REAL-TIME THREE-DIMENSIONAL DYNAMIC SIMULATION OF MOBILE ROBOTS IN ROUGH TERRAIN

Ricardo Morrot, morrot@gmail.com
Pedro F. C. Blois de Assis, prblois@gmail.com
Marco Antonio Meggiolaro, meggi@puc-rio.br
Pontifical Catholic University of Rio de Janeiro – PUC Rio, http://www.puc-rio.br/

Abstract. *The present work introduces the development of a 3D real-time dynamic simulator of robotic vehicles in rough terrain. An intersection algorithm is developed between a 3D generic terrain and each wheel of a vehicle. A tire-soil contact force model is implemented combining longitudinal and lateral slip conditions. The model includes the effects of direct current motors, interaction between its mechanical and electric parts, a continuous approximation of LuGre's friction model applied to its shaft, and power limitations of the system batteries. The simulator also allows the implementation of a user defined control algorithm with the necessary virtual sensors. Equations for a 2D stability control, taking into consideration just the stabilization of the pitch angle of the vehicle were implemented as an example and its results have shown agreement to the ones found in the article that proposed the control algorithm. The simulator is validated using two scenarios where analytical solutions are known. A maximum error of 1.23% is found between the analytical and simulated trajectories, but it is shown that this error might be decreased if the geometric representation of the wheel is refined, at a price of an increased computational cost. This enhancement is easily programmed in the software through a script file, which also defines other settings such as geometric and dynamic parameters of the vehicle and tires, terrain types and their spatial resolution, the simulation time step, and output files for analyses.*

Keywords*: Simulation, Real-Time, 3D, Rough Terrain, Traction Control*

1. INTRODUCTION

Numerical simulations are almost always associated with high-performance computers. Thus, research groups that had little financial resources could not get workable results, because in order to make possible a more realistic modeling of the studied system, a very large amount of processing was needed, which greatly increased the cost. With the implementation in 1995 of the first computer clusters (clusters of private computers working in parallel to solve a certain task), the cost of processing, compared to supercomputers, reduced significantly (Chee et al., 2006).

All that computational progress in recent years has also favored Robotics. Thousands of microprocessors, microcontrollers, and integrated circuits of various types have enabled a great technological explosion in the creation of robots with devices and tools available to any user. However, the development of cutting-edge mobile robots, especially rovers, still has a high cost in development and production. With the aid of modern computer systems, virtual simulations of these robots can assist in the design and development of control algorithms.

The robots for exploration of other planets have achieved notoriety in studies of virtual simulations in Robotics (Iagnemma and Dubowsky, 2004). But equally important are rovers designed to explore our own planet. There are areas with surfaces as intriguing and challenging as those found in other planets. When observing more carefully our planet, and specifically in Brazil, one realizes one of the richest biomes with savannas, forests, swamps, among others. All this becomes a big challenge for companies based in those regions that need constant monitoring.

One important region is the Amazon (Silva, 2007), for it presents a rich structure of soils, forests and thousands of rivers. Overcoming the barriers imposed by different types of soil is a very big challenge.

Petrobras, through the project Cognitus (Silva, 2007) has developed a series of robots for monitoring, called "AmazonBots." These robots are designed with the main goal of monitoring pipelines in a region of difficult access, where the prevention of environmental risks is regarded as high priority.

Current pieces of software are not yet capable of performing real-time simulation of robotic vehicles in rough terrain, with the possibility of implementing advanced techniques for stability and traction control. Here, rough terrain is defined as the one where its surface gradients may significantly vary between each robot wheel. Current simulators are usually only capable of modeling smoother terrains. Research in the field of dynamic behaviors of vehicles in rough terrains enables operational cost savings, since the entire control development phase can be performed on a virtual environment.

An example of applicability of this work is to develop software for virtual simulation of the Hybrid Environmental Robot (HER, see Figure 1), developed by CENPES / PETROBRAS as part of the project Cognitus. This robot is used to monitor the Amazon region through which the Coari-Manaus gas pipeline passes, with 420 km in extension (Silva, 2007), requiring an advanced algorithm for traction and stability control.

