Planning throwing motions for mobile manipulators

Borvayeh, Sina Gravina, Giovanbattista Nunziante, Luca

March 2023

Abstract

The aim of this project is to plan a throwing motion for a planar mobile manipulator (MM), in an attempt to increase its workspace or reduce the time needed for a pick and place task. The motion planning problem is formulated as an Optimal Control Problem (OCP) and solved using numerical optimization via the Optimization Toolbox of MATLAB. Robot balance is guaranteed along the whole planned trajectory using an appropriate nonlinear constraint based on the MM full dynamics, ensuring non-negative moments around the edges of the support polygon. A first set of simulations show the results obtained via different approaches to the optimization framework, and then aggressive manoeuvres are analyzed.

1 Introduction

Mobile manipulators are robotic systems that consist of a mobile base and a manipulator arm. Thanks to their versatility they are effectively used in various areas such as logistics, due to to their augmented workspace, and disaster response since they combine high dexterity and mobility and can be designed to traverse hazardous environments. However, one of the main challenges in designing MMs is achieving balance: due to their structure, the center of gravity and the ZMP of the system constantly change as the robot moves, and this clearly affects the robot stability, leading to a possible tip over and potential damage to the system or harm to the environment and people around it.

Research in planning throwing motion is practically motivated, since this kind of dynamic manipulation can increase the working range and shorten an operation cycle time. Although finding a throwing state (joints position and velocity) for a MM is a non-injective problem, estimating a kinodynamically feasible throwing trajectory that also satisfies the balance constraint is not trivial. Moreover, when dealing with arbitrary objects one should also consider that the grasp of the object, its mass distribution and aerodynamics heavily influence its trajectory. This has been successfully addressed with model-free methods [5]. In this work we will consider to throw a ball modeled as a point

mass, neglecting its rotation. This implies that after its release, the ball will experience planar ballistic flight, as in [1].

Another approach to throwing problems is using sampling based methods as in [6], where they sample a state that sends the object on a ballistic arc to the target, and reach it via probabilistic bidirectional planners from the initial and final state, naturally dividing the task in a throwing and stopping phase.

Generation of throwing motion has also been formulated as an OCP [2]: in optimal control theory, the fields of optimization and automatic control are combined to compute optimal control inputs to the system that is being controlled. In general, a local solution to a Non Linear Program (NLP)[1] is found using iterative schemes. The major difference between different iterative solvers is how the search direction and step length are computed. When computing local solutions to general NLPs, there are mainly two families of optimization methods that are used: Sequential Quadratic Programming (SQP) methods and nonlinear Interior-Point (IP) methods — in this work the former will be used.

We investigate various approaches to the NLP formulation and the respective results are shown in the simulations. Comparisons on the planned throwing trajectories are carried out, analyzing for each motion: relevant robot configurations, ZMP position, velocities and control inputs profiles. Finally, we further discuss the influence of the balance constraint by performing aggressive maneuvers.

The following work is organized as follows. The considered problem is formulated in Section 2. In Section 3 the dynamic model of the mobile manipulator (MM) is presented, along with the equations of motion and the balance criterion. In Section 4 the approach to solve the motion planning problem are detailed. Some simulations are presented in Section 5.

2 Problem formulation

Consider a mobile manipulator operating in a 2-dimensional (vertical) workspace $\mathcal{W}$, that is required to transport an object from an initial to a final goal position $\boldsymbol{p}_{goal}$. We consider a case in which the robot is unable to reach the intended final position due to structural limitations (e.g. obstacles blocking the path) or it is desirable to reduce the time of the operation. For this reason, the robot is called to release the object from a distance, rather than delivering it to the intended goal.

Ballistic motion We assume the thrown object follows a planar ballistic trajectory. Given a world frame $\mathcal{F}_w$ as in Fig. 1, the equations that describe the object motion are

$$\begin{cases} x(t) = & \dot{x}_0 t + x_0 \\ z(t) = & \dot{z}_0 t - \frac{1}{2} g t^2 + z_0 \end{cases} \tag{1}$$

[1] A NLP is an optimization problem where at least one of the involved functions is nonlinear.

where $\boldsymbol{p}_0 = (x_0, z_0)$ is the position of the point-mass object at the initial instant of the ballistic motion and $\dot{\boldsymbol{p}}_0 = (\dot{x}_0, \dot{z}_0)$ is its velocity; $g = 9.81\ m/s^2$ is the gravity acceleration and t is the time variable. Assuming, without loss of generality, that the desired final object position lies on the ground, i.e., $\boldsymbol{p}_{goal} = (x_{goal}, 0)$, from (1) we can write

$$-x_{goal} + x_0 + \frac{\dot{x}_0 \dot{z}_0}{g} + \dot{x}_0 \sqrt{\frac{\dot{z}_0^2}{g^2} + \frac{2z_0}{g}} = 0 \tag{2}$$

Assuming that at time instant t_{init} the robot is in its initial configuration $\boldsymbol{q}_{init}$ at rest with the object attached to its end-effector, we wish to plan a motion that:

R1: starts from the robot initial state $\boldsymbol{x}_{init} = (\boldsymbol{q}_{init}, \boldsymbol{0})$ and ends to a final resting state $\boldsymbol{x}_{final}$ at time t_{final}, while it is such that there exists a time instant $t_{rel} \in [t_{init}, t_{final}]$ at which, if released, the object can reach its goal position $\boldsymbol{p}_{goal}$;

R2: is kinodynamically feasible, in the sense that it is consistent with the robot dynamic model and respects existing constraints on joint and velocity limits and input torque bounds;

R3: preserves the robot balance, i.e., all wheels maintain contact with the ground.

