Whole-Body MPC for a Dynamically Stable Mobile Manipulator

Maria Vittoria Minniti, Farbod Farshidian, Ruben Grandia, Marco Hutter

Abstract—Autonomous mobile manipulation is the cutting edge of modern robotic technology, which offers a dual advantage of mobility provided by a mobile platform and dexterity afforded by the manipulator. A common approach for controlling these systems is based on the task space control. In a nutshell, a task space controller defines a map from user-defined end-effector references to the actuation commands based on an optimization problem over the distance between the reference trajectories and the physically consistent motions. The optimization, however, ignores the effect of the current decision on the future error, which limits the applicability of the approach for dynamically stable platforms. On the contrary, the Model Predictive Control (MPC) approach offers the capability of foreseeing the future and making a trade-off in between the current and future tracking errors. Here, we transcribe the task at the end-effector space, which makes the task description more natural for the user. Furthermore, we show how the MPC-based controller skillfully incorporates the reference forces at the end-effector in the control problem. To this end, we showcase here the advantages of using this MPC approach for controlling a ball-balancing mobile manipulator, Rezero. We validate our controller on the hardware for tasks such as end-effector pose tracking and door opening.

Index Terms—Mobile Manipulation, Optimization and Optimal Control

I. INTRODUCTION

The primary focus of this work is on whole-body motion planning of mobile manipulators where the platform is only dynamically stable. A common strategy to achieve this goal is to define the desired trajectories in terms of a time history of the end-effectors position, velocity, and acceleration and then use a task space control scheme to compute the corresponding actuator torques. These approaches often use an inverse model (e.g., the inverse dynamics) to minimize an instantaneous error in between the desired trajectories and the dynamically consistent motion. In contrast to the task space control scheme, the optimal control approach makes a trade-off in between minimizing the current and future tracking errors. So unless the user provides meticulously designed references which are dynamically consistent, we should expect that the optimal control approach results in a superior solution in terms of tracking error and minimizing the actuation torque.

To this end, we employ an optimal control approach where we devise a cost function to penalize the deviation of the pose and force of the end-effectors from user-defined references. By using the full dynamics of the system as constraints for the optimal control formulation, the dynamical consistency of the state-input trajectories is guaranteed. To further increase the robustness of the control structure, we here employ a Model Predictive Control (MPC) approach to adjust the plan based on the most recent estimation of the system’s state in a receding horizon fashion.

The platform which we choose to validate our approach is Rezero as shown in Fig. 1. Rezero is a torque-controlled robot with mobile manipulation capabilities. Its mobile base is an omnidirectional platform moving on a ball through three omni wheels. Manipulation is achieved by means of a 3 DoF arm mounted on top of the base.

However, for under-actuated manipulators (which are often mobile manipulators), they would result in different motions if the reference trajectories are not dynamically consistent. Here, the task space control method only minimizes the instant tracking error with no regard about how much this decision will affect the future tracking error. In contrast, the optimal control makes a trade-off in between minimizing the current and future tracking errors. So unless the user provides meticulously designed references which are dynamically consistent, we should expect that the optimal control approach results in a superior solution in terms of tracking error and minimizing the actuation torque.

Fig. 1: Rezero is a torque-controlled robot with mobile manipulation capabilities. Its mobile base is an omnidirectional platform moving on a ball through three omni wheels. Manipulation is achieved by means of a 3 DoF arm mounted on top of the base.
scheme should use a whole-body approach where the full dynamics of the robot and the interaction forces at the end-effectors are concurrently planned to maintain the stability of the robot. This characteristic makes Rezero an ideal platform for validating our whole-body motion control approach.

Our main contributions in this manuscript are: (1) Employing an MPC approach for whole body control of a mobile manipulator while using the full dynamics and directly reasoning on the interaction force at the end-effectors, (2) Transcribing the task directly for the motion and force at the end-effector and devising the actuation command from the forward model instead of an inverse model, (3) Using such a control scheme for the first time on a ball-balancing robot. To the best of our knowledge, this is the first time that such a system is shown performing coordination between dynamic balancing and manipulation in an MPC fashion.