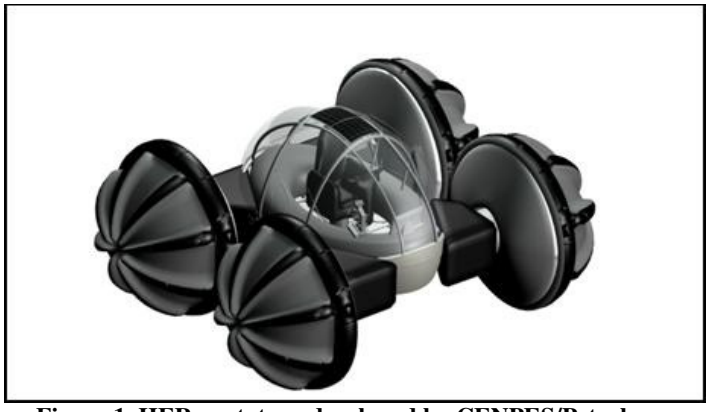

Figure 1. HER prototype developed by CENPES/Petrobras.

2. SIMULATOR

A C++ software to dynamically simulate mobile robots in rough terrain has been developed, called VirtualBotz 3D Robotic Vehicle Simulator (Morrot, 2010), using the OpenGL graphic interface. It aims to provide a means to simulate such systems with high degree of accuracy, to be able to properly capture the entire non-linear behavior, which has a great influence in the robot's response at high speeds. One of the implemented virtual mobile robot models in the software reproduces the kinematic and dynamic behavior of the VIVI robot (Vehicle for Internal Visual Inspection of pipelines, Figure 2) developed by the Robotics Laboratory at PUC-Rio. The robotic vehicle was assumed be skid-steered in order to replicate the behavior of the VIVI robot. The dynamic model used to calculate the forces acting on the robotic vehicle does not consider that the wheels can be steered. Future implementations of the software will include steerable wheels. Therefore, in the current simulator, all wheels will always be parallel to each other.

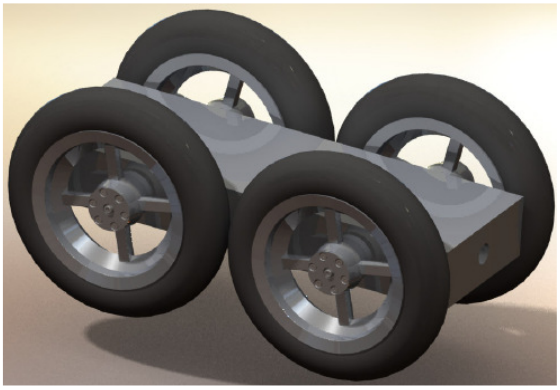

Figure 2. Vehicle for Internal Visual Inspection of pipelines (VIVI).

Besides allowing the 3D visual representation in real-time of the interactions between mobile robots and several types of terrain, the simulator also features the possibility to:

- implement user-defined control algorithms;
- implement different DC motor models, through the setting of its parameters in a script file in text format;
- import terrain profiles through an elevation matrix or heights map, or through user-defined functions;
- import texture files for the terrain and scale settings through the simulator script;
- import different visualization masks for robot parts, such as chassis and wheels, through wavefront files, including visual details of each material;
- interact with the robotic vehicle in the virtual world in real-time through a joystick;
- change the viewpoint through keyboard or joystick;
- set different virtual cameras through the keyboard or joystick, allowing the visualization of different angles;
- save system data such as speeds, accelerations, torques, forces, etc., over time in text files for further analysis of the control algorithm or system dynamic behavior;
- record the simulation in a video file in different types and formats;
- visualize the vehicle's path during the simulation with vectors or continuous lines;
- show the vehicle in a specific point of its path after the end of the simulation;

- reserve an exclusive microprocessor core to process the dynamic equations (optional, for Dual Core or multiple core microprocessors); and
- adjust the frequency of the control loop.