3 Modeling

We consider a planar (vertical) MM consisting of one driving wheel and two caster wheels, as reported in Fig. 1, and a 2R manipulator mounted on top. For the actuated wheel we assume no slippage, thanks to the wheel-ground friction, while the two caster wheels only contribute to the robot balance. Moreover, we assume all of them to be in point contact with the ground. Being a fully planar problem, the mobile base can only move forward and backward, i.e. it is not subject to any non-holonomic constraint.

Following [4], to represent the pose of the mobile base we use 3 fictitious joints that connect the world frame $\mathcal{F}_w$ to a frame attached to the point of contact of the driving wheel of the robot with the ground ($\mathcal{F}_2$ in Fig. 1). The fictitious joints are arranged with the first two being prismatic and along the world axes x_w, z_w and a revolute one along y_w, as depicted in Fig. 1. So, the robot configuration is $\boldsymbol{q} = (x_b, z_b, \theta_y, q_1, q_2) \in \mathbb{R}^5$ where the first three components represent the base pose, and the last two the arm configuration.
To derive the dynamic model of this MM, we take the Lagrangian approach [3] and compute the positions and velocities of the Center of Mass (CoM) of each body using the Denavit-Hartenberg (DH) frames of Fig. 1, and the associated Table 1.

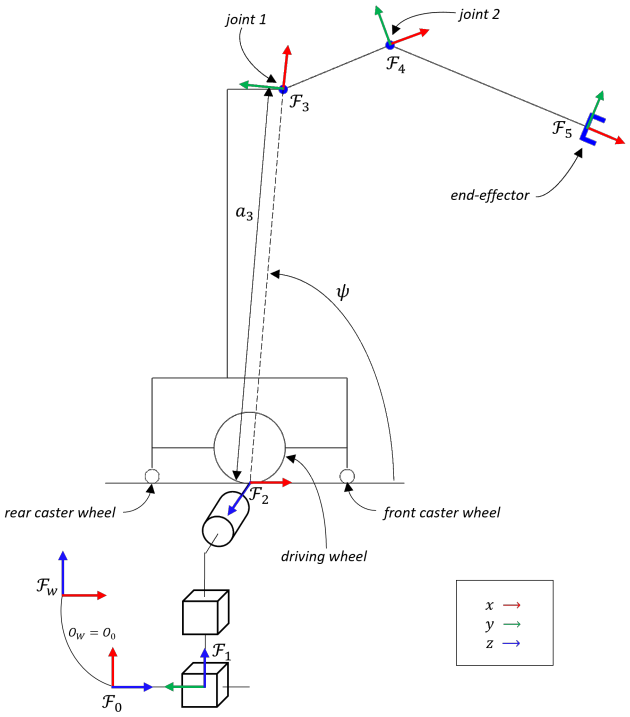


Figure 1: DH frames attached to the robot to derive its dynamic model.

The obtained robot dynamics is

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{S}(\boldsymbol{q})\boldsymbol{u} + \boldsymbol{A}(\boldsymbol{q})\boldsymbol{\lambda} \tag{3}$$

where $\boldsymbol{B}(\boldsymbol{q}) \in \mathbb{R}^{5\times5}$ is the inertia matrix, $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^5$ encompasses the Coriolis, centrifugal and gravitational terms, $\boldsymbol{S}(\boldsymbol{q}) \in \mathbb{R}^{5\times3}$ is the matrix that maps the control inputs $\boldsymbol{u} \in \mathbb{R}^3$ to forces performing work on the generalized coordinates, and the matrix $\boldsymbol{A}(\boldsymbol{q}) \in \mathbb{R}^{5\times2}$ is used to express at the generalized coordinates level the contact forces $\boldsymbol{\lambda} \in \mathbb{R}^2$.

In the case under consideration:

- $\boldsymbol{S}(\boldsymbol{q})$ is constant, and since the inputs $\boldsymbol{u}$ are the torque of the wheel and of the two manipulator joints, it is

$$\boldsymbol{S} = \begin{bmatrix} \frac{1}{r_w} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  where r_w is the driving wheel radius.

i	a_i	α_i	d_i	θ_i
1	0	$-\pi/2$	x_b	$-\pi/2$
2	0	$\pi/2$	z_b	$-\pi/2$
3	a_3	0	0	$\theta_y + \psi$
4	h_1	0	0	$q_1 - \psi$
5	h_2	0	0	q_2

Table 1: DH table associated to the frames in Fig. 1 where h_1 and h_2 are the manipulator link lengths.

- due to the robot motion on the horizontal ground, the coordinates z_b, θ_y — associated to the second and third fictitious joints — are constrained to a constant value (zero for the choice of θ_3 in Table 1), therefore $\boldsymbol{h}(\boldsymbol{q}) = \begin{pmatrix} z_b \\ \theta_y \end{pmatrix} = \boldsymbol{0}$ and $\boldsymbol{A}(\boldsymbol{q}) = (\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}})^T \in \mathbb{R}^{5\times 2}$.

