COBEM 2021 Florianópolis - Brasil
26th ABCM International Congress of Mechanical Engineering
November 22-26, 2021. Florianópolis, SC, Brazil

COB-2021-0908
DRIVING MOTIONS FOR WHEELED-LEGGED ROBOTS IN ROUGH TERRAIN USING 2D TRAJECTORY OPTIMIZATION

Vivian Suzano Medeiros
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rua Marquês de São Vicente, 225, Gávea, Rio de Janeiro, RJ, Brasil, 22451-900
viviansuzano@puc-rio.br

Edo Jelavic
Robotic Systems Lab, ETH Zurich, 8092 Zurich, Switzerland
jelavice@ethz.ch

Marco Antonio Meggiolaro
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
meggi@puc-rio.br

Abstract. *A field of research that has received a lot of funding in the last few years is the development of mobile robots, autonomous or remotely operated, that can perform tasks in rough terrain. In most of these applications, hybrid wheeled-legged robots are well-suited since they combine the high mobility of the legs with the efficiency of the wheels. For such robots, driving motions are particularly interesting, because they allow for faster and more stable locomotion, and obstacle negotiation is still possible through the presence of the legs. When planning for driving motions, common scenarios often include trajectories in terrains with little variation in the lateral direction. Such scenarios can benefit from a simplification of the planning problem and can be handled with a two-dimensional approach. In this context, we present a formulation of the 2D Trajectory Optimization (TO) problem for quadrupedal wheeled-legged robots driving in rough terrain that optimizes over the robot's state and ground reaction forces in a single planar problem. The formulation uses the map of the terrain to optimize the driving motions while taking into account the dynamics of the robot. The main advantage of this approach is the fast planning times, which are achieved by representing the robot with a simplified two-dimensional single rigid body model that still ensures physically feasible motions. The optimization is formulated as a Nonlinear Programming Problem (NLP) and solved using a fast interior-point solver. The suitability of several cost functions is evaluated for the approach in conditions of fast locomotion and high acceleration. The trajectories are validated in simulations with a quadrupedal robot equipped with non-steerable wheels in different challenging scenarios.*

Keywords: *Wheeled-legged robots, Trajectory optimization, Optimal control, 2D model, Rough terrain.*

1. INTRODUCTION

The combination of legs and wheels in mobile robots can be an effective solution for fast locomotion in challenging scenarios, such as industrial sites, farming plantations, subterranean tunnels and extra-planetary surfaces. For most of these cases, a purely driving approach allows for a faster locomotion, while still able to handle abrupt changes in the terrain profile due to the presence of the legs. Furthermore, driving motions are less energy consuming, more stable and there is no need for a complex foothold placement strategy. For these reasons, we focus on planning dynamic driving motions, combined with the use of the actuated legs to overcome obstacles.

When performing driving motion in a quasi-static condition, a purely reactive controller can be employed for adapting to terrain variations by maintaining a constant desired center of mass (CoM) pose or by continuously adjusting the CoM to achieve a statically-stable configuration. In these cases, the environment is usually perceived through proprioceptive sensing and obstacle negotiation is performed on a static step-by-step basis, as presented in (Grand *et al.*, 2010; Reid *et al.*, 2016; Jarrault *et al.*, 2011). Other active articulated wheeled robots that employ similar kinematic reconfigurability methods for traversing uneven terrain with driving motions can be found in (Cordes *et al.*, 2018; Freitas *et al.*, 2010).

Another common approach is to switch between driving and walking based on a map of the environment (Klamt *et al.*, 2018). In this case, driving is performed only in flat terrain, while obstacles are overcome by walking motions. In contrast, the motion planner presented by (Jelavic and Hutter, 2019) combines driving and stepping maneuvers to generate statically-stable motions for wheeled-legged excavators.

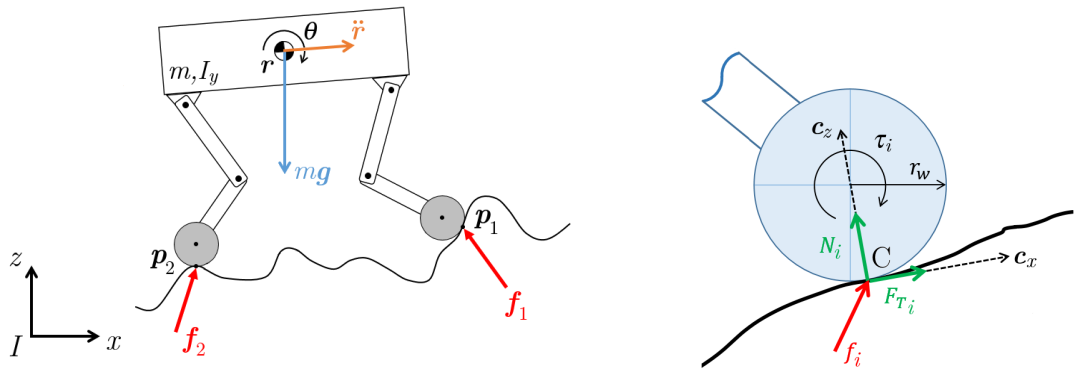
Reactive blind locomotion is implemented for the robot ANYmal with actuated wheels showing successful experimental tests in (Bjelonic *et al.*, 2019). The robot is shown to execute driving motions to overcome slopes and going down steps. The reference trajectories for the robot's CoM are computed by a Zero-Moment Point (ZMP) optimization. However, the motion planner uses a flat terrain assumption, which limits its ability to overcome more abrupt obstacles in driving mode.

Obstacle negotiation with dynamic driving motions can be performed by taking into account the terrain map information and including the ground reaction forces in the optimization problem. The TO presented in (Medeiros *et al.*, 2020) for wheeled-legged robots shows real experiments with the robot ANYmal with wheels negotiating steep steps in various configurations with dynamic driving motions. The framework uses a 3D whole-body planning approach that takes into account the terrain geometry and the robot's stability. The wheeled-legged robots in (Geilinger *et al.*, 2018) have shown dynamic driving motions employing a similar TO approach, but limited to flat terrain.

