Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Master's degree in Advanced Mathematics and Mathematical Engineering

Master's thesis

Model Predictive Control for a Mecanum-wheeled robot in Dynamical Environments

Iñigo Moreno Caireta

Supervisor: Lluís Ros Giralt

Tutor: Mercè Ollé Torner

25th June, 2019

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Abstract

Mecanum-wheeled robots have been thoroughly used to automate tasks in many different applications. However, these robots are usually controlled by neglecting their dynamics and relying only on their kinematics alone. We model the behaviour of the robot by taking into account both the robot's geometry and the dynamic response of the actuators. All of the parameters of the robot are calculated, including its moments of inertia and friction on the actuators. We then introduce a controller for the system that takes into account the full non-linear dynamic model of the robot by using Model-Predictive Control (MPC). The controller is able to meet non-linear inequalities such as input limits and obstacle avoidance. The controller is very agile and can quickly adapt to changes in the environment. This allows control in unpredictable environments and can generate fast and energy efficient maneuvers.

Table of Contents

List of Figures

Acknowledgements

I am very thankful to Institut de Robòtica i Informàtica Industrial (IRI) for providing me access to their robots and installations. Especially to Lluís Ros, who has been an excellent supervisor and has helped me in every step of this project. I am also thankful to Ferran Cortés for constructing the setup required to measure the inertial parameters of the robot, to Patrick Grosch for his advice and support on the hardware aspects of the project and to Encric Celaya for his help on the first steps of the project.

Barcelona — Iñigo Moreno Caireta

25^{th} June, 2019

Chapter 1

Introduction

1-1 Theme Relevance and Justification

1-1-1 Omni-Wheeled Robots

Most ground robots and transport systems used today employ traditional wheels to move around. However, traditional wheels have an important feature: The movement of a wheel is limited to a one-dimensional displacement; it cannot move in the second dimension because of friction. This characteristic is beneficial in some cases. In cars or in other vehicles that go at high speeds it is important to avoid the freedom of movement so as to limit unwanted behaviour and make the maneuverability more intuitive for the driver. However, when considering lower speeds, this constraint is undesirable, as it forces the system to make several maneuvers to do simple movements (think, for example, of trying to park a car in a tight spot). It also increases the complexity of the robot as, to allow its turning, one has to be able to rotate the wheels.

Omni-directional wheels solve these problems by mounting rollers on the edge of the wheels, as can be seen in Fig. 1-1. The rollers only permit force to be applied in one direction, while the perpendicular direction is free to slide. As we will see in the thesis, having three or more of these wheels on the robot gives us full control of the displacement of the rollers, achieving full robot control.

This control permits the robot to move and rotate freely on the plane without needing to maneuver. The wheels do not even need to turn, making the robots much simpler in design and giving rise to many interesting applications such as those seen in Fig. 1-2. The down-side of omni-directional wheels is that they are very inefficient in terms of energy consumption, as the wheels sometimes produce opposing forces (as can be seen in the bottom right of Fig. 1-1). Therefore it would be good to have a controller that tries to minimize this effect.

Due to their freedom of movement and simplistic design, there has been a significant increase in the use of these robots in very different environments such as in intelligent storage facilities, assembly lines of trains or planes and many other applications. These setups usually

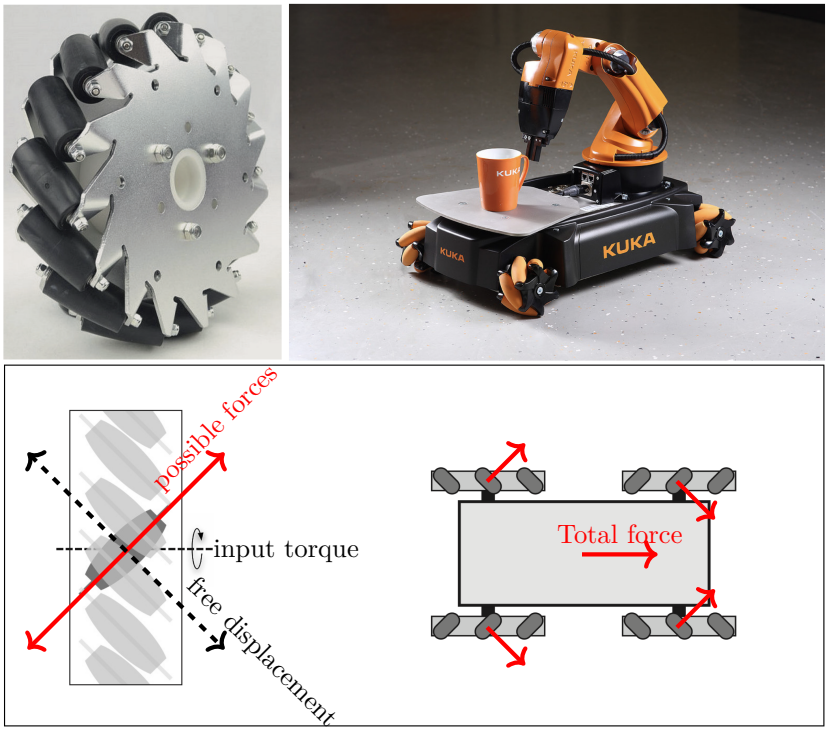


Figure 1-1: Mecanum wheels summary
Top: a mecanum wheel (left) and an omni-directional vehicle based on such wheels(right)
Bottom: Roller configuration (left) and directions of the ground reactions (right)

require the robot to go from one place to another without colliding with any of the elements of the environment. These obstacles can even be dynamic and have unpredictable behaviour such as human workers or other robots. Thus the goal of this work is to provide a controller for these kind of robots in dynamic environments.

1-1-2 Dynamic Model Importance

Omni-directional wheeled robots are a very well-studied subject in most mechanics courses and robotics books [1, 2]. However, the majority of systems that are used to control these robots are based on the kinematics of the robot alone, that is, they only take into account the velocities and positions of the robot, neglecting the acceleration and assuming that the robot can reach its desired velocity instantly.

Kinematic control works for certain scenarios, where the robot is light-weight and it can accelerate very fast, or where the environment is predictable and safe, so that the small errors that come from this assumption are admissible. However if, for example, we have a heavy-weight robot that needs some time to accelerate, or a highly unpredictable environment, we need to be able to control the robot by taking into account the inertia and the acceleration capacity of the robot. To do so we will need to obtain a dynamic model of the robot that takes into account both the equations of motion of the robot and the governing equations of its actuators in order to achieve a much more precise control.

Figure 1-2: Some applications of Mecanum wheels

Modelling the equations of the actuators will also allow us to compute the energy consumption of the system. This will later be used in order to minimize the energy consumption and achieve more efficient trajectories.

1-1-3 Model Predictive Control

A lot of control algorithms used for control of omni-wheeled robots are based on planning the whole system's movement for an operation and then having the robots follow the desired trajectory as close as possible [3]. This is not applicable to highly dynamical environments, as there could be obstacles that behave unexpectedly.

Our solution uses Model-Predictive Control (MPC), which is takes into account the dynamics of a dynamic model, a cost on the trajectories and some restrictions on the trajectories (all possibly non-linear). Using all of this information the MPC then outputs the trajectory which minimizes the total cost while complying with the restrictions. This repeated every time step in order to have real-time planning.

The use of a model predictive controller provides extra flexibility to the control system. If we encounter sudden changes in the environment or if our system behaves unexpectedly due to small inaccuracies of the model, the system will be able to account for those changes and generate a new plan from the current state.

1-2 Research Objective

The objective of the research included in this thesis is to:

> *Design a model predictive controller that allows the control of a mecanum-wheeled robot in highly unpredictable environment while also minimizing its energy consumption.*

We will identify an accurate electromechanical model of the robot and its actuators (including actuator friction). Then we will design and test this controller in simulation, but an effort will be made to take the first steps towards the empirical tests of the controller which could be done in a future stage.

1-3 Thesis Context

This thesis is a result of a collaboration between Facultat de Matemàtiques i Estadística (FME) and Institut de Robòtica i Informàtica Industrial (IRI). During the summer of 2018 I contacted Lluís Ros looking for a suitable project for my Master's Thesis and he proposed me to work on the topic described in this thesis. I decided to accept it and I have been working on it for the past year while doing my other master courses in parallel. The robot used for this thesis was a aquired by IRI from the robot maker company "SuperDroid Robots".

1-4 Notation

Throughout this thesis the following notation will be used:

Symbol	Definition
x	A scalar
$\dot{x}$	The time derivative of x
$\boldsymbol{x}$	A vector
$\boldsymbol{x}(i)$	The i-th element of vector $\boldsymbol{x}$
$\boldsymbol{X}$	A matrix
$\boldsymbol{R}_\theta$	The rotation matrix around the z-axis with angle θ
$\boldsymbol{I}_n$	The $n \times n$ identity matrix

Table 1-1: Thesis Notation

All vectors are column vectors and all positions are expressed in the East-North-Up (ENU) inertial frame unless otherwise stated.

Chapter 2

System model

2-1 Robot Geometry

2-1-1 Configuration Variables

The robot can be represented as a rigid body in a plane. The position of the robot's center of mass (COM) is defined by the coordinates x and y and the orientation of the robot is represented by the yaw angle ψ, all relative to an absolute coordinate system.

Assuming the robot has n wheels of radius r, we can define the rotation angle of each wheel as φ_i. Every wheel is in a position (s_i, d_i) and orientation β_i in a coordinate frame fixed to the robot and centered on the COM. The rollers of the wheel are at an angle α_i with respect to the wheel. A representation of the wheel configuration can be seen in Fig. 2-1.

Using these variables we can fully define the current configuration of the chassis using the variable $\boldsymbol{q}_r = \begin{bmatrix} x & y & \psi \end{bmatrix}^T$ and the configuration of the wheels using the variable $\boldsymbol{q}_w = \begin{bmatrix} \varphi_1 & \dots & \varphi_n \end{bmatrix}^T$. Let us call $\boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_r \\ \boldsymbol{q}_w \end{bmatrix}$.

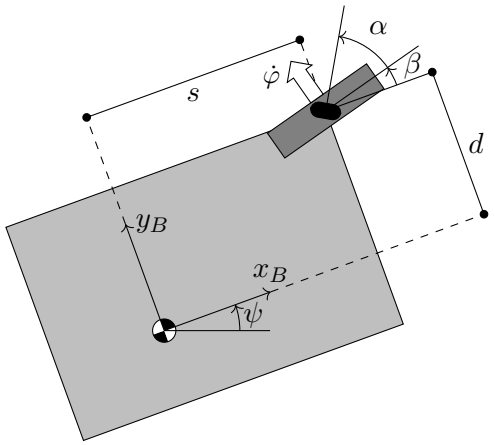
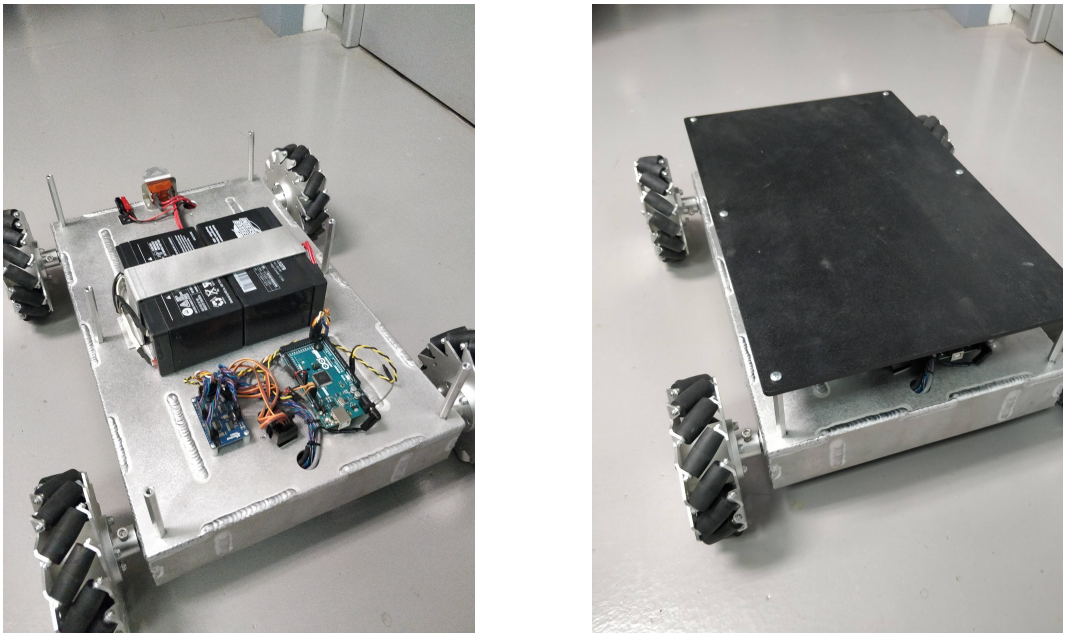


Figure 2-1: Wheel configuration

2-1-2 Specification of the Robot

The robot we used is made by "SuperDroid Robots"[1], a robotics company based in North Carolina (USA). As it has been built by an American company, a lot of the robot parameters are given in imperial units. Pictures of the robot can be seen in Fig. 2-2. A schematic of the robot configuration can be seen in Fig. 2-3 and Tables 2-1 and 2-2 provide its main geometric and mass parameters.