3.1 Equations of motion

We can split $\boldsymbol{q}$ in $\boldsymbol{q}_f = (z_b, \theta_y)$ and $\boldsymbol{q}_r = (x_b, q_1, q_2)$ and have 2 selection matrices so that $\boldsymbol{q}_f = \boldsymbol{Q}_f \boldsymbol{q}$ and $\boldsymbol{q}_r = \boldsymbol{Q}_r \boldsymbol{q}$. Since $\ddot{\boldsymbol{q}}_f = \boldsymbol{0}$ it is $\ddot{\boldsymbol{q}} = \boldsymbol{Q}_r^T \ddot{\boldsymbol{q}}_r$, and by left-multiplying (3) by $\boldsymbol{Q}_r$ we have

$$\boldsymbol{Q}_r \boldsymbol{B}(\boldsymbol{q}) \boldsymbol{Q}_r^T \ddot{\boldsymbol{q}}_r + \boldsymbol{Q}_r \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{Q}_r \boldsymbol{S}(\boldsymbol{q}) \boldsymbol{u} \tag{4}$$

Note that in our work $\boldsymbol{Q}_r \boldsymbol{A}(\boldsymbol{q}) \boldsymbol{\lambda} = \boldsymbol{0}$ since $\boldsymbol{h}(\boldsymbol{q})$ does not depend on $\boldsymbol{q}_r$.

Starting from (4) and with some manipulation — following [3] — we can derive the robot state space model. Indeed, it is particularly simple in our case due to the absence of non-holonomic constraints. Given the state $\boldsymbol{x} = (\boldsymbol{q}_r, \dot{\boldsymbol{q}}_r)$, we have

$$\dot{\boldsymbol{x}} = \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{u}) = \begin{pmatrix} \dot{\boldsymbol{q}}_r \\ \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{E}\boldsymbol{u} - \boldsymbol{m}(\boldsymbol{q}, \dot{\boldsymbol{q}})) \end{pmatrix} \tag{5}$$

where

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{q}) &= \boldsymbol{Q}_r \boldsymbol{B}(\boldsymbol{q}) \boldsymbol{Q}_r^T \ , \\ \boldsymbol{m}(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \boldsymbol{Q}_r \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \ , \\ \boldsymbol{E} &= \boldsymbol{Q}_r \boldsymbol{S} \ . \end{aligned}$$

3.2 Contact forces

To analyze the robot balance, it is sufficient to know the sum of the forces orthogonal to the ground and the moments parallel to the ground that the robot exerts.

Left multiplying (3) by $\boldsymbol{Q}_f$ we get

$$\boldsymbol{Q}_f \boldsymbol{B}(\boldsymbol{q}) \boldsymbol{Q}_r^T \ddot{\boldsymbol{q}}_r + \boldsymbol{Q}_f \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\lambda} \tag{6}$$

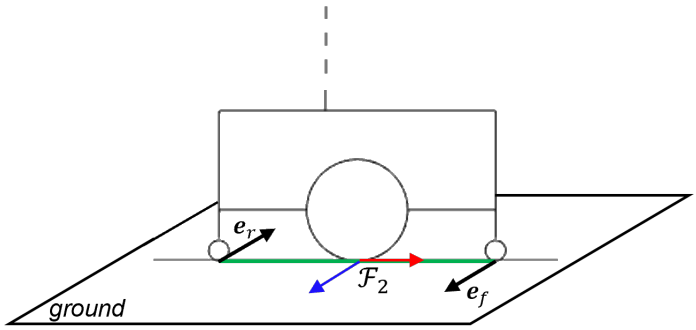


Figure 2: In green the robot support polygon.

where $\boldsymbol{Q}_f \boldsymbol{S}(\boldsymbol{q})\boldsymbol{u} = \boldsymbol{0}$ since the control torques do not perform work on $\boldsymbol{q}_f$. Note that $\boldsymbol{Q}_f \boldsymbol{A}(\boldsymbol{q})$ in (6) is the identity matrix, hence $\boldsymbol{\lambda}$ represents the contact force and moment along the axes of the second and third fictitious joints.

Let f_{f_2}, μ_{f_3} be the force and moment that the robot exerts along the z-axis of frame $\mathcal{F}_1, \mathcal{F}_2$ respectively (see Fig. 1), then

$$\begin{pmatrix} f_{f_2} \\ \mu_{f_3} \end{pmatrix} = -\boldsymbol{\lambda} \ . \tag{7}$$

Given the expression of $\ddot{\boldsymbol{q}}_r$ from (5), we can plug it in (6) and get f_{f_2}, μ_{f_3} as function of the state and inputs, i.e.

$$\begin{aligned} \begin{pmatrix} f_{f_2} \\ \mu_{f_3} \end{pmatrix} = & - \boldsymbol{Q}_f \boldsymbol{B}(\boldsymbol{q}) \boldsymbol{Q}_r^T \boldsymbol{M}^{-1}(\boldsymbol{q}) \boldsymbol{E}\boldsymbol{u} \\ & + \boldsymbol{Q}_f \boldsymbol{B}(\boldsymbol{q}) \boldsymbol{Q}_r^T \boldsymbol{M}^{-1}(\boldsymbol{q}) \boldsymbol{m}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ & - \boldsymbol{Q}_f \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \ . \end{aligned} \tag{8}$$