A. Related Work

The problem that we address in this paper concerns the motion and force control of dynamically stable mobile manipulators. One of the most remarkable examples of robots capable of solving such a problem is SpotMini by Boston Dynamics, which has shown a robust reactive behavior while executing manipulation tasks such as opening a door. Recently, Bellicoso et al. [1] solved a similar problem with ANYmal equipped with an arm. However, their approach considered balance and force control at the current time instant using a task space approach, while in our approach we tackle the same problem on a time horizon in an MPC fashion.

The first application of real-time whole-body MPC to a humanoid robot is described in [2]. Recently, a similar problem has been treated in [3], where contact forces are optimized based on a ZMP-based optimization method. In these works, the primary focus is to exploit contacts to keep balance, while here we concentrate on planning over contact forces to execute manipulation tasks.

Whole-body MPC with the planning of external forces is also addressed in [4], where the authors plan over the contacts sequence while minimizing one component of the external contact force. On the contrary, here we penalize the norm of the error between the desired force to exert a task and the measured one.

In addition to legged systems equipped with manipulators, other types of dynamically balancing robots with manipulators have also addressed the motion control problem. In [5], [6] a hierarchical motion controller for a wheeled inverted pendulum humanoid robot is presented, showing results in simulation.

In the robotics community, many ball-balancing robots have been built in the past years ([7], [8], [9], [10]), although there are not many examples of such systems provided with manipulation capabilities. The first Ballbot system was introduced by Hollis et al. [7] at Carnegie University. The same group enhanced the capabilities of their robot by providing it with a pair of 2 DoF arms. In [11], the authors describe a planning strategy for this system. The capabilities of this robot are shown in [12]. However, their control problem significantly differs from ours. Indeed, unlike Rezero, CMU Ballbot does not allow any yaw motion for either its base or its arm. Therefore, it is possible to model it in 2D and to obtain equations where shape and position variables are decoupled. The same approach would not allow controlling our system. A model-based controller for a ballbot with a manipulator was also presented in [13], where results are shown in a simulation.

The first generation of Rezero [14] was capable of showing dynamic motions while navigating through humans and obstacles. The system was controlled through interpolation of LQR controllers at different operating points. Later, to cope systematically with the nonlinear dynamics of the system, other control techniques were tested on this platform, such as an MPC framework [15] and a reinforcement learning method [16].

II. System

A. Physical System

Rezero is a torque-controllable robot with a ball-balancing mobile base and a 3 DoF arm (Fig. 1). The platform weighs about 25 kg. It is equipped with an onboard computer (Intel i7-7500U, 2.7-3.5 GHz, dual-core 64 bit) which is used for state estimation, motion control, and low-level actuators communication. Actuation is achieved by means of series elastic actuators, ANYdrives [17]. Three omni wheels are placed in a triangular layout between the main body of the robot and the ball. Because of the tangential forces that they apply on the ball, they allow the robot to move in any direction and to also rotate around its own axis. The manipulator on top of the robot is an anthropomorphic arm [18], meaning that the joints can rotate around the local z, x and y axes. It is designed so that it is easy to change its end-effector or to add additional degrees of freedom.

B. System Modeling

In general, the motion of a mobile manipulator with an arm is characterized by \( n_b + n_a \) degrees of freedom, where \( n_b \) is the number of free variables of the mobile base, and \( n_a \) is the number of degrees of freedom of the arm.

We describe the motion of Rezero with a minimal set of generalized coordinates. These coordinates are the 2D position of the ball in the horizontal plane, the orientation of the base, and 3 joint angles of the arm as depicted in Fig. 2. We refer to the base as the main body of the robot, between the ball and the arm. The system has in total 8 degrees of freedom \( (n_b = 5, n_a = 3) \). Rezero is equipped with 6 actuators \( (n_e = 6) \), 3 for actuating the ball and 3 at each joint of the arm.

In Fig. 2 the frames and variables used for the kinematics of the ballbot are depicted. We decide to place a frame parallel to the base frame at the center of the ball.