(a) Without the plastic lid **(b)** With the plastic lid

Figure 2-2: Global views of the robot

[1] https://www.superdroidrobots.com/shop/item.aspx/programmable-mecanum-wheel-vectoring-robot-ig42-db/2458/

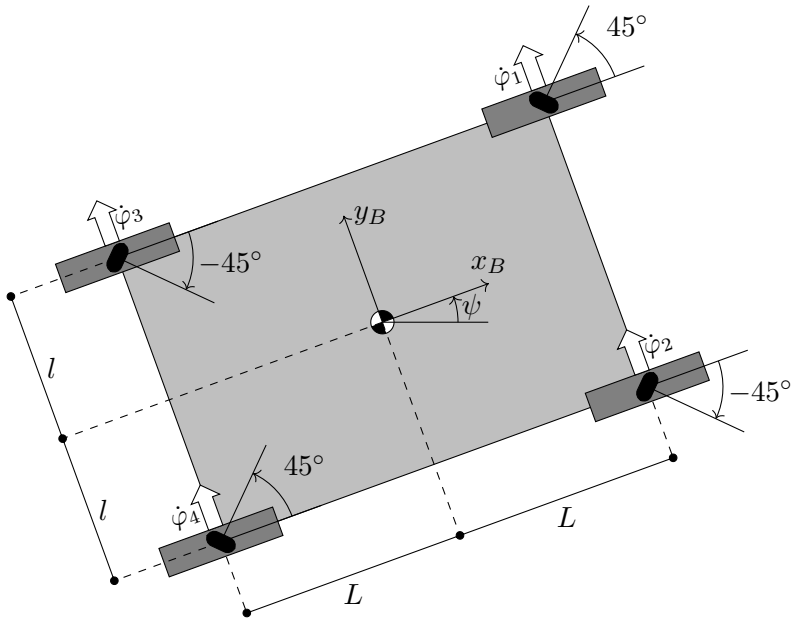


Figure 2-3: Robot configuration

L	l	r	m
8.25 in	8.25 in	2.625 in	15.75 kg

Table 2-1: Specification of robot variables

Wheel	α	β	s	d
1	$45°$	0	L	l
2	$-45°$	0	L	$-l$
3	$-45°$	0	$-L$	l
4	$45°$	0	$-L$	$-l$

Table 2-2: Specification of wheel variables

2-2 Kinematic Constraints

Although the variables $\boldsymbol{q}_r$ and $\boldsymbol{q}_w$ represent the full configuration of the robot, they are not independent from each other. As we will see, there is a kinematic constraint relating their time derivatives.

Let us focus on point C at the center of a wheel. Note that we can consider this point to be both a point on the chassis and a point on the wheel. Therefore, there are two different ways to calculate the velocity of C: either as a point in the wheel or as a point in the chassis. By equating the resulting velocity expressions we will find an equality constraint relating $\dot{\boldsymbol{q}}_r$ and $\dot{\boldsymbol{q}}_w$.

C as a point of the chassis

If we regard C as a point of the chassis, we can use a basis B fixed to the chassis to deduce its velocity in this basis from the velocity of the COM and the angular velocity $\dot{\psi}$. For simplicity let point O denote the COM, and let its speed in the basis B be given by $\begin{bmatrix} v_1 & v_2 \end{bmatrix}^T$. Then, the velocity of C in basis B is given by:

$$[\boldsymbol{v}(C)]_B = [\boldsymbol{v}(O)]_B + \boldsymbol{\Omega}_B \times \left[\overrightarrow{OC}\right]_B$$

$$[\boldsymbol{v}(C)]_B = \begin{bmatrix} v_1 \\ v_2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} s \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} v_1 - \dot{\psi} d \\ v_2 + \dot{\psi} s \\ 0 \end{bmatrix}$$

or, if we only keep the xy components:

$$[\boldsymbol{v}(C)]_B = \begin{bmatrix} 1 & 0 & -d \\ 0 & 1 & s \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \dot{\psi} \end{bmatrix} \tag{2-1}$$

C as a point of the wheel

If we regard C as a point of the wheel, we can use a basis B' that is fixed to the robot but with the same angle than the wheel. Let J be the contact point between the ground and the wheel and assume that this point is sliding along the ground with speed σ. A representation of these variables can be seen in Fig. 2-4. We can now find the speed of C in basis B' based on the velocity of J and the angular velocities of both the wheel and the chassis:

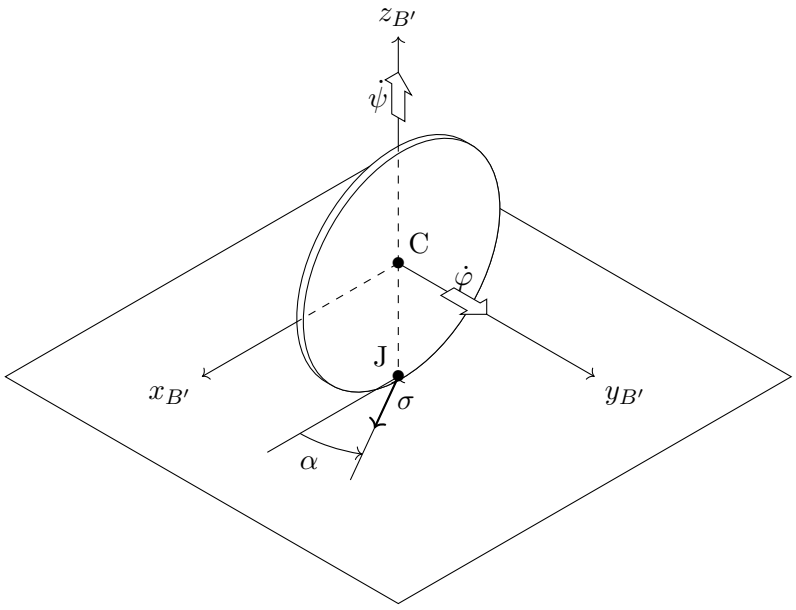


Figure 2-4: Wheel detail

$$[\boldsymbol{v}(C)]_{B'} = [\boldsymbol{v}(J)]_{B'} + \boldsymbol{\Omega}_{B'} \times \left[\overrightarrow{JC}\right]_{B'}$$

$$[\boldsymbol{v}(C)]_{B'} = \begin{bmatrix} \sigma\cos\alpha \\ \sigma\sin\alpha \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\varphi} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ r \end{bmatrix} = \begin{bmatrix} \sigma\cos\alpha + \dot{\varphi}r \\ \sigma\sin\alpha \\ 0 \end{bmatrix}$$

or, if we only keep the xy components:

$$[\boldsymbol{v}(C)]_{B'} = \begin{bmatrix} r & \cos\alpha \\ 0 & \sin\alpha \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \sigma \end{bmatrix} \tag{2-2}$$

Finding the constraint

We have found two expressions (Eqs. (2-1) and (2-2)) for $\boldsymbol{v}(C)$, one in basis B and the other in basis B'. As we have stated before, the difference between these two bases is a rotation of angle β about the z axis. Therefore we can write:

$$\begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix} [\boldsymbol{v}(C)]_{B'} = [\boldsymbol{v}(C)]_B$$

$$\begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} r & \cos\alpha \\ 0 & \sin\alpha \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \sigma \end{bmatrix} = \begin{bmatrix} 1 & 0 & -d \\ 0 & 1 & s \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \dot{\psi} \end{bmatrix}$$

$$\begin{bmatrix}\dot{\varphi}\\ \sigma\end{bmatrix} = \begin{bmatrix} r & \cos\alpha \\ 0 & \sin\alpha \end{bmatrix}^{-1} \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & -d \\ 0 & 1 & s \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \dot{\psi} \end{bmatrix}$$

$$\begin{bmatrix}\dot{\varphi}\\ \sigma\end{bmatrix} = \begin{bmatrix} \frac{\sin(\alpha+\beta)}{r\sin\alpha} & \frac{-\cos(\alpha+\beta)}{r\sin\alpha} & \frac{-d\sin(\alpha+\beta)-s\cos(\alpha+\beta)}{r\sin\alpha} \\ \frac{-\sin\beta}{\sin\alpha} & \frac{\cos\beta}{\sin\alpha} & \frac{d\sin\beta+s\cos\beta}{\sin\alpha} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \dot{\psi} \end{bmatrix}$$

Assigning a variable to each of the elements, the previous can be written in the more compact form:

$$\begin{bmatrix}\dot{\varphi}\\ \sigma\end{bmatrix} = \begin{bmatrix} A & B & C \\ a & b & c \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \dot{\psi} \end{bmatrix}$$

For a robot with n wheels:

$$\begin{bmatrix}\dot{\varphi}_1\\ \dot{\varphi}_2\\ \vdots \\ \dot{\varphi}_n\end{bmatrix} = \underbrace{\begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ \vdots & \vdots & \vdots \\ A_n & B_n & C_n \end{bmatrix}}_{\boldsymbol{R}} \begin{bmatrix} v_1 \\ v_2 \\ \dot{\psi} \end{bmatrix} \tag{2-3}$$

Let $\boldsymbol{R}$ be the matrix indicated in Eq. (2-3). If we remember, we defined v_1 and v_2 as the velocity components of the COM in basis B. However, note that

$$\begin{bmatrix} v_1 \\ v_2 \\ \dot{\psi} \end{bmatrix} = \boldsymbol{R}_\psi^T \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix}$$

where

$$\boldsymbol{R}_\psi = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Therefore we can rewrite Eq. (2-3) as

$$\dot{\boldsymbol{q}}_w = \boldsymbol{R}\boldsymbol{R}_\psi^T \dot{\boldsymbol{q}}_r \tag{2-4}$$

which is the constraint on $\dot{\boldsymbol{q}}_w$ and $\dot{\boldsymbol{q}}_r$ that we were looking for.

Particularization for our robot

In Table 2-1 we can find the values of α, β, s and d for all of the wheels in our robot. We can use these values to construct the matrix $\boldsymbol{R}$ for our case. Therefore, for our robot:

$$\boldsymbol{R} = \frac{1}{r}\begin{bmatrix} 1 & -1 & -(L+l) \\ 1 & 1 & L+l \\ 1 & 1 & -(L+l) \\ 1 & -1 & L+l \end{bmatrix} \tag{2-5}$$

2-3 Mechanical Model

To develop the dynamic model for this system we will use Lagrange's Equations with multipliers. However, it is first important to explain the plain Lagrange's Equations to understand why we need the multiplier form of the equation.