Then, we consider the orthogonal to the ground force and parallel to ground moment exerted by the robot at the origin of $\mathcal{F}_2$ and express them in $\mathcal{F}_2$ as

$$\begin{aligned} {}^2\boldsymbol{f} = \boldsymbol{f} = {}^2\boldsymbol{R}_w(\boldsymbol{q}) f_{f_2} {}^w\hat{\boldsymbol{z}}_1 \\ {}^2\boldsymbol{\mu} = \boldsymbol{\mu} = {}^2\boldsymbol{R}_w(\boldsymbol{q}) \mu_{f_3} {}^w\hat{\boldsymbol{z}}_2 \end{aligned} \tag{9}$$

where ${}^2\boldsymbol{R}_w(\boldsymbol{q})$ is the rotation matrix of $\mathcal{F}_2$ w.r.t. $\mathcal{F}_w$, and ${}^w\hat{\boldsymbol{z}}_1, {}^w\hat{\boldsymbol{z}}_2$ are the unit vectors of the z-axis of $\mathcal{F}_1$ and $\mathcal{F}_2$ expressed in $\mathcal{F}_w$.

3.3 Balance constraint

The evaluation of the robot balance is done considering the moments applied by the robot around the support polygon edges: balance is preserved if the resulting moments are non-negative.

Being a 2-dimensional problem, the support polygon reduces to the segment joining the contact points of the two caster wheels, and the two edges collapse into the contact points of the rear and front wheel, as shown in Fig. 2.

We express in $\mathcal{F}_2$ two unit vectors $\boldsymbol{e}_r$, $\boldsymbol{e}_f$ orthogonal to the motion plane, and denote their starting point — the passive wheel contact points with the ground — as $\boldsymbol{p}_r$, $\boldsymbol{p}_f$. Then, the balance criterion is expressed as

$$\begin{aligned} \mu_r &= \boldsymbol{e}_r^T(-\boldsymbol{p}_r \times \boldsymbol{f} + \boldsymbol{\mu}) \geq 0 \ , \\ \mu_f &= \boldsymbol{e}_f^T(-\boldsymbol{p}_f \times \boldsymbol{f} + \boldsymbol{\mu}) \geq 0 \ . \end{aligned} \tag{10}$$

where all the vectors are expressed in $\mathcal{F}_2$. From (8), (9) it is possible to notice that μ_r, μ_f are function of the robot state and control input. Indeed, substituting (8) in (9), and then plugging the result in (10), we can derive the balance constraint

$$\boldsymbol{b}(\boldsymbol{x}, \boldsymbol{u}) \geq \boldsymbol{0} \tag{11}$$

Zero Moment Point computation Since (11) is equivalent to constraining the ZMP to be inside the support polygon [4], in the simulations we will check the position of the ZMP. Writing the static equilibrium with all the vectors expressed in $\mathcal{F}_2$, we have

$$\begin{aligned} \boldsymbol{s} + \boldsymbol{f} &= \boldsymbol{0} \\ \boldsymbol{p}_{zmp} \times \boldsymbol{s} + \boldsymbol{\mu} &= \boldsymbol{0} \end{aligned} \tag{12}$$

where $\boldsymbol{s}$ is the ground reaction force, and $\boldsymbol{p}_{zmp}$ is the position vector of the ZMP. From (12), since in our planar case it is $\boldsymbol{\mu} \perp \boldsymbol{f} \perp \boldsymbol{p}_{zmp}$, we get

$$\boldsymbol{p}_{zmp} = \begin{pmatrix} \frac{\mu_z}{f_y} \\ 0 \\ 0 \end{pmatrix} \tag{13}$$

where μ_z, f_y are respectively the z and y component of $\boldsymbol{\mu}, \boldsymbol{f}$, which also are the only non-zero components of these vectors.

4 Proposed method

When planning a throwing trajectory for a robot, the motion is divided into two main phases: the *throwing phase* that takes place in the time interval $[t_{init}, t_{rel}]$ and the *stopping phase* in $(t_{rel}, t_{final}]$. The robot first has to arrive at instant t_{rel} at an appropriate throwing state $\boldsymbol{x}_{rel}$, release the object to be thrown, and then it enters the stopping phase where it must decelerate reaching a resting state $\boldsymbol{x}_{final}$ at instant t_{final}.

To solve the motion planning problem we solve a single OCP incorporating all the phases. Note that the final stopping configuration is never specified, only the final velocity is required to be zero.