All of the authors tackled a general planning problem in 3 dimensions. However, for quadrupedal robot driving planning, common scenarios often include little variation in lateral direction. Such scenarios can benefit from a simplification of the planning problem. To this end, we present a TO formulation for a simplified 2D model of the wheeled-legged robot that can overcome challenging terrain with dynamic driving motions at high speeds. The 2D formulation offers several advantages compared to a full fledged 3D problem: 1) lower computational cost, which allows for fast solutions even for trajectories with a long time horizon; 2) easy tuning of the optimization parameters; 3) reduced implementation effort, which helps with the study of suitable stability metrics and cost functions; 4) reduced complexity, which is useful for teaching purposes. Compared to the work presented in (Medeiros *et al.*, 2019), we present the full formulation of the 2D problem and a detailed study on the influence of different cost functions on the solution. Furthermore, we plan trajectories for more challenging obstacles. Lastly, compared to (Medeiros *et al.*, 2020), we reduce the computational cost by more than an order of magnitude for scenarios with no variation in the lateral direction.

2. 2D ROBOT MODEL

For wheeled-legged robots to move safely on challenging terrain, it is important that the motion planner produces only trajectories that are physically feasible. To ensure that, the TO must take into account how the wheels' positions and forces affect the robot's state. In this approach, the robot is approximated by a single rigid-body with mass and inertia located at the robot's CoM, as illustrated in Fig. 1(a).



(a) (b)

Figure 1. (a) Planar model of the ANYmal robot traversing rough terrain; (b) Contact forces on each wheel, considering a frame C_i located at the wheel's contact point with the terrain. The axis $\boldsymbol{c}_z$ is aligned with the terrain normal and $\boldsymbol{c}_x$ is aligned with the rolling direction of the wheel.

In the 2D formulation, the wheels are assumed to be skid-steered, so only forces in the XZ plane are considered. The displacement in the y–direction, and the roll and yaw angles of the robot's base are assumed to be zero for the entire motion. Moreover, we assume that the contact between the wheel and the ground can be reduced to a single virtual contact point and the friction coefficient of the soil is known or can be estimated.

Using this simplified model, the dynamics of the robot is given by:

$$\begin{aligned} m\ddot{\boldsymbol{r}}(t) &= \boldsymbol{f}_1(t) + \boldsymbol{f}_2(t) - m\boldsymbol{g} \\ I_y\ddot{\theta}(t) &= \sum_{i=1}^{2} \boldsymbol{f}_i(t) \times (\boldsymbol{r}(t) - \boldsymbol{p}_i(t)) \end{aligned} \tag{1}$$

where the CoM position and orientation are given by $\boldsymbol{r}$ and θ, respectively. The contact points on the wheels are given by $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, m denotes the robot's mass, $\boldsymbol{g}$ the gravity vector and I_y the inertia of the robot around the Y axis. The contact

forces on each wheel are given by $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$. All the forces and positions vectors are defined with respect to the inertial frame, in $\mathbb{R}^2$. The orientation is simply the pitch angle of the base, defined in $\mathbb{R}$.

Our planner relies on the *a priori* knowledge of the terrain in form of the height map. With the terrain information, the contact force on each wheel $\boldsymbol{f}_i$ can be accurately decomposed into a normal force N_i and a tangential force F_{Ti} (traction force), considering a local frame C_i located in the point of contact between the wheel and the ground, as shown in Fig. 1(b).

Given the terrain normal at the i^{th} wheel contact point $\boldsymbol{n}_i = \begin{bmatrix} n_i^x & n_i^z \end{bmatrix}^T \in \mathbb{R}^2$, the 2D rotation matrix $\boldsymbol{R}_{C_iI}$ from the inertial frame I to the contact frame C_i is given by:

$$\begin{aligned} {}^{C_i}\boldsymbol{f}_i &= \boldsymbol{R}_{C_iI}\, {}^{I}\boldsymbol{f}_i \\ {}^{C_i}\boldsymbol{f}_i &= \begin{bmatrix} F_{Ti} \\ N_i \end{bmatrix} = \begin{bmatrix} n_i^z & -n_i^x \\ n_i^x & n_i^z \end{bmatrix} {}^{I}\boldsymbol{f}_i \end{aligned} \tag{2}$$

For all the variables, the right superscript denotes a component of the vector and the left superscript indicates the coordinate frame, e.g. ${}^{I}f_i^z(t)$ represents the the $z-$component of the contact force on the i^{th} wheel expressed in the inertial frame.

3. PROBLEM FORMULATION

The TO is formulated as an Optimal Control Problem (OCP) of the form:

$$\begin{aligned} \underset{\boldsymbol{x}(t),\boldsymbol{u}(t)}{\text{minimize}} \quad & J(\boldsymbol{x}(t),\boldsymbol{u}(t)) \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t)) \leq \boldsymbol{0}, \quad \boldsymbol{h}(\boldsymbol{x}(t),\boldsymbol{u}(t)) = \boldsymbol{0}, \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0, \quad \boldsymbol{x}(T) = \boldsymbol{x}_f, \end{aligned} \tag{3}$$

where $\boldsymbol{x}(t)$ is the set of variables that defines the motion of the robot's CoM and the wheel's contact positions, given by $\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{r}(t) & \dot{\boldsymbol{r}}(t) & \theta(t) & \dot{\theta}(t) & \boldsymbol{p}_{1,2}(t) & \dot{\boldsymbol{p}}_{1,2}(t) \end{bmatrix}$, and $\boldsymbol{u}(t)$ represents the contact forces on the wheels $\boldsymbol{f}_{1,2}(t)$. The initial and final state of the robot are respectively $\boldsymbol{x}_0$ and $\boldsymbol{x}_f$ and T is the total time duration of the trajectory.

Firstly, this continuous-time optimization problem needs to be formulated as a NLP, with a finite number of decision variables, as follows

$$\begin{aligned} \underset{\boldsymbol{\xi}\in\mathbb{R}^{18n}}{\text{minimize}} \quad & \bar{J}(\boldsymbol{\xi}) \\ \text{subject to} \quad & \bar{\boldsymbol{g}}(\boldsymbol{\xi}) \leq \boldsymbol{0}, \quad \bar{\boldsymbol{h}}(\boldsymbol{\xi}) = \boldsymbol{0} \\ & \boldsymbol{\xi}_L \leq \boldsymbol{\xi} \leq \boldsymbol{\xi}_U \end{aligned} \tag{4}$$

For that, the time horizon is discretized in fixed time intervals ΔT, including the initial state, generating $n = \text{floor}(T/\Delta T) + 1$ nodes. Each node k is defined by the optimization variables at the time $t_k = k\Delta T$ of the trajectory. Thus, the NLP decision variables are the values for the robot's state and inputs on all nodes:

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{u}_0 & \boldsymbol{x}_0 & \boldsymbol{u}_1 & \boldsymbol{x}_1 & ... & \boldsymbol{u}_{n-1} & \boldsymbol{x}_{n-1} \end{bmatrix}^T \tag{5}$$

where $\boldsymbol{u}_k = \boldsymbol{u}(t_k)$ and $\boldsymbol{x}_k = \boldsymbol{x}(t_k)$.

