n$ and $\boldsymbol{x}_g$, $R_i(\boldsymbol{x}_i, \boldsymbol{u}_i)$ is a *running* cost that, when added for $i = 1$ to n, accounts for the control effort employed along the trajectory, and $\mathbf{I}(\boldsymbol{x}_i, \boldsymbol{u}_i) \geq \mathbf{0}$ is a system of inequality constraints characterising the set of admissible states and actions for the robot.

Note that, while n actions $\boldsymbol{u}_1, \dots, \boldsymbol{u}_n$ are computed, only the first one is fed to the system, so $\boldsymbol{u}_c = \boldsymbol{u}_1$ (Maciejowski, 2002). The whole problem is then solved again at the next iteration of the control loop. In doing so, the control system can be made very robust, as it is easy to account for eventual changes in the model parameters, or in the state or action constraints, and update them into equations (27) and (28). We next explain how we have formulated equations (25)-(28) in our case.

3.1 Constraints formulation

To discretise equation (20) into equation (27), we have applied the explicit Runge-Kutta method of 4th order. This method provides good integration accuracy while still making $\boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i)$ sufficiently simple to evaluate. The inequalities in equation (28) are in turn defined by the torque constraints and workspace limits of the robot, and by collision avoidance constraints. The torque constraints can be formulated as direct bounds of the form $v_k^{min} \leq v_k \leq v_k^{max}$, where v_k^{min} and v_k^{max} are the minimum and maximum voltages allowed for the kth motor. For the workspace constraints, we assume that our robot has to stay within a rectangular region $\mathcal{W}$ defined by bottom-left and upper-right corner points $(x_{l,l}, y_{l,l})$ and $(x_{u,r}, y_{u,r})$ in the absolute coordinate frame. We then define a rectangular bounding box $\mathcal{B}$ for the robot and force its four vertices to stay within $\mathcal{W}$. Thus, if (w, h) are the coordinates of one such vertex in the body-fixed frame $\{G, X_B, Y_B\}$ of Fig. 2, we add the following equations to equation (28):

$$x_{l,l} \leq x + w\ \cos\psi - h\ \sin\psi \leq x_{u,r} \tag{29}$$
$$y_{l,l} \leq y + w\ \cos\psi + h\ \sin\psi \leq y_{u,r} \tag{30}$$

Precise collision constraints are not easy to enforce. To keep the formulation simple, we assume that each one of the obstacles is enclosed within a bounding ellipse and then force each of the contour segments of $\mathcal{B}$ to not intersect any such ellipse using the inequalities in (Potdar, 2018, Appendix C).

3.2 Terminal and running cost functions

Since the main goal of the system is to move the robot to a certain goal position (x_g, y_g) and orientation ψ_g, the terminal cost function $T(\boldsymbol{x}_n, \boldsymbol{x}_g)$ is simply defined with the weighted squared distance

$$W_{pos}(x_g - x_n)^2 + W_{pos}(y_g - y_n)^2 + W_{ang}(\psi_g - \psi_n)^2, \tag{31}$$

whose weights W_{pos} and W_{ang} are tuned to account for the mixed positional and angular coordinates intervening. The distance is squared in order to ease its automatic differentiation by the solver employed (Section 3.3).

To select energy-efficient trajectories and minimise ground opposing forces as much as possible (recall Fig. 1), our running cost $R_i(\boldsymbol{x}_i, \boldsymbol{u}_i)$ quantifies the energy consumed by the motors. The energy consumed by the k-th motor during a time step Δt is $P_k \Delta t$, where P_k is the motor power during such step. Clearly $P_k = v_k\ i_k$ but since i_k can be written in terms of v_k using equation (21), we have