Let the duration of the throwing and stopping phase be respectively $T_t = t_{rel} - t_{init}$,$T_s = t_{final} - t_{rel}$ and the sampling time δ. The control intervals are $N = (T_t + T_s)/\delta$, hence the decision variables are

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{x}_0^T & \boldsymbol{u}_0^T & \boldsymbol{x}_1^T & \boldsymbol{u}_1^T & \dots & \boldsymbol{x}_{N-1}^T & \boldsymbol{u}_{N-1}^T & \boldsymbol{x}_N^T \end{bmatrix}^T \in \mathbb{R}^{9N+6} \tag{14}$$

where $\boldsymbol{x}_i, \boldsymbol{u}_i$ are the robot state and inputs at the $i-th$ control instant. The NLP is formulated as

$$\min_{\boldsymbol{w}} \sum_{i=0}^{N-1} (\dot{\boldsymbol{q}}_{r,i}^T \boldsymbol{W} \dot{\boldsymbol{q}}_{r,i} + x_{b,i}^2 \rho + \boldsymbol{u}_i^T \boldsymbol{R} \boldsymbol{u}_i) + \dot{\boldsymbol{q}}_{r,N}^T \boldsymbol{W} \dot{\boldsymbol{q}}_{r,N} \tag{15a}$$

subject to:

$$\boldsymbol{x}_{i+1} - \boldsymbol{\phi}_{d\text{-}t}^l(\boldsymbol{x}_i, \boldsymbol{u}_i) = \boldsymbol{0} \quad i = 0 \dots k-1 \tag{15b}$$

$$\boldsymbol{x}_{i+1} - \boldsymbol{\phi}_{d\text{-}t}(\boldsymbol{x}_i, \boldsymbol{u}_i) = \boldsymbol{0} \quad i = k \dots N-1 \tag{15c}$$

$$g(\boldsymbol{x}_k) = 0 \tag{15d}$$

$$\boldsymbol{x}_0 - \boldsymbol{x}_{init} = \boldsymbol{0} \tag{15e}$$

$$\dot{\boldsymbol{q}}_{r,N} = \boldsymbol{0} \tag{15f}$$

$$\boldsymbol{x}_{min} \leq \boldsymbol{x}_i \leq \boldsymbol{x}_{max} \quad i = 0 \dots N \tag{15g}$$

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}_i \leq \boldsymbol{u}_{max} \quad i = 0 \dots N-1 \tag{15h}$$

$$\boldsymbol{b}^l(\boldsymbol{x}_i, \boldsymbol{u}_i) \geq \boldsymbol{0} \quad i = 0 \dots k-1 \tag{15i}$$

$$\boldsymbol{b}(\boldsymbol{x}_i, \boldsymbol{u}_i) \geq \boldsymbol{0} \quad i = k \dots N-1 \ . \tag{15j}$$

The cost function is composed of three terms: we have the running and terminal cost associated to the joint velocity $\dot{\boldsymbol{q}}_r$ that are damping terms, and the cost associated to the control effort with its weighting matrix $\boldsymbol{R}$. In the constraints, $\boldsymbol{\phi}_{d\text{-}t}(\cdot,\cdot)$ is the discrete-time dynamics obtained with the 4th order Runge-Kutta integration method, $\boldsymbol{b}(\cdot,\cdot)$ is the previously derived balance constraint, and the superscript l indicates whether we are considering the robot dynamics with the load attached to the end-effector or not. Finally, $k = T_t/\delta$ is the index that identifies the throwing instant, $g(\cdot) = 0$ is the ballistic constraint — derived in the following — and $\boldsymbol{x}_{init}$ is an arbitrary resting initial state;

Ballistic constraint To derive the ballistic constraint (15d) as function of the state of the robot $\boldsymbol{x}$, it is sufficient to note that in (1), (2) the position $\boldsymbol{p}_0$ and velocity $\dot{\boldsymbol{p}}_0$ of the object at the starting instant of its ballistic motion coincide with the end-effector position and velocity at the throwing instant t_{rel}. We can express the direct and differential kinematics of the robot in the form

$$\boldsymbol{p}_{ee} = \boldsymbol{\sigma}(\boldsymbol{q}_r) \ , \ \dot{\boldsymbol{p}}_{ee} = \boldsymbol{J}(\boldsymbol{q}_r)\dot{\boldsymbol{q}}_r \tag{16}$$

where $\boldsymbol{\sigma}(\cdot)$ is the function that maps the joint coordinates of the robot to the end-effector position, and $\boldsymbol{J}(\boldsymbol{q}_r) = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{q}_r} \in \mathbb{R}^{2\times 3}$ is the end-effector jacobian. Then, substituting (16) in (2) we obtain the (scalar) ballistic constraint as function of the robot state in the form

$$g(\boldsymbol{x}) = 0 \ . \tag{17}$$

5 Simulations

In every experiment we intend to throw a ball of $3kg$ and, if not otherwise specified, we have the following parameters

$$T_t = 1\ s\ , \qquad T_s = 1.25\ s\ , \qquad \delta = 0.025\ s\ , \tag{18a}$$

$$\boldsymbol{W} = diag(\left[\tfrac{1}{h_1 h_2},\ 1,\ 1\right])\ , \qquad \boldsymbol{R} = \boldsymbol{I}_3\ , \qquad \rho = 5\ . \tag{18b}$$

where h_1, h_2 are the manipulator link lengths, and $\boldsymbol{I}_3 \in \mathbb{R}^{3\times 3}$ is the identity matrix. With these choices, we have $N_t = 40$ and $N_s = 50$. As a result, the decision variables are $\boldsymbol{w} \in \mathbb{R}^{816}$.