2.1. Terrain Model

The model used to build the terrain is the Rectangle Regular Grid, which can be created from the set of ground samples or by any geometric 3D modeler that stores data in a matrix of elevations. It is specified by a two-dimensional matrix called $\underline{M}$, where each element in it corresponds to the z coordinate (terrain surface height) of a certain x-y coordinate system in a horizontal plane. The boundaries of the represented terrain in VirtualBotz 3D (Figure 3) are the coordinates (*xmin*, *xmax*) and (*ymin*, *ymax*), where

$$x = x_{\min} + i\frac{(x_{\max} - x_{\min})}{(n_x - 1)} \tag{1}$$

$$y = y_{\min} + j\frac{(y_{\max} - y_{\min})}{(n_y - 1)} \tag{2}$$

where n_x and n_y are respectively the number of lines and columns of $\underline{M}$ and *(i,j)* is the element of $\underline{M}$ from line *i* and colum *j*.

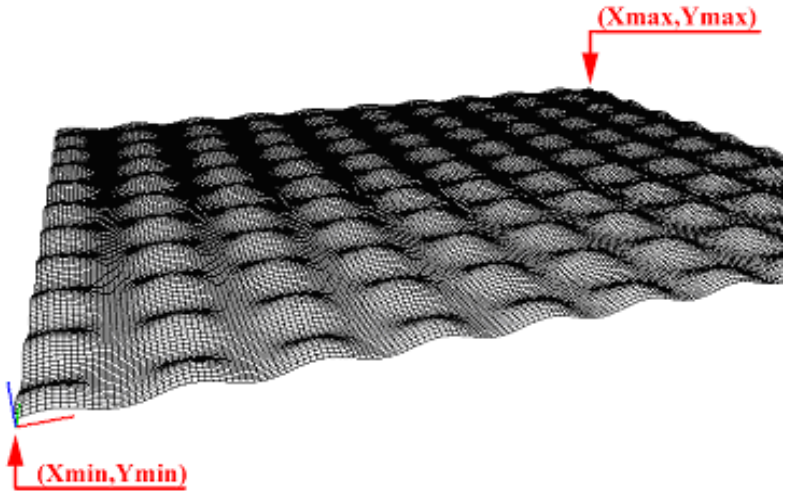

Figure 3. Terrain boundaries in VirtualBotz 3D.

Any given point of the terrain surface is calculated through a linear interpolation algorithm that uses the four points of $\underline{M}$ that are closest to the desired point (x, y). The interpolation equations are

$$z_1 = M_{ix+1,iy}(aux_x - i_x) + M_{ix,iy}(1 - (aux_x - i_x)) \tag{3}$$

$$z_2 = M_{ix+1,iy+1}(aux_x - i_x) + M_{ix,iy+1}(1 - (aux_x - i_x)) \tag{4}$$

$$z_f = z_2(aux_y - i_y) + z_1(1 - (aux_y - i_y)) \tag{5}$$

Where: (i_x, i_y) are the coordinates of the element of $\underline{M}$ associated with a point immediately before the desired point, and aux_x and aux_y are the coordinates of the desired point converted to the coordinate system of $\underline{M}$; $\underline{M}_{i,j}$ is the element (*i,j*) of $\underline{M}$; and z_f is the result of the interpolation.

2.2. Robotic Vehicle Model

The vehicle is modeled as a rigid chassis containing *n* wheels with nonlinear independent suspensions, as described next.

2.2.1. Geometric Parameters of the Robotic Vehicle

The vehicle's orientation is given by three unit vectors $\boldsymbol{n}$, $\boldsymbol{t}$ and $\boldsymbol{b}$, corresponding respectively to the directions $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ of a local frame attached to it. The dimensions of the chassis are defined in these three directions, with the possibility of positioning the center of mass asymmetrically using two constants for each direction. Those constants are w_1 and w_2 for the width, l_1 and l_2 for the length, and h_{top} and h_{base} for the height.

2.2.2. External Forces on the Robotic Vehicle

The external forces (Figure 4) acting on the robotic vehicle can be summarized by the forces of interaction with the terrain and by the gravitational force $\boldsymbol{P}$. The interaction force with the terrain can be decomposed into three forces, $\boldsymbol{F_x}$ (representing the longitudinal force), $\boldsymbol{F_y}$ (representing the lateral force), and $\boldsymbol{N}$ (representing the normal force).