In order to ensure the consistency of the derivatives, the following constraints are imposed between the nodes:

$$\dot{\boldsymbol{q}}_k = \frac{\boldsymbol{q}_{k+1} - \boldsymbol{q}_k}{\Delta T}, \quad k = 0, \ldots, n-2, \tag{6}$$

where $\boldsymbol{q}_k = \begin{bmatrix} \boldsymbol{r} & \theta & \boldsymbol{p}_1 & \boldsymbol{p}_2 \end{bmatrix}_k^T$, the set of position variables at node k.

Having defined the decision variables for the discretized problem, the most straightforward approach is to enforce the equality and inequality constraint at all nodes:

$$\bar{\boldsymbol{g}}(\boldsymbol{\xi}_k) \leq \boldsymbol{0}, \quad k = 0, ..., n-1, \tag{7}$$

$$\bar{\boldsymbol{h}}(\boldsymbol{\xi}_k) = \boldsymbol{0}, \quad k = 0, ..., n-1, \tag{8}$$

where $\boldsymbol{\xi}_k = \begin{bmatrix} \boldsymbol{u}_k & \boldsymbol{x}_k \end{bmatrix}^T$.

Once the problem is solved, the continuous solution for each variable is obtained by a simple linear interpolation between the nodes, which gives a feasible continuous trajectory, assuming ΔT sufficiently small.

The complete TO formulation for the 2D motion optimization problem with all decision variables and constraints is given by

$$
\begin{aligned}
\text{find} \quad & \boldsymbol{r}(t), \dot{\boldsymbol{r}}(t) \in \mathbb{R}^2 && \text{(CoM position)} \\
& \theta(t), \dot{\theta}(t) \in \mathbb{R} && \text{(CoM angle)} \\
& \boldsymbol{p}_{1,2}(t), \dot{\boldsymbol{p}}_{1,2}(t) \in \mathbb{R}^2 && \text{(wheels' positions)} \\
& \boldsymbol{f}_{1,2}(t) \in \mathbb{R}^2 && \text{(wheels' forces)} \\
\text{s.t.} \quad & [\boldsymbol{r}, \theta](0) = [\boldsymbol{r}_0, \theta_0] && \text{(initial state)} \\
& [\boldsymbol{r}, \theta](T) = [\boldsymbol{r}_g, \theta_g] && \text{(goal state)} \\
& [\ddot{\boldsymbol{r}}, \ddot{\theta}]^T = \boldsymbol{F}_d(\boldsymbol{r}, \boldsymbol{\theta}, \boldsymbol{p}_{1,2}, \boldsymbol{f}_{1,2}), && \text{(dynamic model)} \\
& \alpha(\boldsymbol{r}, \theta, \boldsymbol{p}_{1,2}, \boldsymbol{f}_{1,2}) > 0, && \text{(stability measure)} \\
& \text{for } i = 1, 2: \\
& \quad \boldsymbol{p}_i(t) \in \mathcal{R}_i(\boldsymbol{r}(t), \theta(t)) && \text{(kinematic constraints)} \\
& \quad p_i^z(t) = h_{terrain}(p_i^x(t)) && \text{(terrain height)} \\
& \quad N_i(t) \geq 0 && \text{(normal force)} \\
& \quad -\tau_{max}/r_w \leq F_{Ti}(t) \leq \tau_{max}/r_w && \text{(maximum torque)} \\
& \quad -\mu N_i(t) \leq F_{Ti}(t) \leq \mu N_i(t) && \text{(no slippage)}
\end{aligned}
\tag{9}
$$

As an additional note, a similar formulation can also be employed for torque optimization in hazardous conditions, in which the robot is close to losing stability and tipping over, as demonstrated for wheeled robots in (Medeiros *et al.*, 2019).

3.1 Constraints

The 2D formulation is only applicable for generating driving motions, which implies that the wheels are in contact with the terrain during the entire trajectory. This can be translated in the problem formulation as constraints on the wheel's contact positions and on the normal contact forces, as:

$$p_i^z = h_{terrain}(p_i^x), \quad i = 1, 2 \tag{10}$$

$$N_i \geq 0, \quad i = 1, 2 \tag{11}$$

where $h_{terrain}$ is the continuous 2D height map of the terrain. The normal component N_i is extracted from the contact force $\boldsymbol{f}_i$ by applying the transform presented in (2). For that, the terrain normal at the contact position $\boldsymbol{n}(p_i^x)$ is obtained through the derivative of the continuous terrain map at that point.

In order to ensure no slippage in the wheels, the traction force is constrained to be less or equal than the normal force times the friction coefficient μ of the terrain:

$$-\mu N_i \leq F_{Ti} \leq \mu N_i, \quad i = 1, 2 \tag{12}$$

where F_{Ti} is obtained from (2) as well. Additionally, the traction forces are limited to a saturation value correspondent to the maximum torque of the wheel's motor. Torque τ in each wheel can be obtained by multiplying the traction force for the wheel radius r_w.

The dynamic consistency of the trajectory is ensured by imposing the Eq. (1) to all nodes. For that, the base accelerations are computed via numerical differentiation of the base velocities. The base accelerations are also constrained to a maximum value for avoiding abrupt motions. In most cases, the velocities for the initial and final nodes are constrained to be zero to prevent any feedforward torque at the beginning or end of the motion.

An additional requirement for motion consistency is to ensure the wheels' positions remain within the reachable workspace of the legs. The exact way to do this is to enforce the joint limits and compute the end-effector position through forward kinematics. However, forward kinematics introduces more non-linear constraints on the formulation, which could affect the solver's performance. Instead, the feasible workspace for each wheel is represented by a rectangle of fixed size centered in the nominal position of the wheels relative to the robot's base, similar to (Medeiros *et al.*, 2020), as depicted by the cyan-marked regions $\mathcal{R}_{1,2}$ in Fig. 2. The size of the rectangle is defined based on the robot's joints limits and, for this paper, was defined as 0.15 m length by 0.10 m height.