Moreover, unless otherwise specified, we choose the following joint position and velocity limits, and input torque bounds:

$$0 \leq x_b < +\infty\ [m]\ , \quad -\frac{\pi}{4} \leq q_1 \leq \frac{9}{10}\pi\ [rad]\ , \quad -\frac{\pi}{4} \leq q_2 \leq \frac{\pi}{2}\ [rad] \tag{19a}$$

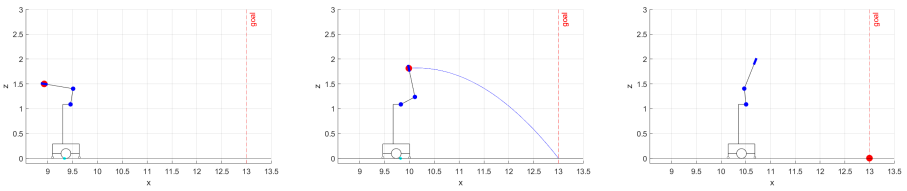
$$|\dot{x}_b| \leq 0.9998\ [m/s]\ , \quad |\dot{q}_1| \leq 5\ [rad/s]\ , \quad |\dot{q}_2| \leq 5\ [rad/s] \tag{19b}$$

$$|u_w| \leq 6\ [Nm]\ , \quad |u_1| \leq 39\ [Nm]\ , \quad |u_2| \leq 39\ [Nm]. \tag{19c}$$

where the joint position limits allow the robot the widest possible range of motion while avoiding self-collision.

The next simulations share the same resting initial configuration $\boldsymbol{q}_{r,0} = (9.3647,\ 1.4062,\ \pi/2)\ [m, rad, rad]$. We examine the maximum reachable x_{goal}, found with a resolution of $0.1\ m$, given $\boldsymbol{x}_{init}$, the parameters (18) and the bounds (19).

Given the previous constraints, it is possible to increase x_{goal} up to $13m$. We report the relevant robot configurations during the throwing motion in Fig. 3



(a) Initial configuration (b) Throwing configuration (c) Stopping configuration

Figure 3: Relevant robot configurations during the optimized throwing motion. The ball is released from the configuration in (b) with a velocity $\boldsymbol{v} = (4.6793,\ 0.3393)\ [m/s]$ and its time of flight is $T_f = 0.6439\ s$. The achieved throwing range is $3.013\ m$. The cyan marker in (a) and (b) is the ZMP. It is not present in (c) as we do not have a control input at instant t_{final}.

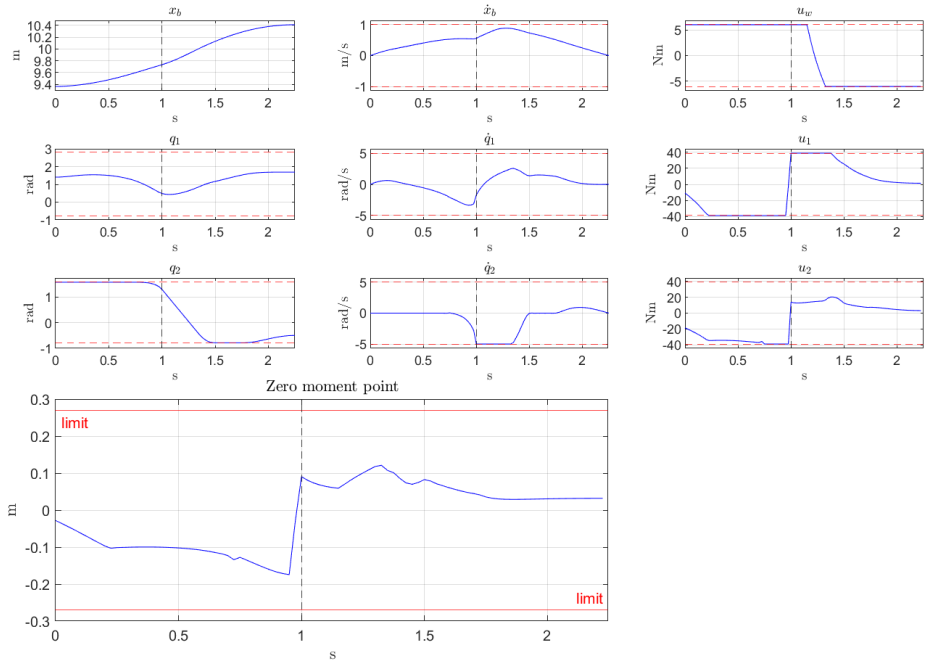


Figure 4: Joints data and ZMP profile. The ZMP profile is one dimensional since the support polygon (see Fig. 2) is a segment. The black vertical dashed line reported in all the plots identifies the throwing instant t_{rel}.

5.1 Increase torque bounds

One thing to notice in the reported simulation is that the balance constraint is inactive since the ZMP in Fig. 4 is far from the bounds. Indeed, removing the balance constraint still yields balance-safe trajectories. To clearly see the influence of the balance criterion, the control input bounds are updated as follows

$$|u_w| \leq 6 \ [Nm] \ , \ |u_1| \leq 70 \ [Nm] \ , \ |u_2| \leq 70 \ [Nm] \ . \tag{20}$$

These new bounds were found via several trials with various torque combinations, and this choice was the one that made the balance constraint active with the least modifications to the original bounds (19c) on $|u_1|, |u_2|$.

Indeed, starting from the same initial configuration and using the input bounds (20), the goal can be increased up to 14.6 m, and from Fig. 5 it is possible to see that without balance constraint the robot would fall, while integrating it in the optimization yields a balance-safe trajectory.