We introduce the vector of generalized coordinates and
velocities $q, \nu \in \mathbb{R}^{n_b+n_a}$, given by:

$$
q = \begin{bmatrix} p_{IS} \\ \theta \end{bmatrix}, \quad \nu = \begin{bmatrix} \dot{p}_{IS} \\ \omega_{IB} \end{bmatrix},
$$

where $p_{IS} \in \mathbb{R}^2$, $\theta \in \mathbb{R}^3$ and $q_a \in \mathbb{R}^{n_a}$ indicate the ball’s position in the horizontal plane, a vector of ZYX Euler angles representing the orientation of the base, and the joint coordinates of the arm, respectively. $\omega_{IB} \in \mathbb{R}^3$ denotes the base angular velocity vector.

Having chosen the vectors $q$ and $\nu$, we treat the robot as a dynamical system with state vector $x = (q, \nu) \in \mathbb{R}^{2(n_b+n_a)}$ and input vector $u = (\tau, \lambda_{ee}) \in \mathbb{R}^{n_r+n_{\lambda_e}}$, where $\tau$ and $\lambda_{ee}$ are the actuation torques and the forces exerted by the environment on the end-effector, which may include both linear forces and moments.

Then, the state equations are given by $\dot{x} = f(x, u)$, where

$$
f(x, u) = \begin{bmatrix} T(q) \nu \\ M(q)^{-1}(-h(q, \nu) + S^T(q)\tau + J^T_{ee}(q)\lambda_{ee}) \end{bmatrix}.
$$

Here, $T(q) \in \mathbb{R}^{n_b+n_a \times n_b+n_a}$ maps the vector of generalized velocities to the vector of generalized coordinate derivatives. The last $n_b+n_a$ equations represent the forward dynamics of the mechanical system. We also point out that the actuator selection matrix $S(q)$ depends on the generalized coordinates due to the way that the ball is actuated.

C. Extracting Rigid Body Equations

As other model-based optimization methods, our MPC method needs to compute the linearization of the system dynamics, which contains the nonlinear rigid body equations of motion. Hence, an automatic differentiation tool (such as Auto-Diff) is required to compute derivatives of the system dynamics. In order to apply such a tool, the forward dynamics function needs to be templated on a special Auto-Diff scalar type. This feature is offered by RobCoGen [19], an efficient code generation framework for modeling Rigid Body Dynamics.

The input of RobCoGen is the kinematic chain of Rezero. This, according to the description given in [11],[12], can be specified through the use of two prismatic joints for the translation of the ball, three revolute joints for the base orientation, and the arm joints. The software generates the equations of motion in the standard form as

$$
\dot{M}(q)\ddot{q} + h(q, \nu) = \begin{bmatrix} f_x \\ f_y \\ S^T(q)\tau \end{bmatrix},
$$

where $\dot{M} \in \mathbb{R}^{(n_b+n_a) \times (n_b+n_a)}$, $h \in \mathbb{R}^{(n_b+n_a)}$ are the mass matrix and the vector of nonlinear effects of the system, respectively. Here, the symbol $\dot{\cdot}$ indicates that the elements in equation (3) are not the same as the ones in equation (2). $f_x, f_y \in \mathbb{R}$ are the generalized forces acting on the two prismatic joints representing the motion of the ball. $\tau \in \mathbb{R}^{3+n_a}$ groups together the actuation torques.

However, equation (3) does not take into account the ground reaction forces that allow pure rolling between the ball and the ground. To include the effect of pure rolling in our modeling framework, we use the following strategy.

Let $\tau_x, \tau_y \in \mathbb{R}$ be the interaction torques in between the ball and the wheels. $\tau_x = S^T_x(q)\tau$ and $\tau_y = S^T_y(q)\tau$ are nonlinear functions of the base orientation and linear functions of the wheels torques. For the derivation of their analytic expressions, we refer the reader to [14]. Furthermore, let $m, r, I \in \mathbb{R}$ be the mass, radius, and moment of inertia of the ball. We have that $I = kmr^2$, where $k \in \mathbb{R}$ is a specific constant. Deriving the equations of motion of the ball yields:

$$
f_x = -km\ddot{p}_{ISx} + \tau_x/r, \quad f_y = -km\ddot{p}_{ISy} - \tau_x/r.
$$

Substituting (4) into (3) yields:

$$
\begin{bmatrix} M & 0 \\ 0 & km \end{bmatrix} \begin{bmatrix} \dot{q} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} 1/r S^T_x \\ -1/r S^T_y \end{bmatrix} \tau, \quad (5)
$$

where with $0$ we indicate a zero matrix with appropriate dimensions.