Lastly, the planner takes into account the stability of the robot along the trajectory, enabling motion in complex and highly uneven terrain. The stability criterion used for the optimization is based on the Force-Angle Stability Measure presented in (Papadopoulos and Rey, 2000), adapted for a planar case. Considering the illustration shown in Fig. 2, the stability measure α is given by the minimum of the two angles β_1 and β_2. Letting $\hat{\boldsymbol{v}} = \boldsymbol{v}/||\boldsymbol{v}||$ for any vector $\boldsymbol{v}$, the

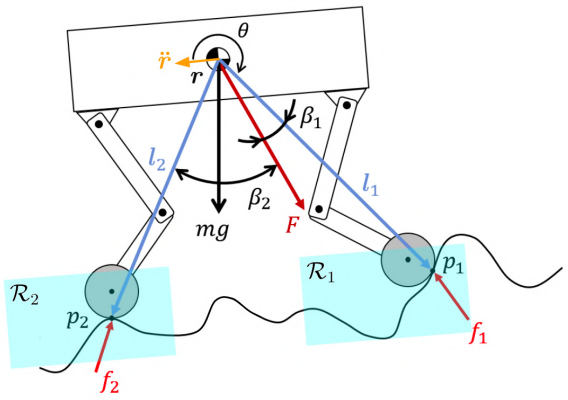


Figure 2. Illustration of the stability measure for the planar case.

tipover angles β_1 and β_2 and the stability measure α are computed by:

$$\beta_i = \cos^{-1}(\hat{\boldsymbol{F}} \cdot \hat{\boldsymbol{l}}_i), \quad i = 1, 2 \tag{13}$$

$$\alpha = \min(\beta_1, \beta_2), \tag{14}$$

where $\boldsymbol{l}_i$ are the tipover axis normals and $\boldsymbol{F}$ is the sum of all forces acting on the robot's CoM that contribute to a tipover motion instability, including gravitational loads, inertial forces[1] and external disturbances.

If the total force $\boldsymbol{F}$ is coincident with either one of the vectors $\boldsymbol{l}_{1,2}$, the measure α becomes zero and the robot is on the eminence of tipping over. When $\alpha < 0$, the robot is already in a tipover condition. Thus, it is desirable that the value of α remains large and positive, to maintain the robot in the stability region. Except for the case where it is explicitly maximized, the stability measure is added as an inequality constraint in the optimization problem, being constrained to remain above a minimum threshold of $6.3°$. The threshold limit is chosen arbitrarily based on empirical testing and on the results presented in (Medeiros *et al.*, 2020).

4. RESULTS

This section discusses the implementation and testing of several motions generated by the 2D motion planning framework. The presented 2D TO formulation was tested in different terrain types, including a half-pipe, slopes, a step 0.2 m high, among others. An initial assessment of the 2D trajectories is performed via visualization in MATLAB®, with a script that illustrates the robot motion together with the contact forces on the wheels, the base acceleration and the stability margin. Further, the 2D trajectories are validated in physical simulations using a real model of the ANYmal robot.

4.1 Implementation

The 2D TO was implemented in C++ using the Ifopt (Winkler, 2018) interface for the interior-point solver Ipopt (Wächter and Biegler, 2006). Ifopt provides an intuitive interface for constructing the optimization problem that allows the user to define independent sets of variables and constraints, and the overall problem is automatically built from these sets using an automatic index management feature. This way, each node of the trajectory is defined as a variable-set that all the constraints are imposed on, rather then defining one huge set with all the variables. The derivatives (Jacobians) for all the constraints and costs are provided analytically, which significantly reduces the computational cost of the solver. The time interval between the nodes ΔT was chosen as 0.1 s, which is refined enough to ensure dynamically consistent motions. The position of the base and the wheels on each node are initialized by a linear interpolation between the initial and final positions of the robot. The contact forces are initialized with the same value at all nodes: the maximum allowed traction force in the x–direction and half the weight of the robot's CoM in the z–direction.

4.2 Suitable Cost Functions

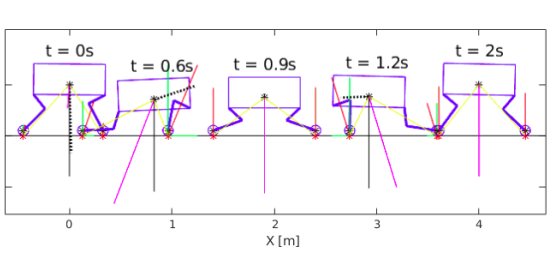
The use of objective functions in the optimization can increase robustness and quality of the motion. Since our problem formulation includes all the forces and positions of the wheels, the objective function for the TO can be numerous: maximize the stability measure, to avoid tipping and loss of wheel contact with the ground; minimize the traction difference between the wheels, which can be very helpful in conditions where the friction coefficient of the terrain changes abruptly; or minimize the difference between the wheels normal forces to improve stability in uneven terrains. However, the use

[1]The fictitious force that appears to act on an object due to its inertia when the frame of reference used to describe the object's motion is accelerating compared to a non-accelerating frame.

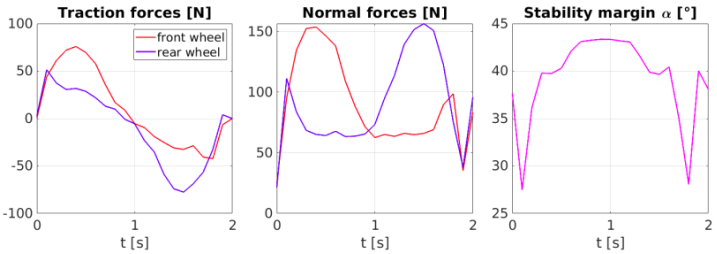
of a cost function can be highly costly and increase the amount of tuning parameters for the optimization, especially in this formulation, where the value of the objective function must be computed as a sum of the costs of all nodes. For that reason, we also evaluate the case with no cost function to minimize, i.e., a *feasibility problem*. The objective is simply to find the decision variables that fulfill all the constraints (Winkler *et al.*, 2018; Medeiros *et al.*, 2020).

Several tests are conducted in flat terrain to evaluate how the use of each cost function would affect the trajectories and the computational cost of the framework. Four objectives were considered for the analysis: minimize the difference between the traction forces on the wheels $\left(\bar{J} = \sum (F_{T_1} - F_{T_2})_k^2\right)$, minimize the difference between the normal forces on the wheels $\left(\bar{J} = \sum (N_1 - N_2)_k^2\right)$, maximize stability $\left(\bar{J} = -\sum \alpha_k\right)$, and no objective function $\left(\bar{J} = 0\right)$.