5.2 Approach 2: aggressive manoeuvres

All the results offered in this section are obtained with bounds (19a), (19b), (20) and the throwing time is updated to $T_t = 2$ s. Moreover, one may argue that the

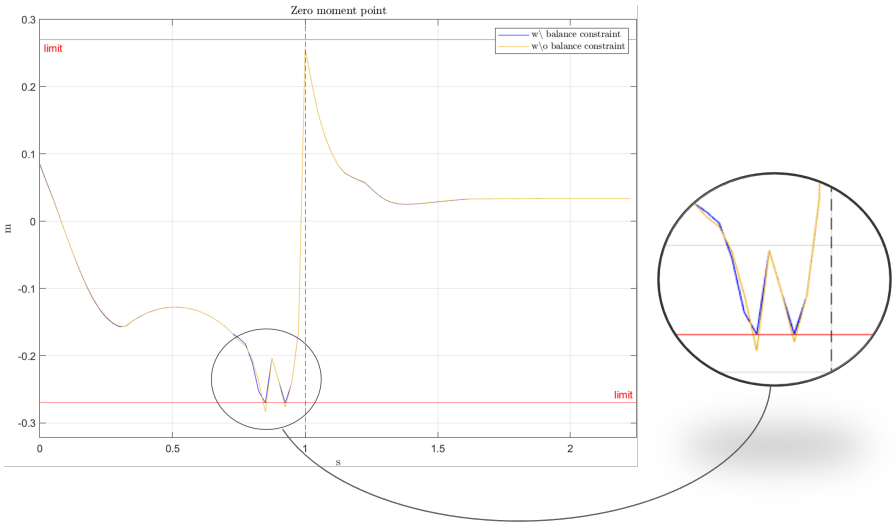


Figure 5: ZMP profile with torque bounds (20), integrating the balance constraint in the optimization (blue) and removing it (orange). In the latter case we see the ZMP exceeds the bounds, i.e. the robot falls.

initial configuration considered so far is clearly intended for throwing and not a generic one. For this reason, other simulations are analyzed with different initial configurations. These changes are done to show more aggressive and dynamic motions, to better appreciate the importance of the balance constraint.

It is worth reporting that solving (15) is computationally expensive as its decision variables now are $\boldsymbol{w} \in \mathbb{R}^{1176}$. However, using as initial guess a previous solution with a slightly different $\boldsymbol{p}_{goal}$ reduces the computation time by a factor of 4. The simulations we carried out involve three different initial configurations:

C1: stretched down manipulator arm;

C2: stretched forward manipulator arm;

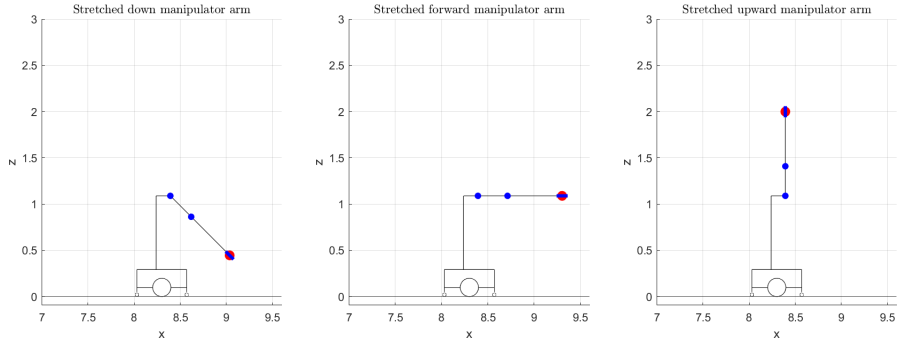
C3: stretched upward manipulator arm.

Moreover, the influence of the parameter ρ is investigated analyzing, for each of the three cases, two scenarios

S1: set $\rho = 5$ in (15);

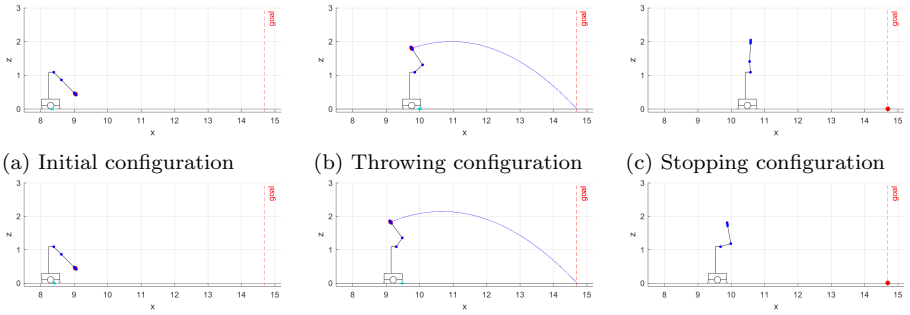
S2: set $\rho = 1000$ in (15), assigning high importance to the robot base distance from the origin;

However, results show that simulations sharing the same $\boldsymbol{p}_{goal}$ only differ in how they reach the throwing state (that is similar in the three cases). Hence, for the sake of clarity, we here focus solely on C1 (see Fig. 6a) as it is a more



(a) $\boldsymbol{q}_r = (8.3,\ -\pi/4,\ 0)$ (b) $\boldsymbol{q}_r = (8.3,\ 0,\ 0)$ (c) $\boldsymbol{q}_r = (8.3,\ \pi/2,\ 0)$

Figure 6: Representation of the initial configurations C1, C2, C3 and the respective joint coordinates value $\boldsymbol{q}_r$.