2-3-1 Lagrange's Equations

To use Lagrange's equations we must define the variables that can fully define the current configuration of the system. This will be the vector $\boldsymbol{q} = \begin{bmatrix} x & y & \psi & \varphi_1 & \dots & \varphi_n \end{bmatrix}^T$ that we defined earlier. We must also find the potential and kinetic energy functions of the system in terms of the configuration variables and their derivatives. As our robot moves on a flat surface, the potential energy will be constant and only the kinetic energy of the system will be relevant. This energy can be written in terms of $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ as follows:

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}m\dot{y}^2 + \frac{1}{2}I_z\dot{\psi}^2 + \sum_{k=1}^{n}\frac{1}{2}I_k\dot{\varphi}_k^2 \tag{2-6}$$

where m is the total mass of the system, I_z is the Inertia of the whole system around the z-axis at the COM and I_k is the inertia of the k-th wheel around its spin axis. We can rewrite this in matrix form as

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2}\,\dot{\boldsymbol{q}}^T\,\boldsymbol{M}\,\dot{\boldsymbol{q}} \tag{2-7}$$

where

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{M}_r & \\ & \boldsymbol{M}_w \end{bmatrix} \quad \boldsymbol{M}_r = \begin{bmatrix} m & & \\ & m & \\ & & I_z \end{bmatrix} \quad \boldsymbol{M}_w = \begin{bmatrix} I_{w1} & & \\ & \ddots & \\ & & I_{wn} \end{bmatrix}$$

We now should formulate the equation

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial T}{\partial \boldsymbol{q}} = \boldsymbol{F} \tag{2-8}$$

where $\boldsymbol{F}$ is the vector of generalized external forces. In our case, the generalized forces associated with $\boldsymbol{q}_r$ are zero and the generalized forces associated with $\boldsymbol{q}_w$ are $\boldsymbol{\Gamma} = \begin{bmatrix}\tau_1 & \dots & \tau_n\end{bmatrix}^T$, where τ_i is the torque applied by each wheel. Then $\boldsymbol{F} = \left[\begin{smallmatrix}\mathbf{0}\\ \boldsymbol{\Gamma}\end{smallmatrix}\right]$.

Solving Eq. (2-8) would give us an expression to find the accelerations $\ddot{\boldsymbol{q}}$ based on the positions $\boldsymbol{q}$, the velocities $\dot{\boldsymbol{q}}$ and the external forces $\boldsymbol{F}$, i.e., the solution of the forward dynamic problem.

Solving Eq. (2-8) gives us

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{F} \tag{2-9}$$

However Lagrange's equations can only be used when all the constraints acting on our system are holonomic (i.e, when they can be written as an equality constraint of the form $f(\boldsymbol{q}) = 0$). The restriction imposed by Eq. (2-4) is non-holonomic, as it constrains the velocities. To solve this problem we will need to use Lagrange's equations with multipliers. This is an important fact to realise as it means that using the plain Lagrange equations gives us a wrong result for the dynamics of the system. In fact, while writing this thesis I have seen other papers such as [4] that use n incorrect dynamic model because of this error.

2-3-2 Lagrange's Equations with Multipliers

In our case we have a restriction on the equations that can be written in the form $\boldsymbol{C}\dot{\boldsymbol{q}} = \mathbf{0}$, where $\boldsymbol{C}$ is a matrix that only depends on $\boldsymbol{q}$. To find $\boldsymbol{C}$ let us proceed from Eq. (2-4):

$$\dot{\boldsymbol{q}}_w = \boldsymbol{R}\boldsymbol{R}_\psi^T\dot{\boldsymbol{q}}_r$$

$$-\boldsymbol{R}\boldsymbol{R}_\psi^T\dot{\boldsymbol{q}}_r + \boldsymbol{I}_n\dot{\boldsymbol{q}}_w = \mathbf{0}$$

$$\left[\begin{array}{c|c} -\boldsymbol{R}\boldsymbol{R}_\psi^T & \boldsymbol{I}_n \end{array}\right] \begin{bmatrix}\dot{\boldsymbol{q}}_r \\ \hline \dot{\boldsymbol{q}}_w\end{bmatrix} = \mathbf{0}$$

$$\boldsymbol{C}\dot{\boldsymbol{q}} = \mathbf{0} \tag{2-10}$$

$$\boldsymbol{C} = \left[\begin{array}{c|c} -\boldsymbol{R}\boldsymbol{R}_\psi^T & \boldsymbol{I}_n \end{array}\right] \tag{2-11}$$

For our wheel setup:

$$\boldsymbol{C} = \begin{bmatrix} -\frac{\cos\psi}{r} - \frac{\sin\psi}{r} & \frac{\cos\psi}{r} - \frac{\sin\psi}{r} & \frac{L+l}{r} & 1 & 0 & 0 & 0 \\ \frac{\sin\psi}{r} - \frac{\cos\psi}{r} & -\frac{\cos\psi}{r} - \frac{\sin\psi}{r} & -\frac{L+l}{r} & 0 & 1 & 0 & 0 \\ \frac{\sin\psi}{r} - \frac{\cos\psi}{r} & -\frac{\cos\psi}{r} - \frac{\sin\psi}{r} & \frac{L+l}{r} & 0 & 0 & 1 & 0 \\ -\frac{\cos\psi}{r} - \frac{\sin\psi}{r} & \frac{\cos\psi}{r} - \frac{\sin\psi}{r} & -\frac{L+l}{r} & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-12}$$

When we have these kind of restrictions we can rewrite Eq. (2-8) into:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial T}{\partial \boldsymbol{q}} + \boldsymbol{C}^T\boldsymbol{\lambda} = \boldsymbol{F} \tag{2-13}$$

where $\boldsymbol{\lambda}$ are the Lagrange multipliers, a new unknown vector with the same dimension as our restriction. As we are adding additional unknowns, we must also add additional equations to be able to solve the system. These additional equations come from differentiating Eq. (2-10) with respect to time:

$$\begin{aligned} \frac{d}{dt}\left(\boldsymbol{C}\dot{\boldsymbol{q}}\right) &= \boldsymbol{0} \\ \dot{\boldsymbol{C}}\dot{\boldsymbol{q}} + \boldsymbol{C}\ddot{\boldsymbol{q}} &= \boldsymbol{0} \end{aligned} \tag{2-14}$$

2-3-3 Solving the System

When computing the derivatives in Eq. (2-13) we obtain:

$$\begin{aligned} \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{C}^T\boldsymbol{\lambda} &= \boldsymbol{F} \\ \begin{bmatrix} \boldsymbol{M}_r & \\ & \boldsymbol{M}_w \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}}_r \\ \ddot{\boldsymbol{q}}_w \end{bmatrix} + \begin{bmatrix} -\boldsymbol{R}_\psi \boldsymbol{R}^T \\ \boldsymbol{I}_n \end{bmatrix} \lambda &= \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\Gamma} \end{bmatrix} \\ \boldsymbol{M}_r\ddot{\boldsymbol{q}}_r - \boldsymbol{R}_\psi\boldsymbol{R}^T\boldsymbol{\lambda} &= \boldsymbol{0} \end{aligned} \tag{2-15}$$

$$\boldsymbol{M}_w\ddot{\boldsymbol{q}}_w + \boldsymbol{\lambda} = \boldsymbol{\Gamma} \tag{2-16}$$

We can also operate on Eq. (2-14):

$$\begin{aligned} \dot{\boldsymbol{C}}\dot{\boldsymbol{q}} + \boldsymbol{C}\ddot{\boldsymbol{q}} &= \boldsymbol{0} \\ \left[\begin{array}{c|c} -\boldsymbol{R}\dot{\boldsymbol{R}}_\psi^T & \boldsymbol{0} \end{array}\right] \begin{bmatrix} \dot{\boldsymbol{q}}_r \\ \dot{\boldsymbol{q}}_w \end{bmatrix} + \left[\begin{array}{c|c} -\boldsymbol{R}\boldsymbol{R}_\psi^T & \boldsymbol{I}_n \end{array}\right] \begin{bmatrix} \ddot{\boldsymbol{q}}_r \\ \ddot{\boldsymbol{q}}_w \end{bmatrix} &= \boldsymbol{0} \\ -\boldsymbol{R}\dot{\boldsymbol{R}}_\psi^T\dot{\boldsymbol{q}}_r - \boldsymbol{R}\boldsymbol{R}_\psi^T\ddot{\boldsymbol{q}}_r + \ddot{\boldsymbol{q}}_w &= \boldsymbol{0} \end{aligned} \tag{2-17}$$

We now have a system of equations given by Eqs. (2-15) to (2-17) where the unknowns are $\ddot{\boldsymbol{q}}_r$, $\ddot{\boldsymbol{q}}_w$ and $\boldsymbol{\lambda}$. To solve it we substitute $\ddot{\boldsymbol{q}}_w$ from Eq. (2-17) to Eq. (2-16):

$$\begin{aligned} \boldsymbol{M}_w \overbrace{\left(\boldsymbol{R}\dot{\boldsymbol{R}}_\psi^T\dot{\boldsymbol{q}}_r + \boldsymbol{R}\boldsymbol{R}_\psi^T\ddot{\boldsymbol{q}}_r\right)}^{\ddot{\boldsymbol{q}}_w} + \boldsymbol{\lambda} &= \boldsymbol{\Gamma} \\ \boldsymbol{\lambda} = \boldsymbol{\Gamma} - \boldsymbol{M}_w\boldsymbol{R}\dot{\boldsymbol{R}}_\psi^T\dot{\boldsymbol{q}}_r - \boldsymbol{M}_w\boldsymbol{R}\boldsymbol{R}_\psi^T\ddot{\boldsymbol{q}}_r \end{aligned} \tag{2-18}$$

Now let's substitute $\boldsymbol{\lambda}$ from Eq. (2-18) to Eq. (2-15):

$$\begin{aligned} \boldsymbol{M}_r\ddot{\boldsymbol{q}}_r - \boldsymbol{R}_\psi\boldsymbol{R}^T \overbrace{\left(\boldsymbol{\Gamma} - \boldsymbol{M}_w\boldsymbol{R}\dot{\boldsymbol{R}}_\psi^T\dot{\boldsymbol{q}}_r - \boldsymbol{M}_w\boldsymbol{R}\boldsymbol{R}_\psi^T\ddot{\boldsymbol{q}}_r\right)}^{\boldsymbol{\lambda}} &= \boldsymbol{0} \\ \boldsymbol{M}_r\ddot{\boldsymbol{q}}_r - \boldsymbol{R}_\psi\boldsymbol{R}^T\boldsymbol{\Gamma} + \boldsymbol{R}_\psi\boldsymbol{R}^T\boldsymbol{M}_w\boldsymbol{R}\dot{\boldsymbol{R}}_\psi^T\dot{\boldsymbol{q}}_r + \boldsymbol{R}_\psi\boldsymbol{R}^T\boldsymbol{M}_w\boldsymbol{R}\boldsymbol{R}_\psi^T\ddot{\boldsymbol{q}}_r &= \boldsymbol{0} \\ \underbrace{\left(\boldsymbol{M}_r + \boldsymbol{R}_\psi\boldsymbol{R}^T\boldsymbol{M}_w\boldsymbol{R}\boldsymbol{R}_\psi^T\right)}_{\boldsymbol{H}}\ddot{\boldsymbol{q}}_r + \underbrace{\boldsymbol{R}_\psi\boldsymbol{R}^T\boldsymbol{M}_w\boldsymbol{R}\dot{\boldsymbol{R}}_\psi^T}_{\boldsymbol{K}}\dot{\boldsymbol{q}}_r &= \underbrace{\boldsymbol{R}_\psi\boldsymbol{R}^T\boldsymbol{\Gamma}}_{\boldsymbol{F}_a} \end{aligned} \tag{2-19}$$

As we can see, Eq. (2-19) gives us a formula for the dynamics of the robot simplified to only the variables $\boldsymbol{q}_r$. This is very useful as we have reduced the dimension of our problem from 7 to 3 dimensions.