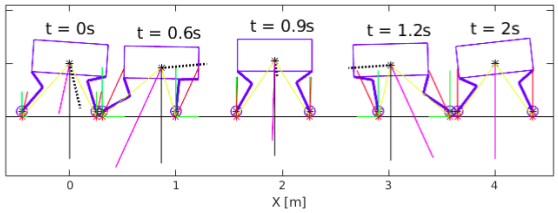
In flat terrain, tipover instability can occur at high acceleration conditions, which is why the acceleration limits for the base and the wheels are set to ± 10 m/s^2 for the following tests. Figure 3 shows the optimization results for each cost function considering the robot moving on a flat terrain from $x = 0.0$ m to $x = 4.0$ m in 2.0 s, starting and finishing at zero velocity. In the 2D optimized trajectory, the red lines indicate the contact forces, the green lines are the decomposed normal and traction forces, the back full lines are the gravity force and the black dotted lines indicate the direction of the CoM's acceleration. The pink line is the total force acting on the robot's CoM and the yellow lines represent the tipover axis normals, the limits for stability.



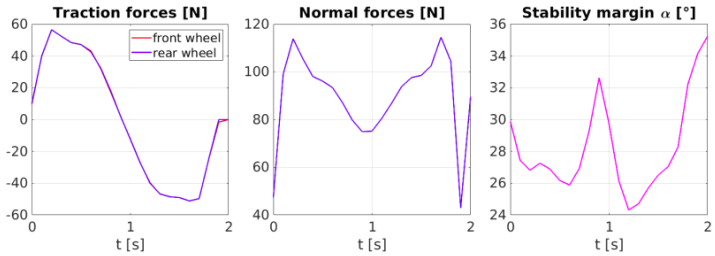
(a) 2D trajectory for maximizing stability.



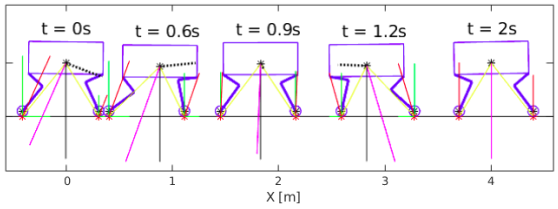
(b) Trajectory plots for the stability cost.



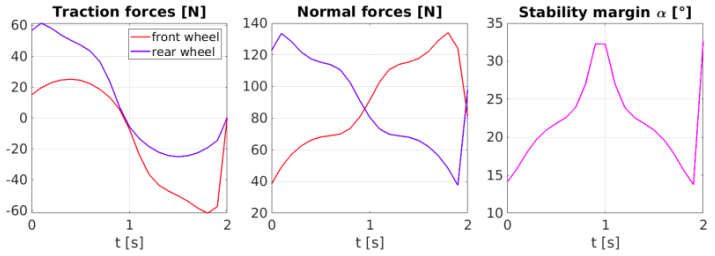
(c) 2D trajectory for minimizing the difference on the wheel's normal forces.



(d) Trajectory plots for the cost on the wheel's normal forces.



(e) 2D trajectory without a cost function.



(f) Trajectory plots for no cost function.

Figure 3. Optimized trajectories for each proposed cost function.

The effect of each cost function on the robot's motion can be visually identified from the results. When maximizing stability (Fig. 3(a)), the CoM is maintained at a lower height and the wheels are positioned farther from each other to increase the area of the polygon defined by the contact points. When accelerating (t = 0.6 s), the base is moved forward and the torque is higher on the front wheel. Similarly, the base is moved backwards when de-accelerating (t = 1.2 s), and the torque is higher on the hind wheel. Furthermore, Fig. 3(b) shows that the stability margin never reaches a value lower than 27° for the entire motion.

In contrast, there was no significant difference between the results for minimizing the traction forces or the normal forces on the wheels. The motion was similar for both cases and it is depicted in Fig. 3(c). In Fig. 3(d), the difference between the normal forces on the front wheel and the hind wheels is so small that is imperceptible on the plot. In addition, the traction forces on both wheels are also almost the same. As it can be seen, the wheels' positions and the base orientation are constantly adjusted to keep a more evenly distribution of forces between the wheels. As a consequence,

the stability measure also remains positively large along the trajectory.

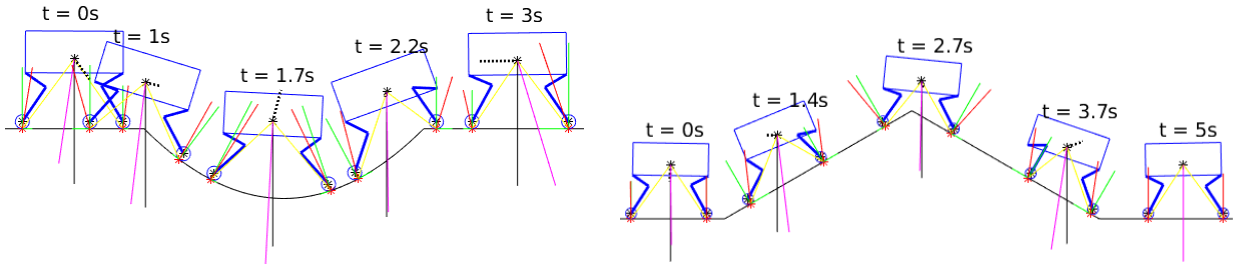
Lastly, Fig. 3(e) shows the trajectory generated for the feasibility problem. In this case, since there is no cost function, the solver converges to the closest feasible solution that fulfills all the constraints. In general, the motion presents a lower average stability margin during the motion. The traction force is higher on the hind wheel when accelerating and higher in the front wheel when braking, which contributes to the lower stability margin, but it is still within the limits.

The computational time[2] that it takes to compute the trajectory in each case is also an important factor to be evaluated, especially considering that the flat terrain is the simplest case and the computational cost will most likely be higher for more complex terrains. The TO for flat terrain took 0.075 s without a cost function, 0.114 s when minimizing the difference between the normal forces, and 0.622 s for maximizing stability. Optimizing the stability measure adds a non-linear non-convex cost function to the problem, which causes the computational cost of the framework to increase in over 8 times compared to not using an objective function.

The general goal of a motion planning framework is to generate a physically feasible trajectory for the system, which is well accomplished by solving the feasibility problem and it is much faster. For that reason, this is the approach used for the TO problem in this work. The inclusion of the stability measure is maintained as an inequality constraint to ensure safety in all conditions.