$$P_k = v_k\ i_k = v_k \frac{v_k - NK\dot{\varphi}_k}{R}, \tag{32}$$

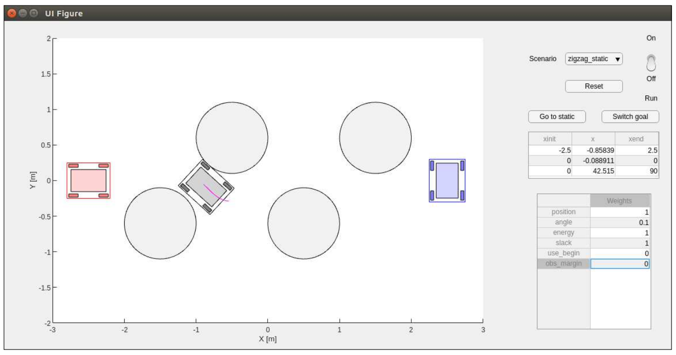
which allows us to easily formulate $R_i(\boldsymbol{x}_i, \boldsymbol{u}_i)$ in terms of the state and action variables employed.

3.3 Numerical optimisation method

Notice that equations (25)-(28) give rise to a nonlinear nonconvex optimisation problem that, due to its general structure, is rather difficult to solve. In practice, thus, the controller is implemented by only finding a locally-optimal solution to the problem. The most popular methods to find such a solution belong to the so-called Newton-type family, which can be subdivided into sequential quadratic programming (SQP) and interior point (IP) methods. The former consist in resolving a sequence of quadratic programming problems obtained from quadratic and linear approximations of J and equations (27)-(28) respectively. While SQP methods are very popular, they are also computationally intensive. This implies that, to obtain $\boldsymbol{u}_c$ in the shortest possible time, only a few iterations of an SQP method are often executed, which leads to suboptimal operation of the system. IP methods, on the other hand, compare favourably with SQP ones, allowing the computation of a locally-optimal action plan in short times. In particular, we have used the recent IP method by Zanelli et al. (2017), which turns to be faster than other state of the art approaches.

4. TEST CASES

To test the performance of the controller, we implemented a robot simulator using the geometric and dynamical parameters of a Mecanum Wheel Vectoring Robot IG42 built



Scenario: zigzag_static

xinit	x	xend
-2.5	-0.85839	2.5
0	-0.088911	0
0	42.515	90

	Weights
position	1
angle	0.1
energy	1
slack	1
use_begin	0
obs_margin	0

Fig. 3. Graphical user interface of the simulator

by SuperDroid Robots Inc., using the motor specifications provided by the supplier and the friction parameters identified on a particular robot [see Moreno-Caireta (2019) for details]. A graphical user interface (Fig. 3) allows the definition of moving obstacles in the simulated workspace, as well as the initial and goal configurations of the robot (shown in red and blue respectively). The weights of the cost function and permitted obstacle clearance margins can be easily set through the interface. The interface is connected to the controller to display the current configuration of the robot (shown in grey) and the obstacles in simulated time. In all test cases the planned trajectory is discretised in intervals of $\Delta t = 0.1$s, and the controller is made to plan for $n = 10$ stages, i.e., the trajectory of the robot is computed for one second ahead in the future. This allows the solver to fulfil the time constraints required to allow real-time execution.

We next examine the performance of the controller in three illustrative situations. The results can be seen in detail in the companion video `https://youtu.be/jr-4Tv9pAtY`. In the video, and in the figures below, the set of robot positions planned for the next $n\Delta t$ seconds is represented as a curvy line in magenta. It can be seen that this line is modified at each stage as the robot moves along.

4.1 Free movement problem

In this test, the robot must go from a start to a goal configuration with the same initial orientation, moving in an obstacle-free environment. Fig. 4 shows the configuration reached by the robot at three different times. It can be observed (and it is clearly seen in the video) that the robot starts moving fast towards the goal, but it slows down when it is near the target position. This is due to the definition of the cost function, which penalises, on the one hand, the energy consumption and, on the other hand, the square of the distance to the goal. This causes that, while the distance is large, shortening it reduces the cost much more than limiting the energy consumption. Conversely, when the distance is short enough, its square becomes tiny and reducing the energy cost is more advantageous. Since our cost function does not involve the time used to reach the goal, the controller minimises energy consumption and takes two more seconds to drive the robot from the configuration shown in the last snapshot of Fig. 4 to the goal.