III. Control Method

A. Optimal Control Problem

In this section, we describe the optimization problem and a general form of the employed cost function, which are the key points of our control approach.

Let $p_{IE} \in \mathbb{R}^3$ be the position of the end-effector with respect to the inertial frame. We also define $e_o \in \mathbb{R}^3$ to be an orientation error, which describes the deviation between a desired orientation and a measured one. Depending on the chosen parameterization for the end-effector orientation, there are multiple ways to express such an error. The reader can find additional details in [18], where all the different forms for the orientation error are exhaustively explained. For the sake of completeness, we briefly describe the quaternion distance measure which we used in this paper.

Let $Q_d = \{q_d, e_d\}$, $Q_e = \{q_e, e_e\}$ be unit quaternions representing the desired end-effector orientation and the measured one, respectively. If the desired end-effector frame and the measured one coincide, the equations $\Delta Q = Q_d*Q_e^{-1} = \{1, 0\}$ must be satisfied, where the symbol $\star$ indicates the
quaternion product operation. This yields the following form for the orientation error:

\[ e_o = \eta_e e_d - \eta_d e_e - [e_d] \times e_e \]  

where \([\cdot] \times\) indicates the skew matrix operator.

Furthermore, we define \( \xi = (v, w) \in \mathbb{R}^6 \) as the twist describing the motion of a rigid body, and \( \lambda = (f, m) \in \mathbb{R}^6 \) as the total wrench acting on a rigid body, which includes linear force and moment. We will use subscripts to indicate the rigid bodies to which these variables refer.

For every rigid body, our algorithm allows planning over pose, twist, and wrench. Therefore the set of optimization variables is \( SE(3) \times \mathbb{R}^6 \times \mathbb{R}^6 \). According to our modeling framework, the motion and force planning problem can be written in terms of the following minimization problem

\[
J = \min_{u(t)} \frac{1}{2} \int_0^T \left[ \|p_{IE}^d - p_{IE}^e\|_{Q_{ee}^{pos}}^2 + \|p_{IS}^d - p_{IS}^e\|_{Q_{pos}}^2 + \|e_{vel}\|_{Q_{vel}^{ee}}^2 + \|e_{vel}\|_{Q_{vel}^{ee}}^2 + \|\xi_{vel}\|_{Q_{vel}^{vel}}^2 + \|\tau\|_{R_e}^2 + \|\lambda_{vel} - \lambda_{vel}\|_{R_{vel}}^2 \right] dt,
\]

such that the system’s equations of motion and correspondent initial conditions are satisfied

\[
\dot{x} = f(x, u), \quad x(0) = x_0.
\]

In (7), \( Q_{ee}^{pos}, Q_{ee}^{vel}, Q_{vel}^{pos}, Q_{vel}^{vel} \) are semidefinite positive matrices which weight position, orientation and (linear and angular) velocity errors at the end-effector and ball frame, respectively. As introduced in section II-B since in our robot model a frame parallel to the base frame is located at the center of the ball, it is equivalent to control the ball or base orientation and angular velocity. \( R_{\lambda} \) is a positive definite matrix for the end-effector wrench error. Finally, to penalize high torque commands, we also add a quadratic penalty on the torque inputs with the weight matrix \( R_{\tau} \).

The choice \( Q_{vel}^{pos} = 0 \) gives the algorithm the freedom to plan a desired motion of the ball and the arm in order to track a desired end-effector pose. However, sometimes it may be useful to have \( Q_{vel}^{pos} \neq 0 \). This is required if the ball of the robot needs to execute specific trajectories while performing manipulation tasks.

Similarly, we may wish not to optimize over end-effector forces and solve a pure motion control problem. Since both \( R_{\tau}, R_{\lambda} \) must be positive definite, there are two possibilities. One consists of excluding force inputs. Otherwise, the same effect is achieved setting \( \lambda_{vel} = 0 \).