4.3 Trajectories

Once the best approach for the cost function was investigated, further testing was carried out in different terrain types. Figure 4 shows the motions for the half-pipe terrain and the slope terrain. In both cases, the terrain profile has positive and negative slopes that require the robot to adjust its behavior accordingly for traversing it. For the half-pipe terrain, there is an abrupt descend at the beginning of the motion and the robot starts adjusting the positions of the CoM and the front wheel prior to the terrain decline, which ensures a stable transition. This is only possible because the motion planner takes into account the terrain information and the dynamics of the robot, which allows for the robot to be already in a proper configuration when faced with the obstacle. In such cases, a purely reactive controller could fail in adjusting the robot's configuration in time to prevent a tipover condition, which is why several reactive approaches employ quasi-static assumptions and low speed limits for the motions (Grand *et al.*, 2010; Turker *et al.*, 2012; Cordes *et al.*, 2018). The same could happen on the abrupt change from positive to negative slope in the second terrain.



(a) 2D motion for the parabola-modeled half-pipe terrain. (b) 2D motion for the increasing/decreasing slope.

Figure 4. 2D motions for terrains with ascending and descending trajectories.

The same predictive behavior is easily perceived when the robot has to drive up steep obstacles, such as the $76°$ step depicted in Figure 5. For such maneuvers, an optimization that includes the ground reaction forces is particularly important. Note how the robot's CoM is already in a lower position and closer to the hind wheels when reaching the step, so it has enough traction to move the front wheels up. Such motions cannot be generated without knowledge of the terrain. In fact, attempts to overcome such terrain with a purely reactive controller have failed (Medeiros *et al.*, 2020).

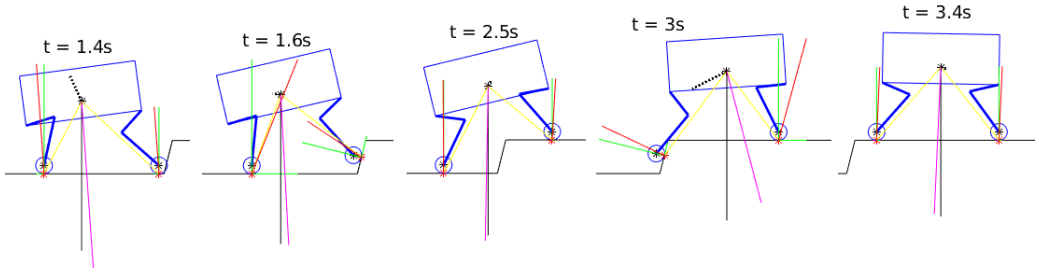


Figure 5. The 2D motion for the robot driving up a step 0.2 m high and a $76°$ slope.

[2]All computational times stated in this work were obtained on a processor Intel® Core™ i7-7500U, 2.7GHz, dual-core, 64-bit.

Table 1 presents a brief description of the terrains used for the tests and the respective computation time for planning the trajectory, given a time horizon and the desired final position for the robot. The solver computation time for the planner depends on the complexity of the terrain, but the average was 34.2 ms per second of trajectory. This represents a Real-Time Factor (RTF) of 30.6, which means the computational time for the solver is in average 30.6 times shorter than the planning horizon. Compared with the results obtained for the 3D trajectory optimization for similar terrain types (with slightly less challenging slopes) presented in (Medeiros *et al.*, 2020), there is an increase of over 10 times the RTF of the framework, as a trade-off for the limitation in the number of applicable terrains.

Table 1. Different terrains used to test the 2D TO framework.

Terrain Description	Traj. Time	Dist. (in x)	Comp. Time
Flat terrain	2.0 s	4.0 m	0.056 s
Step with a 0.2 m height with a 76° slope	4.5 s	2.0 m	0.121 s
Ascending sine function with 5.0 rad/s frequency and 0.1 m amplitude	4.0 s	4.0 m	0.116 s
Increasing and decreasing ramp with 30° slope and 2.0 m length	5.0 s	5.0 m	0.220 s
Parabola-modeled gap with 0.5 m height and 2.0 width	3.2 s	3.2 m	0.139 s

4.4 From 2D to 3D motions

For applications with the actual robot, the 2D motions can be transcribed into 3D by fixing the y-coordinate of the robot's CoM, as well as the roll and yaw angles of the base, as shown in Fig. 6. For the wheels' positions, the default stance y-position is maintained for the entire motion. The trajectory of the wheels is assumed to be mirrored between the left and right wheels, which means the optimized contact forces are divided by two between them, keeping no contact force in the lateral direction.

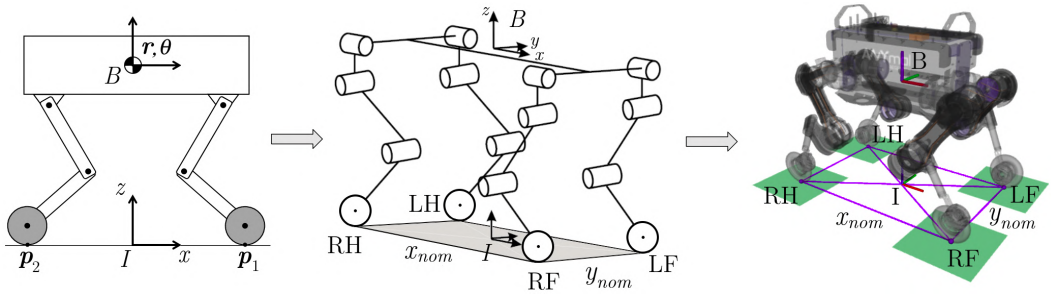


Figure 6. Trajectories for the planar case transcribed into 3D trajectories for applications with the actual robot. For the wheels, the first letter indicates the left (L) or right (R) and the second, indicates front (F) or hind (H).

Thus, the 6D base motion, composed by its position ${}^I\boldsymbol{r}_B(t) \in \mathbb{R}^3$ and orientation $\boldsymbol{\Phi}_{IB}(t) \in \mathbb{R}^3$ (represented in Euler angles), is given by:

$$\begin{aligned} {}^I\boldsymbol{r}_B(t) &= [r^x(t) \quad r_0^y \quad r^z(t)]^T, \\ \boldsymbol{\Phi}_{IB}(t) &= [0 \quad \theta(t) \quad 0]^T, \end{aligned} \tag{15}$$

where ${}^I\boldsymbol{r}_0 \in \mathbb{R}^3$ is the initial position for the base, usually given by ${}^I\boldsymbol{r}_0 = [0 \quad 0 \quad z_0]^T$.