2-3-4 Solution for our Configuration

Let's now compute the values for the 3 matrices $\boldsymbol{H}$, $\boldsymbol{K}$ and $\boldsymbol{F}_a$ from Eq. (2-19). We will use the fact that, for our configuration, all wheels are equal and therefore $I_1 = \cdots = I_n = I_w$. Computing and simplifying all of these matrices we get:

$$\boldsymbol{H} = \begin{bmatrix} m + \frac{4\,I_w}{r^2} & 0 & 0 \\ 0 & m + \frac{4\,I_w}{r^2} & 0 \\ 0 & 0 & I_z + \frac{4\,I_w\,(L+l)^2}{r^2} \end{bmatrix} \tag{2-20}$$

$$\boldsymbol{K} = \begin{bmatrix} 0 & \frac{4\,I_w\,\dot{\psi}}{r^2} & 0 \\ -\frac{4\,I_w\,\dot{\psi}}{r^2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2-21}$$

$$\boldsymbol{F}_a = \begin{bmatrix} \frac{\sin\psi\,(\tau_1 - \tau_2 - \tau_3 + \tau_4)}{r} + \frac{\cos\psi\,(\tau_1 + \tau_2 + \tau_3 + \tau_4)}{r} \\ \frac{\sin\psi\,(\tau_1 + \tau_2 + \tau_3 + \tau_4)}{r} - \frac{\cos\psi\,(\tau_1 - \tau_2 - \tau_3 + \tau_4)}{r} \\ -\frac{(L+l)\,(\tau_1 - \tau_2 + \tau_3 - \tau_4)}{r} \end{bmatrix} \tag{2-22}$$

2-3-5 Importance of a Good Configuration

In Eq. (2-20) we can see that the matrix $\boldsymbol{H}$ no longer depends on ψ. This is because the matrix $\boldsymbol{R}^T\boldsymbol{R}$ has the form

$$\begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & b \end{bmatrix}$$

and therefore $\boldsymbol{R}_\psi \boldsymbol{R}^T \boldsymbol{R} \boldsymbol{R}_\psi^T = \boldsymbol{R}^T\boldsymbol{R}$.

Let's look in which other configurations this happens. If we recall how we defined $\boldsymbol{R}$ in Eq. (2-3):

$$\boldsymbol{R} = \begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ \vdots & \vdots & \vdots \\ A_n & B_n & C_n \end{bmatrix} \quad \text{Where:} \quad \begin{aligned} A_i &= \tfrac{\sin(\alpha_i+\beta_i)}{r\sin\alpha_i} \\ B_i &= \tfrac{-\cos(\alpha_i+\beta_i)}{r\sin\alpha_i} \\ C_i &= \tfrac{-d_i\sin(\alpha_i+\beta_i) - s_i\cos(\alpha_i+\beta_i)}{r\sin\alpha_i} \end{aligned}$$

$$\boldsymbol{R}^T\boldsymbol{R} = \sum_{i=0}^{n} \begin{bmatrix} {A_i}^2 & A_i\,B_i & A_i\,C_i \\ A_i\,B_i & {B_i}^2 & B_i\,C_i \\ A_i\,C_i & B_i\,C_i & {C_i}^2 \end{bmatrix}$$

One way of satisfying the condition set above is to set that $A_i^2 = B_i^2$ for all i. Then:

$$\begin{aligned} A_i^2 &= B_i^2 \\ \sin(\alpha_i + \beta_i)^2 &= \cos(\alpha_i + \beta_i)^2 \\ \alpha_i + \beta_i &= \pi/4 \mod \pi/2 \end{aligned} \tag{2-23}$$

This shows the advantage of having the wheels set at 45 degree angles. Now let's try to find other conditions. One way of satisfying Eq. (2-23) is to set $\beta_i = 0$ for all i and substitute α_i by $(\pi/4 + k_i\,\pi/2)$ where k_i is an integer number. Then we can use the fact that $2\sin(\alpha_i)\cos(\alpha_i) = \sin(2\alpha_i) = \sin(\pi/2 + \pi k_i) = \cos(\pi k_i) = (-1)^{k_i}$ to compute the following:

$$\boldsymbol{R}^T\boldsymbol{R} = \frac{1}{r^2}\sum_{i=0}^{n}\begin{bmatrix} 1 & -(-1)^{k_i} & -d_i - (-1)^{k_i}\,s_i \\ -(-1)^{k_i} & 1 & s_i + (-1)^{k_i}\,d_i \\ -d_i - (-1)^{k_i}\,s_i & s_i + (-1)^{k_i}\,d_i & (s_i + (-1)^{k_i} d_i)^2 \end{bmatrix} \tag{2-24}$$

From this we can find three other conditions needed:

$$\sum_{i=0}^{n}(-1)^{k_i} = 0 \qquad \sum_{i=0}^{n} s_i + (-1)^{k_i} d_i = 0 \qquad \sum_{i=0}^{n} d_i + (-1)^{k_i} s_i = 0$$

If all of these conditions are met we get:

$$\boldsymbol{R}^T\boldsymbol{R} = \frac{1}{r^2}\sum_{i=0}^{n}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & (s_i + (-1)^{k_i} d_i)^2 \end{bmatrix} \tag{2-25}$$

And now the condition is fully satisfied.

2-3-6 Measuring Inertia

As we have seen before, we need to calculate the inertia of the vehicle and the inertia of the wheel. As they are both complex shapes we cannot simply assume that the mass is evenly distributed across a simple shape. Therefore we conducted an experiment to determine these moments of inertia. The experiments is as follows:

First we need to hang the object to be measured from cables. The cables must have attach to the object in an horizontal plane with the attachment points forming an equilateral triangle. The cables must all go perpendicular to the ground and hang from a horizontal platform. We then turn the object by a small angle and let it oscillate. The period of the oscillations can then be measured and from it, using some computation that we will explain now, obtain the moment of inertia.

Calculations

Let us define the angles ψ and θ as the rotation of the object and the rotation of the cables respectively. The length of the cable will be called h and the distance between the anchor points and the center of mass r. You can see a schematic of these variables in Fig. 2-5. Let us call F the tension force in each cable (all equal due to rotational symmetry). From the Newton-Euler equations of dynamics we get two equations (one for the vertical forces and another for the rotational torques):

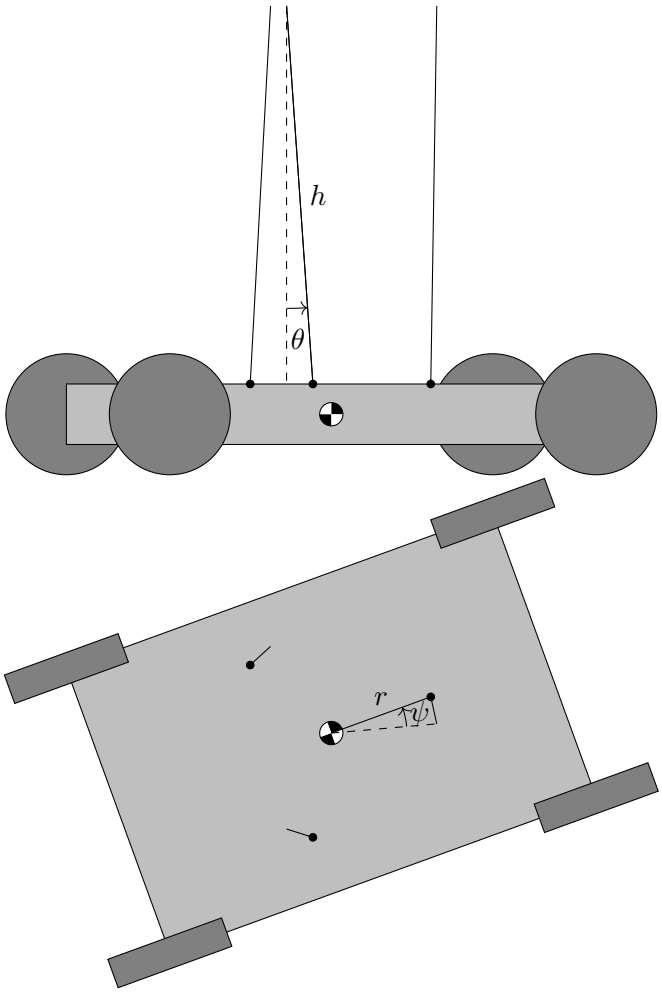


Figure 2-5: Side view and top view of the suspended robot

$$\left.\begin{array}{l} mg = 3\,F\cos(\theta) \\ I\,\ddot{\psi} = -r\,3\,F\sin(\theta) \end{array}\right\} I\,\ddot{\psi} = -m\,g\,r\,\tan(\theta)$$

$$\text{Assuming small angles: } \tan(\theta) \simeq \theta \text{ and } h\theta \simeq r\psi$$

$$\ddot{\psi} + \frac{mg\,r^2}{h\,I}\psi = 0 \tag{2-26}$$

The ODE obtained in Eq. (2-26) is of the type $\ddot{x} + kx = 0$, which is a well-known class of ODEs which has oscillatory solutions with frequency $w = \sqrt{k}$. Therefore, the period T can be calculated as follows:

$$T = \frac{2\pi}{w} = \frac{2\pi}{\sqrt{k}} \rightarrow T^2 = \frac{4\,\pi^2\,h\,I}{mg\,r^2}$$

Solving for I in the previous equation we obtain:

$$I = \frac{mg\,r^2\,T^2}{4\,\pi^2\,h} \tag{2-27}$$

which gives us a formula to find the moment of inertia from the period, as we were looking for.

Building the experiment

As we want to do the experiment for both the robot and the wheels, we must find a way to suspend both onjects from cables attached in a equilateral triangle centered on the COM.

For the wheels this is an easy task, as they came with six holes on one side that formed a regular hexagon. We can therefore attach small hooks on each hole and use them as anchor points. In Fig. 2-7 you can see how the wheel is suspended from the cables.

For the robot we have no holes distributed in such way, and we couldn't add any holes to the robot as we didn't want to perforate any part of the robot. Therefore we decided to substitute the black plastic cover of the robot by a wooden one where we could make the holes needed and attach the hooks. Fig. 2-6 shows how the robot is suspended from the cables.

Now we have the hooks we need to build a platform with hooks distributed in triangles of the same size as the ones in the robot and the wheel so as to attach the cables perpendicularly to the ground. To do this we used another wooden plank and drilled some holes to attach the hooks. This platform is held above ground by placing it on a structure that was already constructed at Institut de Robòtica i Informàtica Industrial (IRI) for another project.

To obtain the values of the period we recorded the oscillations in slow-motion and used video editing software to find the exact frames where the oscillations were at a maximum.