For numerical reasons, it may be useful to add a regularization term to the cost function penalizing the arm joint velocities.

For completeness, in (7), we added a quadratic term with respect to \( e_{vel} \). We clarify here that we always only penalize the yaw component \( \theta_z \) of the base orientation, and let the controller optimize for \( \theta_y \) and \( \theta_x \) based on the desired task.

B. SLQ-MPC Control

The nonlinear optimal control problem defined in equations (7) and (8) has been solved using the Sequential Linear Quadratic (SLQ) algorithm in a model predictive control fashion; details on the employed solver and the MPC scheme can be found in [20], [21].

In summary, this algorithm designs a continuous controller for a nonlinear system. In addition, it has a linear computational complexity with respect to the optimization time-horizon.

In each iteration, the continuous-time SLQ algorithm forward integrates the system dynamics along the time horizon based on the last approximation of the optimal controller. Then, it linearizes the system dynamics and quadratizes the cost function over the state and input trajectories derived from the forward integration.

Finally, the algorithm solves a constrained, time-varying LQR problem based on the quadratic cost function and linearized system. As a result, the updated control input has the form

\[
u(t, x) = u_{ff}(t) + \alpha \delta u_{ff}(t) + K(t)(x(t) - x_{ff}(t)),
\]

where \( \alpha \) is a line-search parameter used to update the feedforward law.

This algorithm runs continuously an iterative MPC scheme in real-time, where each MPC problem is triggered by a new state measurement.

For the tracking part, we use the feedforward torques, positions and velocities \( \tau_{ff}, q_{ff}, \nu_{ff} \), which are outputs of the SLQ-MPC algorithm.

So far, we have assumed perfect tracking of the actuators torques. This is usually not true when dealing with series elastic actuators, such as those used during the experiments. To deal with this modeling mismatch, one possibility is to slightly modify the optimal control problem according to a frequency aware approach, which is described in [22]. This addition allows considering bandwidth limitations due to the presence of spring elements and has been successfully used to improve torque tracking of the ball actuators.

C. Tracking Problem

Depending on the complexity of the minimization problem, the frequency of the MPC planner can be less than the control loop frequency (which is 400 Hz for Rezero). Hence, we need to close a local feedback loop to improve the performance of the control.

Therefore, it is useful to introduce a set of actuation positions and velocities \( q_{act}, v_{act} \), which can be different from the variables used to describe the motion of the system.

There usually exists a linear relationship between the actuation and the generalized velocities: \( q_{act} := T_{act}(q)v \), from which feedforward actuation positions can be obtained by integration.

In a ballbot system, \( T_{act}(q) \) represents the relationship between the generalised velocities used in the modeling and the velocities of the wheels joints. We remind here that the
behaviour is mainly dictated by the feedforward term and, as it will be shown in the results section, the actuators torque. Indeed, the elements of $K_{st}$ stability and represents a small contribution to the total modeling part.

Finally, actuation torques can be computed according to:

$$\tau_{act} = \tau_{ff} + K_D(q_{act\,ff} - \dot{q}_{act}) + K_P(q_{act\,ff} - q_{act}),$$

(10)

where $K_P$, $K_D$ are two positive definite diagonal matrixes. The feedback term in equation (10) is used to guarantee stability and represents a small contribution to the total torque. Indeed, the elements of $K_P$, $K_D$ are very small and, as it will be shown in the results section, the actuators behaviour is mainly dictated by the feedforward term $\tau_{ff}$.

We point out that (10) has been used to control Rezero but it is not the only possible choice for a tracking controller. In general, any dynamically consistent controller acting at the torque level can be used for tracking.

IV. RESULTS

In this section, we validate the performance of our control algorithm. In each test, all computations run on the robot’s onboard computer (refer to subsection II-A). The controller relies on a state estimator running at 400 Hz which fuses odometry measurements from the wheels encoders and IMU information to get an estimate of the base pose, linear and angular velocity. The algorithm used for state estimation is based on [23]. The MPC loop runs in a separate ROS node at a frequency between 70 and 80 Hz. In all the experiments, the time horizon has been fixed to 1.3 s. The tracking controller is implemented inside the internal motor controllers and runs at 2.5 KHz. The structure of the controller is schematically described in Fig. 3.