Interestingly, despite the orientation of the initial and final configurations are the same, the robot changes its orientation along the trajectory. This can be explained as a way to reduce the energy consumption by trying to minimise the strength of the opposing forces which do not contribute to the robot movement but that in most situations are generated by the wheels against the ground. Clearly, when moving in a diagonal direction, two of the wheels do not turn, they simply slide along their free displacement direction (see Fig. 1) and no torque needs to be applied to them. Torque must be applied to the other two wheels and, since the reaction forces of the ground are directed along the axes of the rollers in contact with ground, all reaction forces are aligned with the direction of the robot's movement. Thus, in the case of diagonal movement, all resulting forces are directed along the advance direction, they fully contribute to the robot propulsion and no opposing forces appear.

4.2 Narrow corridor problem

The second test shows the ability of the controller to avoid obstacles. The robot is faced with a challenging environment with four circular obstacles whose separation is less than the longest side of the robot (Fig. 5). The controller is able to find a path by adjusting the orientation of the robot as necessary. Notice that the robot does not keep advancing forwards for the whole trajectory: when approaching the gap between the two central obstacles, it turns clockwise to cross the opening by moving backwards (2nd and 3rd snapshots).

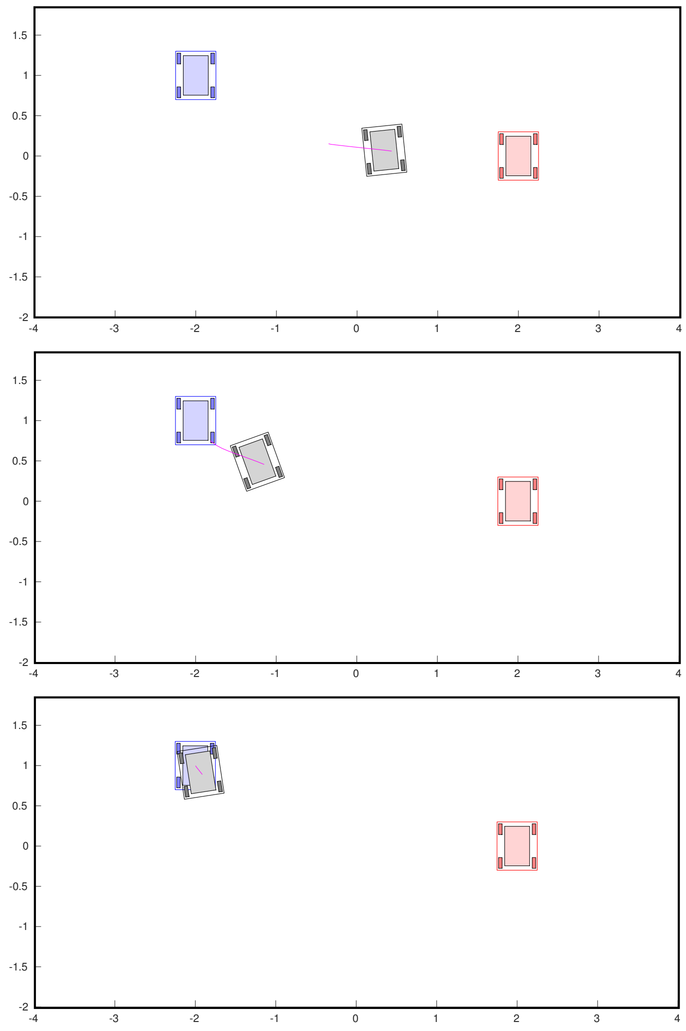


Fig. 4. Three snapshots of the motion that results in the "free movement" problem, taken after 2, 3.5, and 7 seconds, respectively.