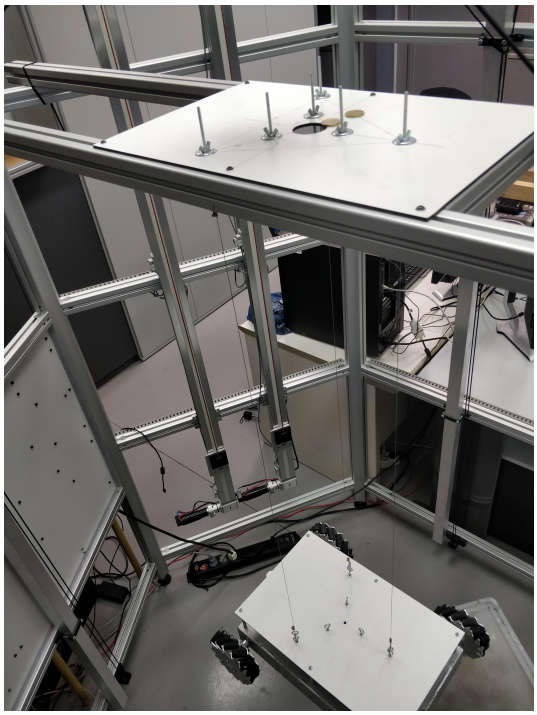


Figure 2-6: Robot suspended from cables

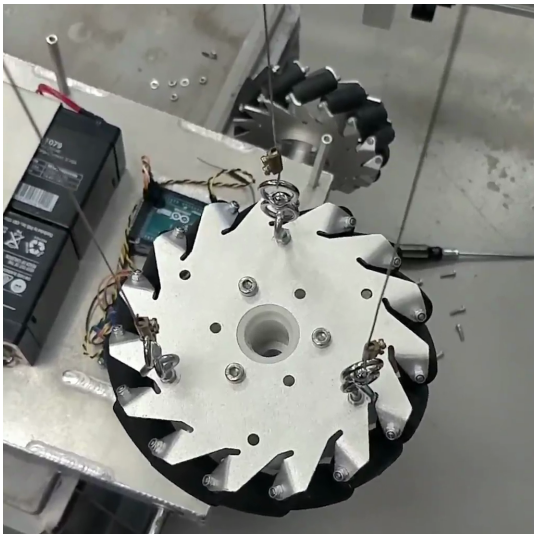


Figure 2-7: Wheel suspended from cables

Results

The measurements of the experiment and the result of the calculations can be seen in Table 2-3.

Magnitude	Robot	Wheel
m	$15.75\,\mathrm{kg}$	$575\,\mathrm{g}$
r	$130\,\mathrm{mm}$	$48.25\,\mathrm{mm}$
h	$112.5\,\mathrm{cm}$	$112.5\,\mathrm{cm}$
T	$2.8\,\mathrm{s}$	$3\,\mathrm{s}$
I	$461\,\mathrm{g\,m^2}$	$2.66\,\mathrm{g\,m^2}$

Table 2-3: Results of inertia measurements

2-4 Motor Model

We have found a full model that can predict the behaviour of the system based on the torques produced by the motors τ_i. However, our input is not the torque, but the voltage to the motors (V_i). Therefore we need to find a way to get τ_i from V_i.

2-4-1 Motor Dynamic Identification

The motors installed in the robot are IG-42CRGM[2], which are 24 V DC motors with a 1/49 reduction ratio. These motors are controlled by voltage inputs. The motors are also equipped with encoders from which we can compute the angle of each wheel. DC motors can be characterized by the following equations:

$$V = NK_m\dot{\varphi} + Ri + L\frac{di}{dt} \tag{2-28}$$

$$\tau_m = \eta N K_e i \tag{2-29}$$

where V and i are the total voltage and intensity through the motor, N,R,K_e,K_m,η and L are parameters of the motor, τ_m is the torque applied by the motor and φ the wheel's angle.

We will now do two assumptions, first we will assume that $K_m = K_e$. This is strange, as K_m and K_e are not in the same units, but when converted to SI units both have the same value. We can call both of them K. We can also assume L to be negligible, as it is generally small. We now combine Eqs. (2-28) and (2-29) into:

$$\tau_m = \frac{\eta N K}{R}V - \frac{\eta N^2 K^2}{R}\dot{\varphi} \tag{2-30}$$

To find the values of N, R, K and η we must look at the datasheet of the motor. The values of N and η, are directly on the datasheet and its values are 49 and 0.6. However, to find R and K, we need to do some computations based on a graph from the datasheet, which can be seen in Fig. 2-8.

As we can see at the right of Fig. 2-8, when the wheel is completely locked $\dot{\varphi} = 0$, $i = 13\,\text{A}$, $V = 24\,\text{V}$. With this information we can use Eq. (2-28) to find that $R = \frac{24\,\text{V}}{13\,\text{A}} \simeq 1.8462\,\Omega$.

As we can see at the left of Fig. 2-8, when the wheel rotates freely $i = 0$, $\dot{\varphi} = 7000\,\text{rpm}$, $V = 24\,\text{V}$. With this information we can use Eq. (2-28) to find that $K = \frac{24\,\text{V}}{7000\,\text{rpm}} \simeq 0.0327\,\text{V s/rad}$.

[2] https://www.superdroidrobots.com/shop/item.aspx/ig42-24vdc-right-angle-122-rpm-gear-motor-with-encoder/1831/

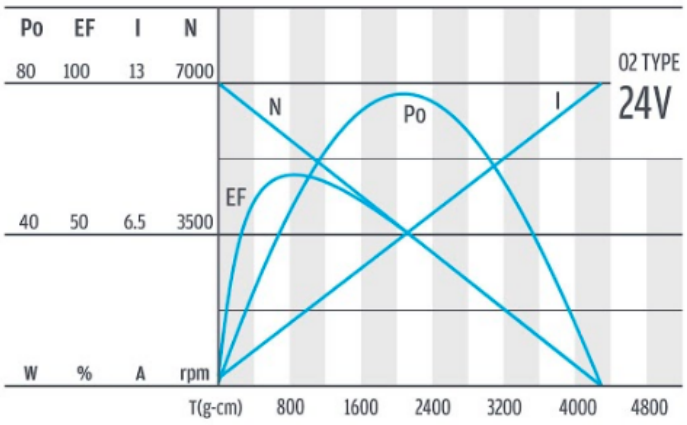


Figure 2-8: Figure from the motor datasheet
In this figure $N\dot{\varphi}$ is represented as N and τ_m/N as T.

2-4-2 Motor Friction Identification

We assume that $\tau = \tau_m - \tau_{fric}$ where τ is the total torque, τ_m is the torque produced by the motor and τ_{fric} is the torque produced by the friction in the motor, gears and transmission. Following a similar procedure as in [5], we try to approximate a model for τ_{fric} as a function of $\dot{\varphi}$ such as:

$$\tau_{fric} = a\,\dot{\varphi} + b\,\text{sign}\,(\dot{\varphi}) \tag{2-31}$$

To do so, we input a certain voltage to the motors and wait for them to reach a constant velocity. We measure this velocity using the data from the wheel encoders. Then, as the velocity is constant, there is no acceleration and therefore, no torque. This means that $\tau = 0$ and, therefore, $\tau_{fric} = \tau_m$. We can compute τ_m using Eq. (2-30).

We repeat this process for different voltages and then plot a graph of τ_{fric} versus $\dot{\varphi}$. We then use the non-linear least squares method to identify the parameters a and b of each motor. We can see a plot of the data and the fitted curves in Fig. 2-9. We can also find a table with the identified parameters in Table 2-4.

Wheel	a	b
1	0.0067	0.5373
2	0.0291	0.4540
3	0.0067	0.3423
4	0.0144	0.2641

Table 2-4: Identified variables for Eq. (2-31)

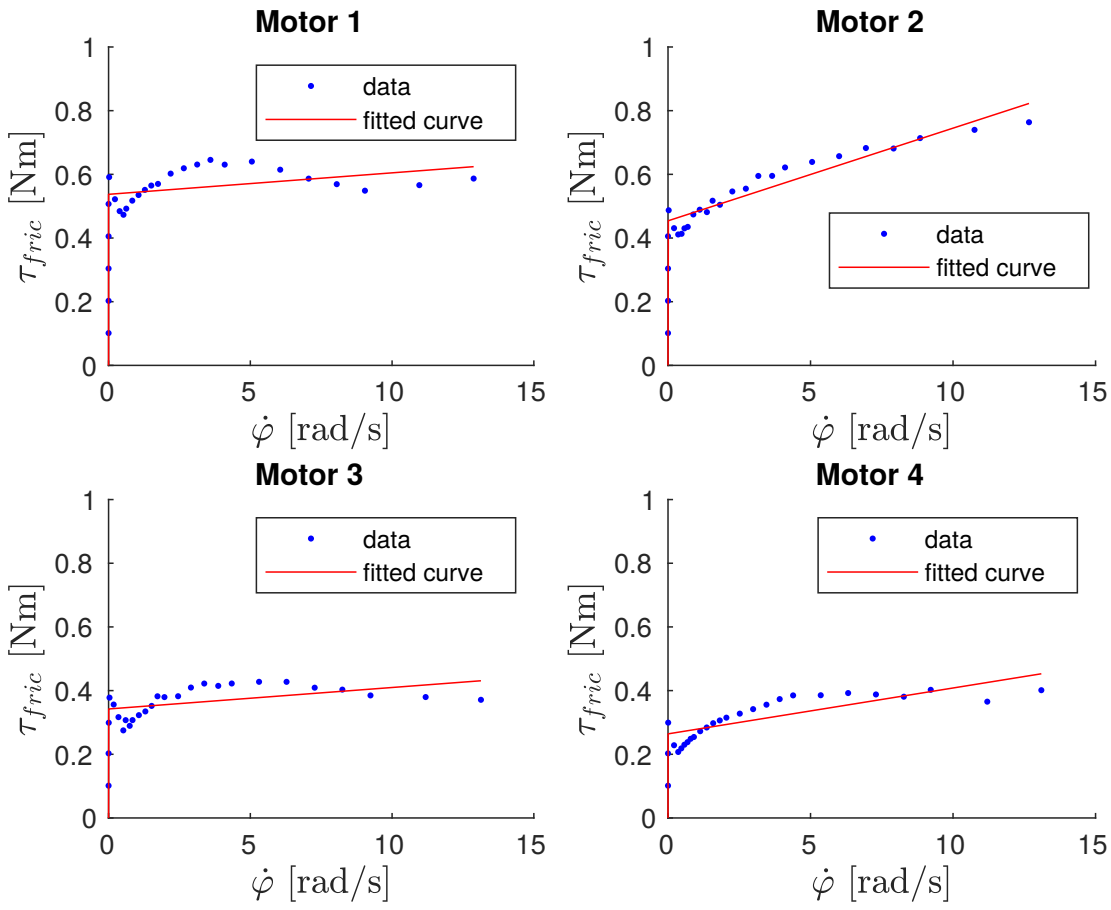


Figure 2-9: Identification of τ_{fric}

Chapter 3

Design of a Model Predictive Controller

3-1 Formal Definition of a Generic Problem

A Model-Predictive Control (MPC) problem is a constrained cost minimization problem for a system that evolves over time. MPC problems need to be discrete and only consider the first N stages of the problem. Every stage k is characterized by:

- $\boldsymbol{x}_k$: A state vector, where all the state variables of the system are stored.
- $\boldsymbol{u}_k$: An input vector, which contains all inputs to our system.
- $\boldsymbol{p}_k$: A parameter vector, which contains constant parameters for our system.

To fully define the problem we need to define the following functions:

- f: Inter-stage equalities, describes how the system evolves between each stage.
- C_k: Cost functions, evaluate the quality of each stage, the objective is to minimize the total cost.
- I: Constraints, a set of non-linear inequalities which our system needs to meet.

The formal definition of the MPC problem is:

$$\begin{array}{lll} \underset{u_1,u_2 \ldots u_N}{\operatorname{argmin}} & \sum_{k=1}^{N} C_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{p}_k) & \text{(Cost Function)} \\ \hline \text{s.t.} & \boldsymbol{x}_1 = \boldsymbol{x}_{init} & \text{(Inital condition)} \\ & \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{p}_k) & \text{(Inter-stage equalities)} \\ & I(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{p}_k) \geq \boldsymbol{0} & \text{(Constraints)} \end{array} \tag{3-1}$$

3-2 Solver Used

The MPC solver we used is FORCES PRO[1], made by the Swiss company Embotech. Free licences are available for researchers. In particular, we used the High-Level Interface (HLI)[2] of FORCES PRO, which is available in MATLAB and in Python. We used MATLAB to run it.