The wheels' motions are defined as

$$\begin{aligned} {}^I\boldsymbol{p}_{LF}(t) &= [p_1^x(t) \quad r_0^y - y_{nom}/2 \quad p_1^z(t)]^T, \quad {}^I\boldsymbol{p}_{LH}(t) = [p_2^x(t) \quad r_0^y - y_{nom}/2 \quad p_2^z(t)]^T, \\ {}^I\boldsymbol{p}_{RF}(t) &= [p_1^x(t) \quad r_0^y + y_{nom}/2 \quad p_1^z(t)]^T, \quad {}^I\boldsymbol{p}_{RH}(t) = [p_2^x(t) \quad r_0^y + y_{nom}/2 \quad p_2^z(t)]^T \end{aligned} \tag{16}$$

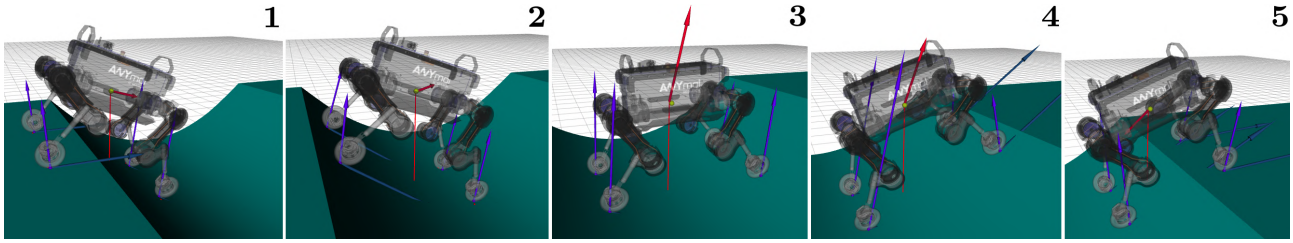
where y_{nom} is the nominal distance between the left and right wheels.

Lastly, since the robot's CoM does not move in the y-direction, the wheels' contact forces are equally divided between left and right wheels:

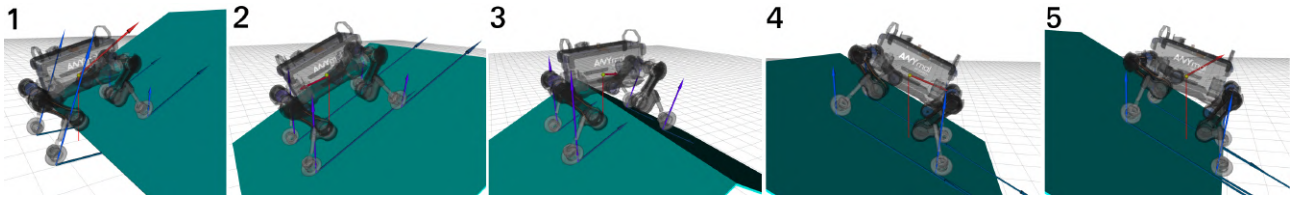
$$\begin{aligned} {}^I\boldsymbol{f}_{LF}(t) &= {}^I\boldsymbol{f}_{RF}(t) = [f_1^x(t)/2 \quad 0 \quad f_1^z(t)/2]^T \\ {}^I\boldsymbol{f}_{LH}(t) &= {}^I\boldsymbol{f}_{RH}(t) = [f_2^x(t)/2 \quad 0 \quad f_2^z(t)/2]^T \end{aligned} \tag{17}$$

In order to verify the physical feasibility of the optimized trajectories, several simulations are carried out in the robot simulation environment Gazebo (Koenig and Howard, 2004), using the full rigid body dynamics of the real quadrupedal robot ANYmal, equipped with actuated non-steerable wheels. The transcribed 3D trajectories are tracked by the hierarchical whole-body controller (WBC) described in (Bjelonic *et al.*, 2019), that computes the actuation torques for the joints and the wheels while accounting for the actuator limits, ensuring realistic results. A video showing the 3D simulation results is available at `https://drive.google.com/file/d/15M4Wl6WLkGthsPsS__nyiePXYoMmdv-m/view`.

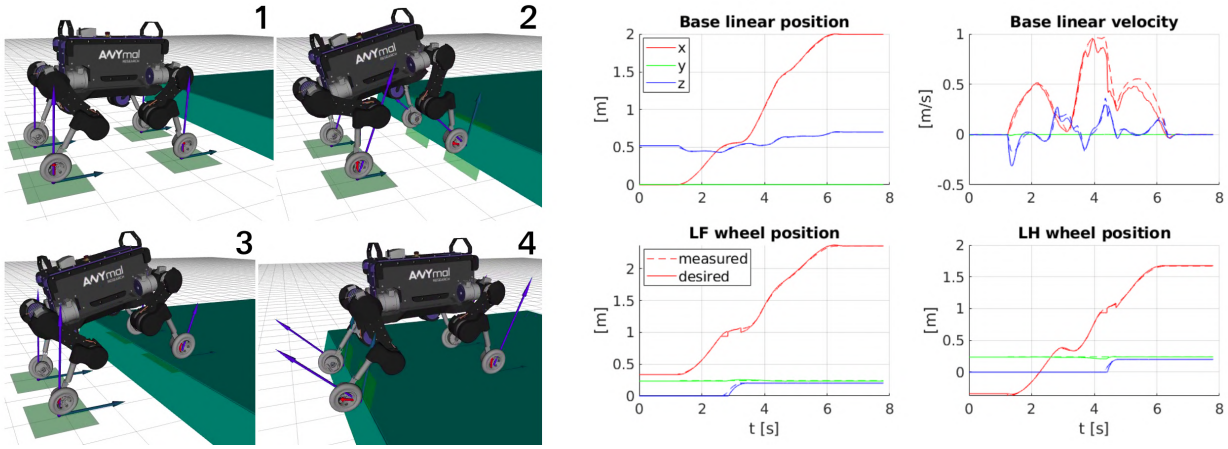
Figure 7 shows snapshots of the simulations for different scenarios. In Fig. 7(a), the robot is crossing a 0.5 m deep half-pipe at an average speed of 1.0 m/s while maintained an stable base position. Figure 7(b) shows the robot driving over the two consecutive ramps with $30°$ slope at an average speed of 1.0 m/s. As expected, the robot adjust its speed when reaching the peak of the slopes so it can safely adapt to the abrupt change in the inclination. Figure 7(c) shows the robot overcoming a $76°$ step with a driving motion. This maneuver, in particular, shows the importance of the optimization of the contact forces in the TO, since the robot relies strongly on the traction forces on the wheels to bring itself up the step. For all the motions, the WBC was able to successfully track the desired trajectories with a Root-Mean-Square-Error (RMSE) of 5 mm for the base and 14 mm for the wheels, as shown by the plots in Fig. 7(c).