(a) Initial configuration (b) Throwing configuration (c) Stopping configuration

(d) Initial configuration (e) Throwing configuration (f) Stopping configuration

Figure 7: Relevant configurations of the simulations starting with the stretched down arm. The first and second row refer to the planned trajectories with S1 and S2 respectively. The cyan marker in (a),(b), (d) and (e) is the ZMP, and it is not present in (c) and (f) as we do not have a control input at instant t_{final}.

likely initial state for a pick-and-place task, but the same considerations can be done also for the other cases.

Videos of all the experiments — including the ones using C2, C3 — are available at this link.

5.2.1 Stretched down arm

The starting configuration is shown in Fig. 6a where we highlight that q_1 is at its lower position limit. In Fig. 7 are reported the resulting relevant robot configurations associated to S1, S2.

As expected, a direct comparison between the throwing configurations — respectively in Fig. 7b, 7e — shows that a higher penalization of the base motion in scenario S2 allows the object to be released at a greater distance from

the goal position, with a difference of 0.5532 m. This is also reflected in the stopping phase as the robot is able to shorten its stopping distance of 0.3433 m (see Fig. 7c, 7f and the top left plot in Fig. 8). Thus, the whole distance covered by the robot base is reduced of 0.8965 m, corresponding to about 41%.

A further analysis of the stopping configurations in Fig. 7 shows a clear difference: in S2 the robot stops in a configuration quite different from the almost stretched up one, reached in S1. A possible motivation is that weighting more the base motion term results in a decrease in the relevance of control effort minimization in the cost function, as confirmed by the torques profile in Fig. 8.

Fig. 8 also shows that the ZMP reaches the limits without exceeding them thanks to the presence of the balance constraint in the optimization scheme, hence the planned trajectory is balance-safe.

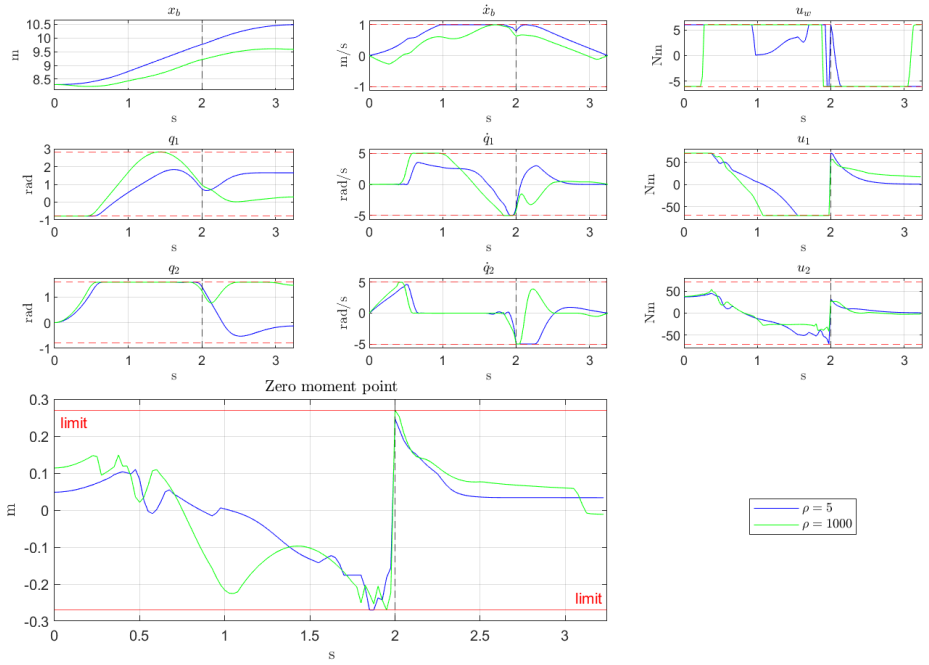


Figure 8: Joints data and ZMP profile for the experiment done starting with the arm down configuration in scenarios S1 (blue), S2 (green). The ZMP profile is one dimensional since the support polygon (see Fig. 2) is a segment. The black vertical dashed line reported in all the plots identifies the throwing instant t_{rel}.

References

[1] Sergey A. Kolyubin and Anton S. Shiriaev. Planning longest pitch trajectories for compliant serial manipulators. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3150–3155, 2016.

[2] Ferenc Lombai and Gabor Szederkenyi. Throwing motion generation using nonlinear optimization on a 6-degree-of-freedom robot manipulator. In *2009 IEEE International Conference on Mechatronics*, pages 1–6, 2009.

[3] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. Robotics: Modelling, planning and control. 2008.

[4] Spyridon G. Tarantos and Giuseppe Oriolo. Real-time motion generation for mobile manipulators via NMPC with balance constraints. In *2022 30th Mediterranean Conference on Control and Automation (MED)*, pages 853–860, 2022.

[5] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics, 2019.

[6] Yajia Zhang, Jingru Luo, and Kris Hauser. Sampling-based motion planning with dynamic intermediate state objectives: Application to throwing. In *2012 IEEE International Conference on Robotics and Automation*, pages 2551–2556, 2012.