The FORCES PRO interface works in the following way: you generate a series of MATLAB structures which define your problem. These structures include the functions defined in the section above. With these structures, you call a MATLAB function which computes the derivatives of the functions C,f and I and then sends the information to a forces server and then downloads a compiled MATLAB solver to your computer. You can then execute this solver by passing it $\boldsymbol{x}_{init}$ and all $\boldsymbol{p}_k$ and it outputs a plan with all $\boldsymbol{u}_k$ and $\boldsymbol{x}_k$. The process of downloading a new solver takes a while, so it is important to put everything that might change in as a parameter in $\boldsymbol{p}_k$, so that if we change it we don't need to download a new solver.

As the solver tries to compute derivatives of the functions you pass it, you cannot use complex algorithms inside the functions, such as if-else structures, as they are not derivable. It is also a good practice to have functions that have easy derivatives.

The solver works iteratively, you input an initial guess and then the solver improves it each iteration until it reaches a good solution. The solver tries to find the optimal solution by looking at different branches, however it might only find a sub-optimal solution. The number of iterations that the solver will take to reach the final solution depends on the difficulty of the problem, the initial guess and the starting position. We can the solver by a maximum number of iterations (denoted as n_{maxit}).

Limiting the number of iterations might force the solver to stop looking for better solutions and settle for a solution that might not be optimal. Usually, the solutions found are still pretty good and permit good control of the system but there are some cases where the solver might not even be able to meet the inter-stage equalities. Luckily, the solver gives us a value for the equality error so that we can find out when this happens and correct it.

[1] https://www.embotech.com/forces-pro/
[2] https://forces.embotech.com/Documentation/high_level_interface/

3-3 Specification for our Problem

3-3-1 State variables

For our problem we have the following state variables:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q}_r \\ \dot{\boldsymbol{q}}_r \end{bmatrix} \qquad \boldsymbol{u} = \begin{bmatrix} V_i \\ \vdots \\ V_N \end{bmatrix}$$

Recall that $\boldsymbol{q}_r = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}$ and that V_i are the voltages at each of the motors.

The vector parameter $\boldsymbol{p}$ includes a lot of data, and will be explained as needed. In addition we also have a slack variable, which will be explained later.

3-3-2 Inter-Stage Equalities

Recall that in our case we can find $\dot{\boldsymbol{x}}$ by combining the solution obtained from the Lagrangian multipliers which models the mechanical behaviour of the system (Eq. (2-19)) and the model for the motor (Eqs. (2-30) and (2-31)).

However this describes a continuous solution instead of a discrete one, therefore, in order to use an MPC solver, we will need to discretize the problem with a fixed Δt. This discretization is done using the Explicit Runge-Kutta of 4th order (ERK4) method. In the ERK4 method we can give use n_{nodes} nodes between each MPC stage, in order to refine discretization while keeping the MPC solver simple.

When discretizing a continuous problem, each MPC stage represents a time in the future and therefore, the output of the solver can be understood as a plan for the future. This means that $N\Delta t$ will be the time horizon for our planner, the total time in the future that we consider in the plan. If we have a large time horizon, we will be able to plan for things that are further ahead in the future, but the solver will take more time to find the solution.

3-3-3 Constraints

Input limit

As the batteries of the robot run at 24 V, that is the maximum voltage we can provide to the motors. Therefore we must limit our inputs, using constraints:

$$-24\,\mathrm{V} \leq V_i \leq 24\,\mathrm{V} \qquad \forall i \in \{0 \dots N\}$$

Stay within limits

We assume that the workspace has a rectangular shape. We define its position by the vectors $\boldsymbol{x}_{lim}$ and $\boldsymbol{y}_{lim}$ in a way that $\begin{bmatrix} x & y \end{bmatrix}^T$ will be inside the workspace if $\boldsymbol{x}_{lim}(1) \leq x \leq \boldsymbol{x}_{lim}(2)$ and $\boldsymbol{y}_{lim}(1) \leq y \leq \boldsymbol{y}_{lim}(2)$. We also store $\boldsymbol{x}_{lim}$ and $\boldsymbol{y}_{lim}$ in the parameter vector $\boldsymbol{p}$.

We also assume that the robot can be contained within a rectangular bounding-box of width w and height h. These variables are stored in the parameter vector $\boldsymbol{p}$.

In order to ensure that the robot stays within the workspace, it is sufficient to add a constraint that forces the four edges of the robot's bounding-box stay within the workspace. That is:

$$\begin{aligned}
\boldsymbol{x}_{lim}(1) &\leq x + \frac{w}{2}\cos\psi + \frac{h}{2}\sin\psi \leq \boldsymbol{x}_{lim}(2) \\
\boldsymbol{x}_{lim}(1) &\leq x + \frac{w}{2}\cos\psi - \frac{h}{2}\sin\psi \leq \boldsymbol{x}_{lim}(2) \\
\boldsymbol{x}_{lim}(1) &\leq x - \frac{w}{2}\cos\psi - \frac{h}{2}\sin\psi \leq \boldsymbol{x}_{lim}(2) \\
\boldsymbol{x}_{lim}(1) &\leq x - \frac{w}{2}\cos\psi + \frac{h}{2}\sin\psi \leq \boldsymbol{x}_{lim}(2) \\
\boldsymbol{y}_{lim}(1) &\leq y + \frac{h}{2}\cos\psi - \frac{w}{2}\sin\psi \leq \boldsymbol{y}_{lim}(2) \\
\boldsymbol{y}_{lim}(1) &\leq y - \frac{h}{2}\cos\psi - \frac{w}{2}\sin\psi \leq \boldsymbol{y}_{lim}(2) \\
\boldsymbol{y}_{lim}(1) &\leq y - \frac{h}{2}\cos\psi + \frac{w}{2}\sin\psi \leq \boldsymbol{y}_{lim}(2) \\
\boldsymbol{y}_{lim}(1) &\leq y + \frac{h}{2}\cos\psi + \frac{w}{2}\sin\psi \leq \boldsymbol{y}_{lim}(2)
\end{aligned}$$

Avoiding obstacles

We represent the obstacles as ellipses with position $\boldsymbol{q}_{obs}$ and radii $\boldsymbol{dim}_{obs}$. These variables are stored in the parameter vector $\boldsymbol{p}$. Here, as the obstacles may be moving, the position $\boldsymbol{q}_{obs}$ will be different in every stage, so each $\boldsymbol{p}_k$ contains the expected position of the obstacle in the time corresponding to the stage k.

As it can be seen in [6], finding a simple formula for the distance between an ellipse and a point is difficult, but if we only want to check if the point is inside or outside the ellipse one can use the following formula for the distance and check its sign.

$$d(x,y) = \left(\frac{x - x_{obs}}{\boldsymbol{dim}_{obs}(1)}\right)^2 + \left(\frac{y - y_{obs}}{\boldsymbol{dim}_{obs}(2)}\right)^2 - 1$$

The analogous function for this distance between our robot and the obstacle will be the minimum distance of the four segments that compose the bounding-box. To compute the distance between a segment and an ellipse we use the same algorithm as in Appendix C of [7].

Softening the constraints

With these constraints in place the system is now able to avoid collisions. However, sometimes a collision is unavoidable, for example when an object moves unexpectedly and steps in the way of the robot. In these cases if we have these constraints in place the solver will stop working, as it will think all solutions are unfeasible. If the solver stops working, the robot will keep going and, eventually, crash without even trying to slow down. To solve this we need to soften the constraints.

Softening the constraints consists of using slack variables and if we have an inequality constraint of the form $I(\boldsymbol{x}) \geq 0$ we transform it as such:

$$I(\boldsymbol{x}) \geq 0 \rightarrow \begin{cases} I(\boldsymbol{x}) + slack \geq 0 \\ slack \geq 0 \end{cases}$$

This will mean that by increasing $slack$, the system will be able to go to values of $I(\boldsymbol{x})$ below zero, breaking the inequality. To ensure that this is not done unless it is completely necessary, we will add a big cost to this slack variable. The cost will have to be magnitudes of order bigger than other costs.

3-3-4 Cost Function

Recall that the cost function C is defined as the sum of all stages' individual cost function C_k. We will generate these cost function as a sum of different costs. Each separate cost will have an associated weight which are added as as parameters in the parameter vector $\boldsymbol{p}$ so that they can be easy to tweak without having to get a new solver.

Distance to objective

The main goal of the system is to get to a certain position. In order to achieve that we add the objective pose, $q_{end} = \left[x_{end}, y_{end}, \psi_{end}\right]^T$, as an additional parameter in the parameter vector $\boldsymbol{p}$. The cost function will be the distance from the current pose to the objective.

As the pose of the system includes both the position and the angle, two different weights need to be used, one for the positions and another for the angle.

However, as we mentioned earlier, it is important to keep the derivatives of the cost function easy. To achieve this we use the square of the distance instead of the actual distance.

While doing the simulated experiments, we saw that the solver ran much faster and resulted in better overall results if this cost was only applied to the final stage. Taking all of this into account we have:

$$C_{k,dest} = \begin{cases} W_{pos}(x_{end} - x_R)^2 + W_{pos}(y_{end} - y_R)^2 + W_{ang}(\psi_{end} - \psi_R)^2 & \text{if k=N} \\ 0 & \text{otherwise} \end{cases}$$

Electrical energy consumed by the motors

Sometimes the system will reach a state where there are multiple solutions that get to the same distance to the objective. The system can also be in a state where the system is already next to the objective. In these cases the final stage of all solutions will have the same distance to objective. And the cost above will be equal for all solutions. We can therefore add another cost so that the system can choose which of these similar solutions to perform.

A good way to decide between the similar solutions is to choose the one which uses the least energy. To calculate the total energy consumed by the plan one can compute the sum of the power of all motors at each stage and multiply it by Δt. Power is equal to the voltage times the intensity, from Eq. (2-28) we get that:

$$P = V\, i = V\, \frac{V - NK\dot{\varphi}}{R}$$

$$C_{k.power} = W_{power} \sum_{i=0}^{4} P_i\, \Delta t = W_{power} \sum_{i=0}^{4} V_i\, \frac{V_i - NK\dot{\varphi}_i}{R}\, \Delta t \qquad \forall k \in \{0 \dots N\}$$

It is important to note that if the weight for this cost is too high, the system will prefer to not do anything at all in order to save energy. Therefore the weight of this should be smaller than the distance weight.

Slack minimization

As we mentioned earlier, slack variables let the system break the inequalities and we need to enforce that this is done only when absolutely necessary by adding a cost for the slack variables with an extremely large weight.

$$C_{k,slack} = W_{slack}\, slack^2 \qquad \forall k \in \{0 \dots N\}$$

Chapter 4

Control Algorithm and Setup

4-1 Simulated Execution

In order to do real time planning of the system and take into account unexpected movement, one can start with the following simple simulation control algorithm, shown here in pseudo-code:

Algorithm 1 Control Algorithm

1: $\boldsymbol{x} \leftarrow \boldsymbol{x}_{init}$
2: **while** true **do** ▷ Limited to run every Δt
3: $external\_data \leftarrow$ GetExternalData()
4: $\{\boldsymbol{p}_1 \dots \boldsymbol{p}_N\} \leftarrow$ GetParameters($\boldsymbol{x}, external\_data$)
5: $\{\boldsymbol{u}_1 \dots \boldsymbol{u}_N\}, \{\boldsymbol{x}_1 \dots \boldsymbol{x}_N\} \leftarrow$ MPCSolver($\boldsymbol{x}, \{\boldsymbol{p}_1 \dots \boldsymbol{p}_N\}$)
6: $\boldsymbol{x} \leftarrow \boldsymbol{x}_2$
7: **end**

First, the algorithm assumes a starting position (Line 1). Then, we start the control loop which gets the parameters using external data (Lines 3 and 4) and then call the Model-Predictive Control (MPC) solver (Line 5). As we explained earlier, the MPC solver outputs a plan for the future. We assume this plan is followed perfectly and instantly and assume that the system reaches the second stage of the plan (Line 6).