4.3 Moving obstacle avoidance

The last test is to show the controller ability to modify the trajectory on-line to account for unexpected changes. It involves a dynamical environment consisting of four obstacles that cross the robot path moving in opposite directions. The robot is able to observe only the current position of the obstacles, and assumes they will keep moving with constant velocities as estimated from the last two configurations observed. An action sequence planned for one second ahead taking into account the current positions and estimated velocities of the obstacles, may lead to an unexpected collision if an obstacle suddenly changes its velocity and irrupts in the robot trajectory in the meantime. The controller deals with such eventualities by executing only the first action of the planned sequence and re-planning on-line the next one-second trajectory taking into account the new situation.

The result is shown in the accompanying video. It can be seen in `https://youtu.be/jr-4Tv9pAtY` how the initial trajectory is deflected downwards to avoid the first obstacle as it invades the robot planned path. The second obstacle is similarly avoided: at first, the planned trajectory tries to pass the obstacle by going up, but as the obstacle invades the path, a different trajectory is planned that drives the robot downwards. The third obstacle, however, forces a more drastic change of the trajectory. While trying to avoid the obstacle by going below it, the obstacle moves further downwards and blocks the path completely. At this point, the robot makes a turn and passes the obstacle by moving above it. Finally, the fourth obstacle has already gone out of the path when it is approached by the robot, so the robot heads directly towards the goal.

5. CONCLUSIONS

In this paper we have shown that model predictive control offers a powerful design methodology to obtain agile and energy-efficient controllers for mecanum wheeled robots. Due to its flexibility, this methodology can account for the full nonlinear dynamics of the robot and important additional constraints like actuator saturations, or collision avoidance constraints in changing environments. While all goal states in the paper have been static, note that the controller could also be used to drive the vehicle towards nonzero velocity states. The goal state could even be made time-varying, allowing to synchronise the robot with a specified goal trajectory. This would be useful, for example, to transfer a load between two wheeled robots moving in parallel along some path. The controller, moreover, could be employed as a local steering method if embedded into a higher-level motion planner. It would fit well onto the schemes of, for example, probabilistic roadmaps or rapidly-exploring random trees (LaValle, 2006). This