To validate the performance of our proposed motion control approach, we have conducted four experiments. These experiments showcase different aspects of our proposed control approach introduced in section II-A. The first experiment focuses on the ability of the controller to coordinate the motion of the base and the arm while the task is just defined at the arm’s end-effector space. The second experiment demonstrates the disturbance rejection capability of the controller for two different tasks, the ball position control, and the end-effector’s position control task. The third experiment highlights the necessity of a whole-body control approach for Rezero. The experiment compares the behavior of the robot under two different control schemes: the proposed whole-body control approach and a method in which the base and the arm are controlled separately. Finally, the last experiment demonstrates how the control scheme skillfully incorporates the reference contact forces at the end-effector into the motion planning for a door opening task. All the results are also presented in the video attachment [1] which shows the ability of the robot to perform different tasks.

A. End-effector Reference Pose Tracking

In this experiment, we focus on end-effector pose tracking while ensuring the balance of the robot. To this end, we set the cost weight for the ball pose and twist to zero. However, we penalize the end-effector pose error and add a small penalty on its twist. An important aspect of this experiment is that without specifying the base motion reference, the controller coordinates the motion of the base and the arm in an optimal way such that Rezero fulfills its end-effector task while dynamically balancing.

The tracking performance of the end-effector is demonstrated in Fig. 4 which is the robot’s response to a sequence of step references for the end-effector’s pose in the world frame. In these plots, the end-effector pose tracks the reference commands with an acceptable rise time response and steady-state error. The quality of the motion can be modified by setting higher weights on the pose tracking error and reducing the penalty over the end-effector’s twist. In the attached video, you can observe how the base orientation is continuously adapting to the motion of the arm so that the platform remains stable.

Note that due to the lack of degrees of freedom at the arm, in order to change the rolling along the normal axis of the end-effector (x axis), the base of the robot should tilt. However, since the robot keeps its balance by regulating its tilt angles, the robot cannot track an arbitrary reference around this direction. This effect can be observed in the third plot of Fig. 4. This aspect highlights the capability of our optimal controller to make a trade-off between the tracking error minimization at the current instant and the balance of the system in the future.

B. Disturbance Rejection

The primary purpose of this experiment is to demonstrate the disturbance rejection capability of our motion controller. To this end, we have designed two tasks. In the first task, we are only interested in controlling the ball position. Thus, the cost weight for the ball position is high. As for the arm, we only weight the end-effector orientation error and we do not penalize end-effector position error. In the second task, we intend to control the position of the end-effector. Therefore, we place a higher weight for the end-effector’s position error.

For the first task experiment, we perturb the robot twice by pushing it away from its initial position. Our disturbances are applied successively along the y and x directions. The evolution of the ball position and roll and pitch angles is shown in Fig. 5. The robot maintains its balance by tilting its base pose against perturbation direction (see the attached video). This is a natural behavior where the robot shifts its center of the mass against the perturbation.

https://youtu.be/a9hnJcQvvYE
In the attached video, we show the disturbance rejection behavior of the second task where we control the end-effector position. Similarly to when the controlled variable is the ball position, the robot reacts to the disturbance with the motion of its whole body. However, in this experiment, it uses its arm more prominently than before.

C. The Necessity of the Whole-body Control

Here, we are interested in controlling the ball and the end-effector position at the same time. As a showcase for this possibility, we give the end-effector a sequence of reference positions, while setting the initial position as a goal for the ball. In Fig. 6 we show how the base adapts to the center of mass location due to the arm motion. Since the robot lower body is consistently heavier than the manipulator, we added weight to the end-effector to highlight the base adaptation.

Furthermore, to manifest that the feedforward term represents the major contribution of the total torque in equation (10), in Fig. 7 we compare the trajectories of the measured torque and the feedforward torque for the second joint of the arm during the experiment. We also compute the ratio

\[ \alpha := \frac{|\tau_{ff}|}{|\tau_{ff}| + |\tau_{fb}|}, \]

where \( \tau_{fb} \) represents the feedback part, and plot it in percentage.