As we can see the system is re-planning every iteration. This means that the time taken by the solver (t_{solve}) will have to be limited so as not to take longer than Δt. However, as explained before, we can only limit the solver by the number of iterations. In [8] it was found that the time per iteration per N $\left(h = \frac{t_{solve}}{n_{iterations}\, N}\right)$ was very stable and could be considered a constant that depends on the complexity of the problem. For this problem we found that $h \simeq 10^{-5}$ and therefore we can set the maximum number of iterations as follows so as to guarantee $t_{solve} < \Delta t$:

$$n_{maxit} = \left\lfloor \frac{\Delta t}{10^{-5} * N} \right\rfloor \tag{4-1}$$

However, as we explained in Section 3-2 limiting the number of iterations means that the solution might not satisfy the inter-stage equalities. This would mean that $\boldsymbol{x}_2$ might not be the correct integration of our system and Line 6 of the algorithm might make our system behave unexpectedly. To correct this we can substitute Line 6 by our discretization of the dynamics of the system.

4-2 Visual Interface

A visual interface has been designed for the control algorithm. It can be seen in Fig. 4-1. In it you can see the following data:

(1): Current position of the system.

(2): Starting position of the problem.

(3): Objective of the problem.

(4): A table with the data above but presented numerically. This table can be edited to change the parameters.

(5): Current plan of the Model Predictive Controller.

(6): Current position of the obstacles.

(7): A table with some editable parameters for the controller, such as the weights of the cost functions.

(8): A switch to turn on or off the controller.

(9): A dropdown menu where one can select pre-built scenarios.

(10): A "Reset" button that moves the system back to its starting position.

(11): A "Go to static" button that makes all the obstacles static.

(12): A "Switch goal" button which switches the starting and ending positions.

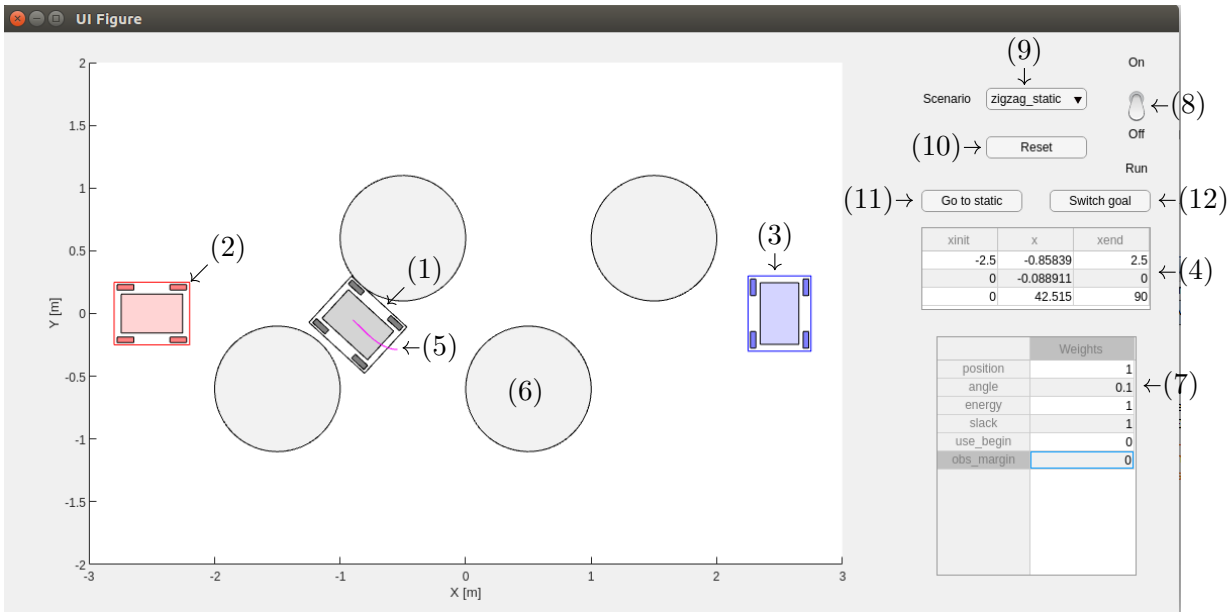


Figure 4-1: Visual interface screenshot, contents are described above

The process of drawing and updating this visual interface can be time-consuming. We need to separate the controller from this process so that we do not interfere with the efficiency of the controller. In order to do so, we coded both of them as separate MATLAB scripts.

4-3 Communication Between the Visual Interface and the Controller

The communication between them is done through a framework called Robot Operating System (ROS)[1]. This framework establishes communication protocols which can be called from many different operating systems and coded with multiple programming languages. The main benefit of using ROS is that the communication can be done through the internet. Therefore, we could have the controller and the visual interface in separate computers. The controller could even be run on an on-board computer.

To communicate through ROS using MATLAB we used MATLAB's Robot Operating Toolbox [2].

Using ROS also allows us to use rosbags. Rosbags let us record all the data that goes through ROS and then replay it. We can even parse this data with MATLAB to analyze it.

[1] http://www.ros.org
[2] https://www.mathworks.com/products/robotics.html

4-4 Steps Towards a Real Experiment

Although we couldn't reach our goal of controlling the real robot due to time-constraints there are several steps that have been taken towards that goal.

4-4-1 Arduino Control

The on-board computer that came with the robot was an Arduino board that took commands from a radio emitter and translated those signals to inputs for the wheels. In order to be able to control the robot from a remote computer we needed to find a different way of communication. We therefore added a WiFi board (ESP-01) to the Arduino.

To send the commands through WiFi we wrote Arduino code that created an artificial WiFi network and listened for commands. The commands are sent from our MATLAB code, which encodes the voltage inputs for the four motors as 4-byte vectors (one byte for each motor). The Arduino board then decodes those inputs and sends actuates the motors accordingly.

We even designed and 3D-printed a case for the WiFi board so that we could screw it into the robot. We also soldered some cables to the WiFi board to be able to connect it to the Arduino board. The 3D-printed case and the soldered cables can be seen in Fig. 4-2.

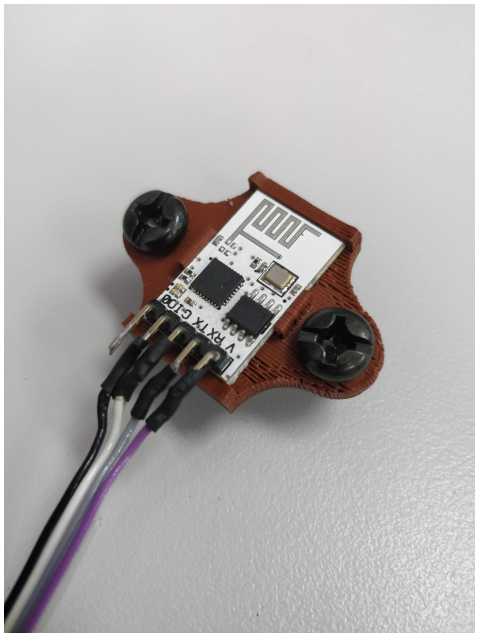


Figure 4-2: WiFi board with 3D-printed case and soldered cables

4-4-2 Capturing Real Data

In order to use real world data, the intention was to use a Motion Capture System (MCS). This would permit us to get fast and accurate readings of the position and the orientation of the robot in 3D space. There is a MCS installed in Institut de Robòtica i Informàtica Industrial (IRI) made by the company OptiTrack. There are ROS libraries for OptiTrack which would have made interfacing this data with our setup very easy.

To smooth out the noise in the data coming from the MCS we could use Kalman Filters, as done in [7].

Chapter 5

Results

We will now demonstrate some maneuvers that our system is able to calculate. A video of these maneuvers can be seen in https://youtu.be/WgyKAvjbOlk.

5-1 Free Movement

We first introduce a simple scenario where the robot simply needs to move sideways from one point to another without changing orientation and without any obstacles. We have

$$\boldsymbol{q}_{start} = \begin{bmatrix} 2 \\ 0 \\ \pi/2 \end{bmatrix} \text{ and } \boldsymbol{q}_{end} = \begin{bmatrix} -2 \\ 1 \\ \pi/2 \end{bmatrix} \tag{5-1}$$

In Fig. 5-1 one can see several steps of the trajectory that our system goes through.

In Fig. 5-1a we see how the system starts moving towards the objective, it starts by just doing sideways movement. This is a good way of accelerating as sideways movement requires all four wheels (See Fig. 1-1) to apply force. As all wheels are applying force, the system is able to accelerate faster this way.

In Fig. 5-1b the robot starts to change its orientation slightly until it starts going almost in diagonal movement in Fig. 5-1c. Diagonal movement only needs two wheels to be actuated and it does not generate any opposing forces. Thus, it is a much more energy-efficient way of moving. The controller has decided to move this way because it has already accelerated to its maximum speed and is now trying to minimize the total energy consumption.

In Fig. 5-1d the system is close to its objective and is therefore starting to slow down and going back to the desired orientation.

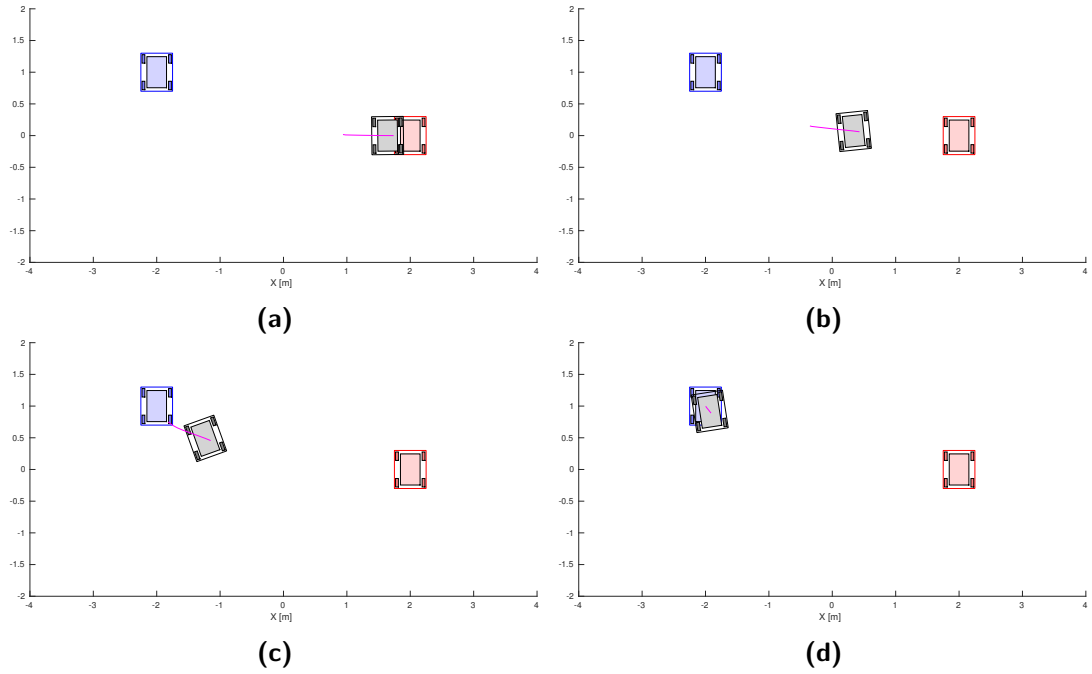


Figure 5-1: Four snapshots of the system moving freely
Plotted with the same rules as the ones explained in Section 4-2 (Visual Interface)

5-2 Narrow Corridor Problem

To demonstrate how our system manages obstacle avoidance, we introduce a scenario where the robot has to weave through four large obstacles. The gap between the obstacles is bigger than one side of the robot but smaller than the other side. This will force the robot to change its orientation in order to fit through the gaps between the obstacles. In Fig. 5-2 one can see several steps of the trajectory that our system goes through.