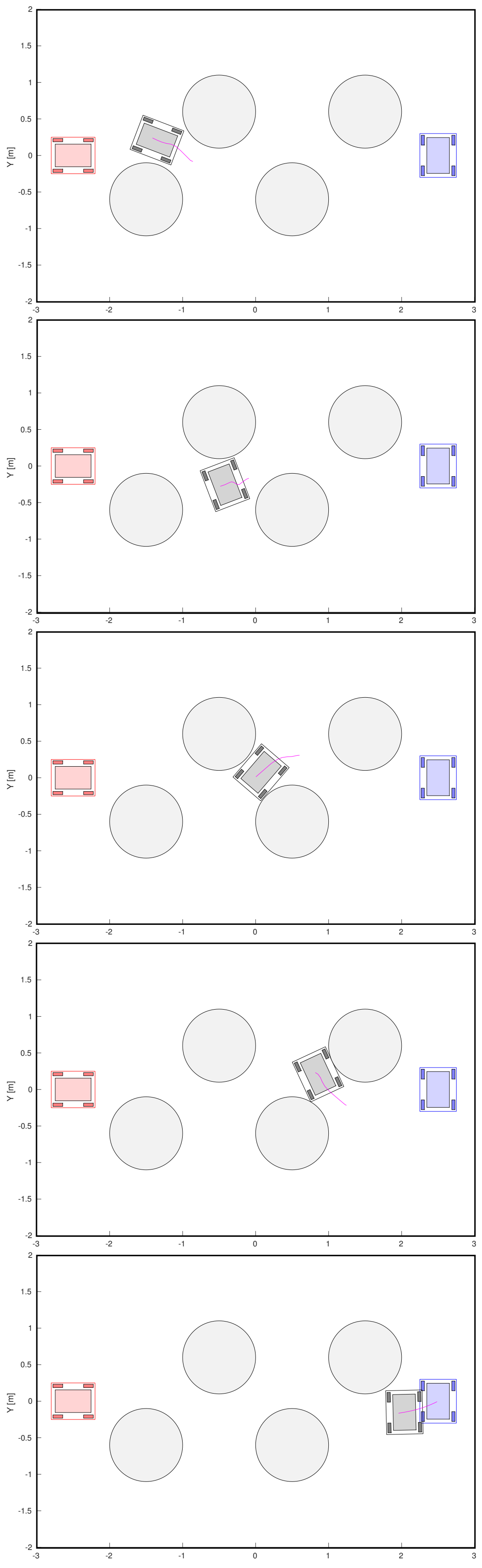


Fig. 5. Five snapshots of the solution for the "narrow corridor" problem.

application would be particularly interesting, as it would endow the robot with the capacity to navigate complex environments like labyrinthic workshops, or situations involving bug traps. The implementation of such possibilities and the experimental testing of the method in a real robot are part the current efforts by the authors (Moreno-Caireta, 2019).

REFERENCES

Adăscăliţei, F. and Doroftei, I. (2011). Practical applications for mobile robots based on mecanum wheels-a systematic survey. *The Romanian Review Precision Mechanics, Optics and Mechatronics*, 40, 21–29.

Agulló, J., Cardona, S., and Vivancos, J. (1987). Kinematics of vehicles with directional sliding wheels. *Mechanism and Machine Theory*, 22(4), 295–301.

Agulló, J., Cardona, S., and Vivancos, J. (1989). Dynamics of vehicles with directionally sliding wheels. *Mechanism and Machine Theory*, 24(1), 53–60.

Campion, G. and Chung, W. (2008). Wheeled robots. *Springer handbook of robotics*, 391–410.

Indiveri, G. (2009). Swedish wheeled omnidirectional mobile robots: Kinematics analysis and control. *IEEE Transactions on Robotics*, 25(1), 164–171.

Kuenemund, F., Hess, D., and Roehrig, C. (2016). Energy Efficient Kinodynamic Motion Planning for Holonomic AGVs in Industrial Applications using State Lattices. *Proceedings of ISR 2016: 47st International Symposium on Robotics*, 2016, 1–8.

Kuenemund, F., Hess, D., and Roehrig, C. (2017). Motion controller design for a mecanum wheeled mobile manipulator. *1st Annual IEEE Conference on Control Technology and Applications, CCTA 2017*, 2017-Janua(August), 444–449. doi:10.1109/CCTA.2017.8062502.

LaValle, S.M. (2006). *Planning algorithms*. Cambridge University Press.

Liu, Y., Zhu, J.J., Williams II, R.L., and Wu, J. (2008). Omni-directional mobile robot controller based on trajectory linearization. *Robotics and Autonomous Systems*, 56(5), 461–479.

Lynch, K.M. and Park, F.C. (2017). *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press.

Maciejowski, J.M. (2002). *Predictive control with constraints*. Pearson education.

Moreno-Caireta, I. (2019). *Model Predictive Control for a Mecanum-wheeled Robot in Dynamical Environments*. Master's thesis, Universitat Politècnica de Catalunya. `https://cutt.ly/3nonSzy`.

Potdar, N.D. (2018). *Online Trajectory Planning and Control of a MAV Payload System in Dynamic Environments*. Master's thesis, TU Delft.

Wampfler, G., Salecker, M., and Wittenburg, J. (1989). Kinematics, dynamics, and control of omnidirectional vehicles with mecanum wheels. *Mechanics Based Design of Structures and Machines*, 17(2), 165–177.

Zanelli, A., Domahidi, A., Jerez, J., and Morari, M. (2017). FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, 93(1), 13–29.