As another example of both ball and end-effector position tracking, in the video attachment we show the behaviour of the robot while we command the ball to move on a circular trajectory and the end-effector to remain in the same position.

To further highlight the importance of our whole-body control approach, in the attached video, we show the performance of a setup which has two separate modules for controlling the base and the arm. Here, we use a joint-level PID controller for the arm. This controller transforms the
transcribed task at the end effector to a joint level task using an inverse kinematics model. For the mobile base, we use a degraded version of our MPC method such that it only controls the base. This control only considers the arm at a nominal configuration. However, it does not know about the arm motion, and it only receives updated measurements of the base and the ball. Therefore, the base controller cannot account for changes in dynamics due to the arm motion.

Similar to the whole-body control experiment, we command the ball to keep a fixed position, while we move the arm’s end-effector references. Contrary to the whole-body controller, the ball position does not remain fixed as the arm changes configuration (refer to the attached video). This experiment manifests the importance of a whole-body control approach for coordinated control of the base and the manipulator in dynamically stable platforms.

D. End-effector Force Planning

In this experiment we want the controller to plan a desired force trajectory at the end-effector in order to push and open a door (see Fig. 8). Therefore, we have not assigned any cost on the end-effector position and velocity. However, we give a penalty to the end-effector orientation and to the error between a desired linear force and a measured one.

In Fig. 9, the force tracking errors and the evolution of roll and pitch angles are shown, for two scenarios where the desired force is 5N and 10N. The planned roll and pitch motion is such that the robot leans in order to exert a force and remain stable during the experiment. This generated behaviour is very similar to that of a human trying to push a heavy door. Therefore, the base is more inclined when the commanded force is higher. A demonstration of the door pushing experiment with a commanded force of 5N can be visualized in the attached video.

V. Conclusion

In this paper, we have proposed a whole-body motion controller for a ball-balancing robot provided with a manipulator. The proposed controller is based on an MPC approach, which tracks user-defined pose and force references at the end-effectors. By defining the desired references as a soft constraint in the cost function, we have allowed the controller to compromise the quality of the tracking to favor its stability.

Similar to the task space approach, we have designed tasks at the end effector space where the user can naturally define the reference trajectories. However, in contrast to the task space controller, the MPC framework makes a trade-off in between the instantaneous tracking error and the future tracking error. Moreover, the user is not required to assign a task to each degree of freedom of the robot (neither as a tracking task nor as a damping task). We have showcased this characteristic in the end-effector tracking task where the desired references are only defined at the arm’s end-effector.

By augmenting the action space of MPC with the end-effector forces, we have shown that our controller naturally modifies the base tilt angle to insert the desired forces for a door opening task. Last but not least, we have applied a whole-body, MPC-based motion controller for a ball-balancing robot with an arm. To the best of our knowledge, this is the first time that such a control approach is used in this family of robots.

A. Future Work

Although this framework gives the user the flexibility to design the optimization problem based on the desired task, there are still more steps forward to take to improve our results. First, we intend to enhance the hardware platform to show more dynamic base and end-effector movements. Second, we need to improve further our contact model such that it includes impact and slippage models.

Finally, we have addressed the force planning problem considering only the robot and neglecting the world dynamics. Although it is not explicitly declared during the door opening experiment, by tracking different force references the optimization assigns an arbitrary impedance to the environment at its interaction port. This means that in order to be consistent with the physics of the environment, we should have added a set state-input equality constraints to the MPC problem which relates the end-effector projected acceleration to the planned forces. However, we have not considered these constraints. Then, the question is why this controller works?

The key to this issue is the replanning scheme of our controller. As long as we underestimate the impedance of the door (i.e., the reference forces are low enough), the door will move less than what our controller planned. Therefore, the contact remains closed and MPC can deal with this discrepancy by replanning from the current state. In our future work, we would like to look closer into this issue and improve the controller based on the estimation of the interaction port impedance.
Fig. 9: Evolution of end-effector forces and base orientation (roll and pitch angles) during a door pushing experiment. The shaded areas highlight the trajectories when the contact switch phases occur. In the left size figure, the commanded force in the end-effector normal direction is $-5\,N$, while in the left size figure the commanded force is $-10\,N$.