In Fig. 5-2a we see how the system starts moving by going slightly upwards to avoid the first obstacle. And then in Fig. 5-2b it manages to go through the first gap easily as it was already in a good orientation to go through it.

In Fig. 5-2c the system reaches the second gap, but it does so in the wrong orientation. In order to overcome this, the system is planning to rotate itself in order to fit through the second gap. We see this happen in Fig. 5-2d.

In Fig. 5-2e the system manages to overcome the third obstacle in a similar way and then it moves towards the goal in Fig. 5-2f.

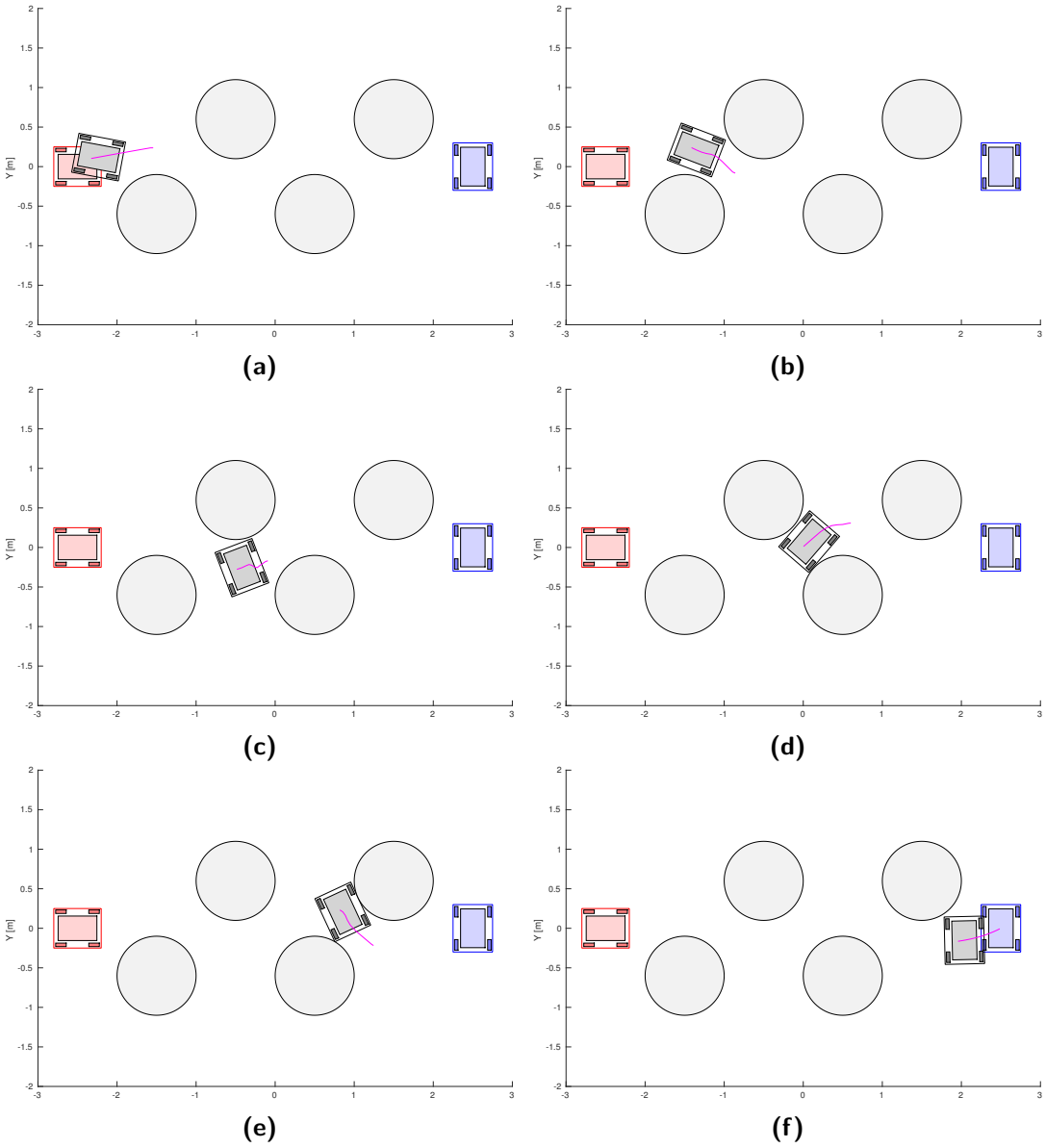


Figure 5-2: Six snapshots of the system going through a narrow corridor
Plotted with the same rules as the ones explained in Section 4-2 (Visual Interface)

5-3 Moving Obstacle Avoidance

To demonstrate the ability to work with moving obstacles we designed a scenario similar to the previous one but with the obstacles moving vertically across the path of the robot.

In Fig. 5-3a we see the system reacting to the first moving obstacle by deciding to go under it and it does so without problem. In Fig. 5-3b the system is stuck between two obstacles but predicts that the second obstacle will continue moving upwards creating a gap under it. It therefore plans to go through the gap under the second obstacle and does so successfully.

In Fig. 5-3c we see how the system plans to go under the third obstacle in Fig. 5-3dbut sees it will not be able to do so and stops before colliding with the obstacle. In Fig. 5-3e it has managed to find a way to go above the third obstacle.

In Fig. 5-3f we see how the system is able to avoid the fourth obstacle and then moves freely towards the objective.

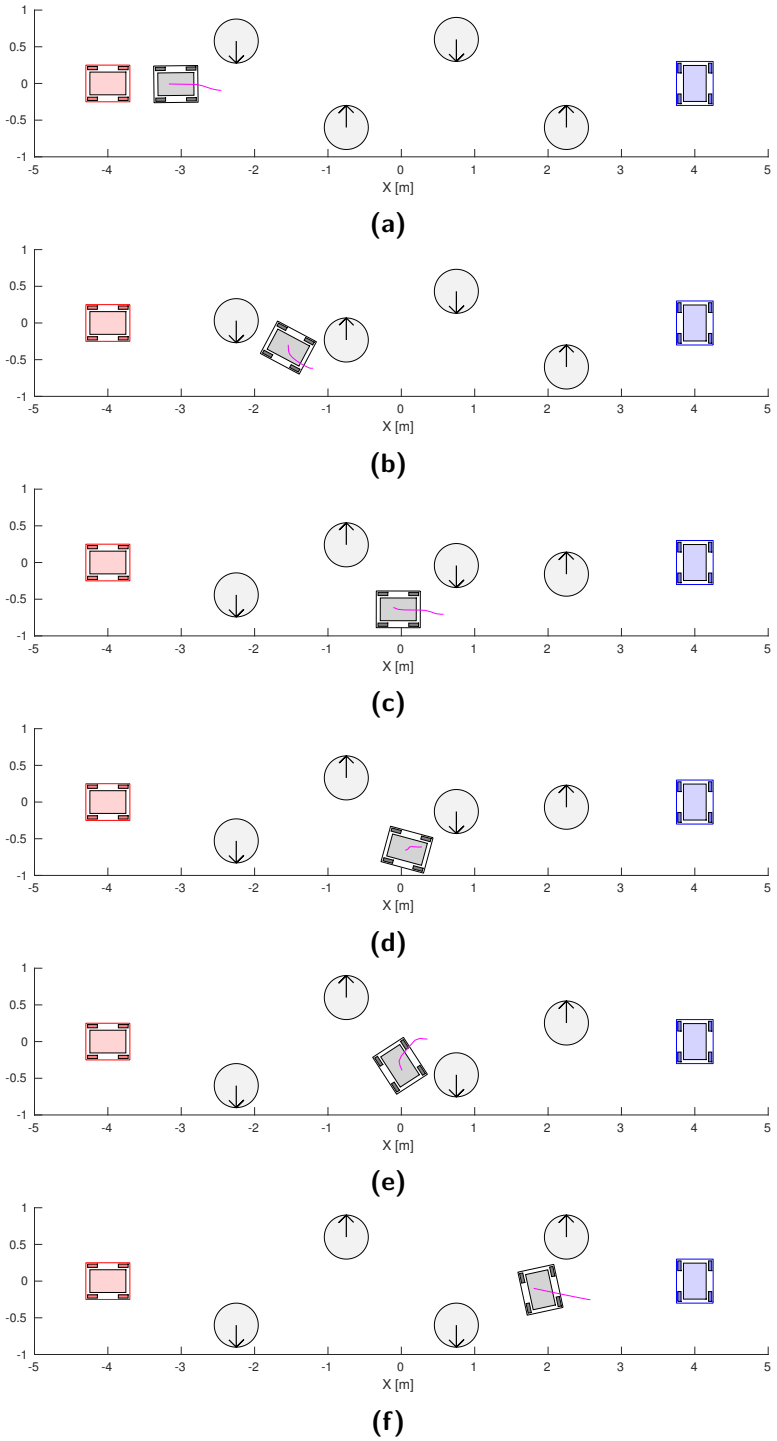


Figure 5-3: Six snapshots of the system going through moving obstacles
Plotted with the same rules as the ones explained in Section 4-2 (Visual Interface)
Black arrows have been added to show the direction of movement of each obstacle.

Chapter 6

Conclusion

In this thesis we developed a dynamic model for the movement of a mecanum-wheeled robot that takes into account the robot's geometry and the dynamic response of its actuators. We then used a model predictive controller to control the system. This controller is able to generate maneuvers dynamically even with uncertainties in the environment. Therefore, it is able to adapt to unexpected changes such as moving obstacles and inaccuracies in the model. The controller has then been demonstrated to be able to achieve real-time control in simulated scenarios.

6-1 Future Studies

For future studies we recommend the following:

- Testing the controller on the real robot by finishing the work started in Section 4-4.
- Adding aerodynamic drag to the dynamic equations of the motor.
- Adding more complex objectives such as moving goals.
- Testing a wider variety of parameters for the solver.
- An on-board controller could be added to rid the system of an external computer.
- On-board telemetry could be used so that the robot could locate itself and the obstacles. This would rid the controller from the need of a Motion Capture System (MCS).
- By using Kalman filters one could create an algorithm that is able to detect errors in the parameters of the model (such as the inertia or the motor friction coefficients) and adjusts their values on the go.

Bibliography

[1] P. Corke, *Robotics, Vision and Control*. 2011.

[2] K. M. Lynch and F. C. Park, *Modern Robotics Mechanics , Planning , and Control*. 2017.

[3] F. Kuenemund, D. Hess, and C. Roehrig, "Motion controller design for a mecanum wheeled mobile manipulator," *1st Annual IEEE Conference on Control Technology and Applications, CCTA 2017*, vol. 2017-Janua, no. August, pp. 444–449, 2017.

[4] Z. Hendzel and Rykała, "Modelling of dynamics of a wheeled mobile robot with mecanum wheels with the use of lagrange equations of the second kind," *International Journal of Applied Mechanics and Engineering*, vol. 22, no. 1, pp. 81–99, 2017.

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

[6] L. Maisonobe, "Quick computation of the distance between a point and an ellipse," no. September 2003, pp. 1–14, 2006.

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

[8] I. Moreno, *Planning and Control of a Multiple-Quadcopter System Cooperatively Carrying a Slung Payload in Dynamical Environments*. Bachelor thesis, 2018.

Glossary

List of Acronyms

FME	Facultat de Matemàtiques i Estadística
IRI	Institut de Robòtica i Informàtica Industrial
MPC	Model-Predictive Control
ERK4	Explicit Runge-Kutta of 4th order
MCS	Motion Capture System
ROS	Robot Operating System
COM	center of mass
ENU	East-North-Up
HLI	High-Level Interface