(a) The ANYmal robot driving over a 0.5 m half-pipe at an average speed of 1.0 m/s.



(b) Simulation of the robot driving over two increasing and then decreasing ramps with a $30°$ slope.



(c) On the left, simulation results for the robot driving up a step 0.2 m high with a $76°$ slope at an average speed of 0.45 m/s. On the right, the desired trajectories are compared with the measured positions obtained from the simulations, confirming the successful tracking of the WBC.

Figure 7. Simulation results for the optimized trajectories on the real robot. The light blue arrows on the wheels are the contact forces, the dark blue arrows are the wheels' linear velocity, and the red arrow is the base's linear acceleration.

5. CONCLUSION

The presented 2D TO framework is able to generate optimized motions and interaction forces for wheeled-legged robots driving over challenging terrain by taking into account the robot's dynamics, the terrain profile and the stability of the robot along the trajectory. A study on possible cost functions for the optimization confirmed that the solution for the feasibility problem (no cost function) is just as effective for motion planning and much faster than the proposed

alternatives.

Physical simulations of the trajectories with the real robot demonstrated the dynamic consistency of the motion plans in conditions with no displacement in the lateral direction. For such conditions, a simplified 2D model allows for a fast computation time, less than 35 ms/s of trajectory in average, which makes it suitable for a future online implementation in its current form. Furthermore, similar planned motions have been successfully tested in real world experiments with the same robot in (Medeiros *et al.*, 2020), indicating that our approach should also present good results in experimental tests.

Further work includes the development of an online trajectory adaptation framework to increase robustness against external disturbances using Model Predictive Control (MPC).

6. REFERENCES

Bjelonic, M., Bellicoso, C.D., de Viragh, Y., Sako, D., Tresoldi, F.D., Jenelten, F. and Hutter, M., 2019. "Keep rollin'—whole-body motion control and planning for wheeled quadrupedal robots". *IEEE Robotics and Automation Letters*, Vol. 4, No. 2, pp. 2116–2123.

Cordes, F., Kirchner, F. and Babu, A., 2018. "Design and field testing of a rover with an actively articulated suspension system in a mars analog terrain". *Journal of Field Robotics*, Vol. 35, No. 7, pp. 1149–1181.

Freitas, G., Gleizer, G., Lizarralde, F., Hsu, L. and dos Reis, N.R.S., 2010. "Kinematic reconfigurability control for an environmental mobile robot operating in the amazon rain forest". *Journal of Field Robotics*, Vol. 27, No. 2, pp. 197–216. doi:10.1002/rob.20334.

Geilinger, M., Poranne, R., Desai, R., Thomaszewski, B. and Coros, S., 2018. "Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels". In A.T. on Graphics (TOG), ed., *Proceedings of ACM SIGGRAPH*. ACM, Vol. 37.

Grand, C., Benamar, F. and Plumet, F., 2010. "Motion kinematics analysis of wheeled–legged rover over 3d surface with posture adaptation". *Mechanism and Machine Theory*, Vol. 45, No. 3, pp. 477–495.

Jarrault, P., Grand, C. and Bidaud, P., 2011. "Robust obstacle crossing of a wheel-legged mobile robot using minimax force distribution and self-reconfiguration". In *2011 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. pp. 2753–2758. ISSN 2153-0858.

Jelavic, E. and Hutter, M., 2019. "Whole-body motion planning for walking excavators". In *2019 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

Klamt, T., Rodriguez, D., Schwarz, M., Lenz, C., Pavlichenko, D., Droeschel, D. and Behnke, S., 2018. "Supervised autonomous locomotion and manipulation for disaster response with a centaur-like robot". In *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. pp. 1–8. ISSN 2153-0866.

Koenig, N. and Howard, A., 2004. "Design and use paradigms for gazebo, an open-source multi-robot simulator". In *2004 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3, pp. 2149–2154 vol.3.

Medeiros, V.S., Jelavic, E., Bjelonic, M., Siegwart, R., Meggiolaro, M.A. and Hutter, M., 2020. "Trajectory optimization for wheeled-legged quadrupedal robots driving in challenging terrain". *IEEE Robotics and Automation Letters*, Vol. 5, No. 3, pp. 4172–4179.

Medeiros, V.S., Rosa, D.G.G. and Meggiolaro, M.A., 2019. "Torque optimization for stability control of wheeled vehicles in rough terrain". In *XVIII International Symposium on Dynamic Problems of Mechanics (DINAME 2019)*.

Papadopoulos, E. and Rey, D., 2000. "The force-angle measure of tipover stability margin for mobile manipulators". *Vehicle System Dynamics - VEH SYST DYN*, Vol. 33, pp. 29–48.

Reid, W., Pérez-Grau, F.J., Göktoğan, A.H. and Sukkarieh, S., 2016. "Actively articulated suspension for a wheel-on-leg rover operating on a martian analog surface". In *2016 IEEE Int. Conf. on Robotics and Automation (ICRA)*. pp. 5596–5602.

Turker, K., Sharf, I. and Trentini, M., 2012. "Step negotiation with wheel traction: a strategy for a wheel-legged robot". In *2012 IEEE Int. Conf. on Robotics and Automation*. pp. 1168–1174. ISSN 1050-4729.

Wächter, A. and Biegler, L.T., 2006. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". *Mathematical Programming*, Vol. 106, pp. 25–57. ISSN 1436-4646.

Winkler, A.W., Bellicoso, C.D., Hutter, M. and Buchli, J., 2018. "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization". *IEEE Robotics and Automation Letters*, Vol. 3, No. 3, pp. 1560–1567. ISSN 2377-3766.

Winkler, A.W., 2018. "Ifopt - A modern, light-weight, Eigen-based C++ interface to Nonlinear Programming solvers Ipopt and Snopt."

7. RESPONSIBILITY NOTICE

The authors are solely responsible for the printed material included in this paper.