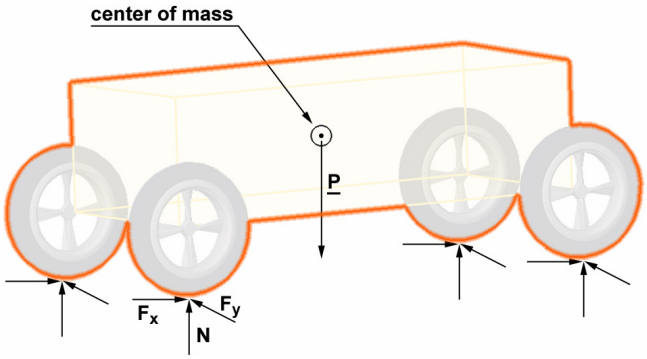

Figure 4. External forces acting on the robotic vehicle.

2.2.3. Internal Forces Between the Wheel and the Chassis

The interaction between each wheel and the chassis is made through the forces $\boldsymbol{F_n}$, $\boldsymbol{F_t}$ and $\boldsymbol{F_{susp}}$ (representing respectively the forces in the directions $\boldsymbol{n}$, $\boldsymbol{t}$ and $\boldsymbol{b}$) and the wheel torque τ. Figure 5 shows the forces acting on one of the wheels and the resulting reactions on the chassis. In the figure, $\boldsymbol{X_{cm}}$ represents the center of mass of the wheel.

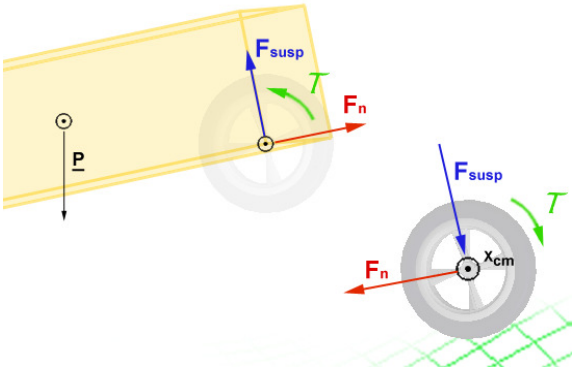

Figure 5. Internal forces between the wheel and chassis.

2.2.4. Search Algorithm for the Wheel-Terrain Contact Point

Although the contact between wheel and terrain is made through an area, it was assumed that the centroid of this area in accordance with a proper contact force model can represent the interaction between the tire and the soil. The effects of those simplifications are well discussed in Pacejka, 2006.

To find the wheel-terrain contact point, the displacement of the suspension imposed by the terrain (Figure 6) is estimated. The wheels are assumed massless in order to avoid high-frequency vibrations in the simulator, which would require smaller simulations time steps, increasing the computational cost. Thus, the suspension is either at rest (when no interaction with the terrain takes place) or under compression if otherwise. In this massless model, it is impossible for the suspension to work under traction, only compression can be observed. Despite ignoring the wheel mass in the

translational dynamic equations, the wheel moment of inertia is taken into consideration so that its angular dynamics can influence on the vehicle's behavior.

The displacement h of the suspension is assumed to be in the range from 0 (zero) to a saturation value H_{sat}. In this range, the force is modeled as linear. Beyond the saturation value, a nonlinear model is used. To estimate h, an iterative method is developed where the following equation must be satisfied:

$$Z_{cp} - f\left(X_{cp}, Y_{cp}\right) = 0 \tag{6}$$

where

$$Z_{cp}\left(h\right) = Z_{cp}\,' + h \cdot b_z \tag{7}$$

and X_{cp}, Y_{cp} and Z_{cp} are the coordinates of the contact point $\boldsymbol{X_{cp}}$; X_{cp}', Y_{cp}' and Z_{cp}' are the coordinates of the theoretical contact point $\boldsymbol{X_{cp}'}$ without any compression of the suspension; b_x, b_y and b_z are the coordinates of the unit vector $\boldsymbol{b}$; and $\boldsymbol{f}$ is an interpolation function representing the terrain profile.

The search algorithm first subdivides the bottom semi-circle of a cross section of the wheel into n equal parts. As seen in Fig. 6, each subdivision is located through an angle γ. With an iterative method similar to the bisection method, the algorithm examines each subdivision, calculating the displacement h that would be required to satisfy the Eq. (6). The angle γ for which the largest suspension displacement h is found is chosen as the contact point angle estimate.

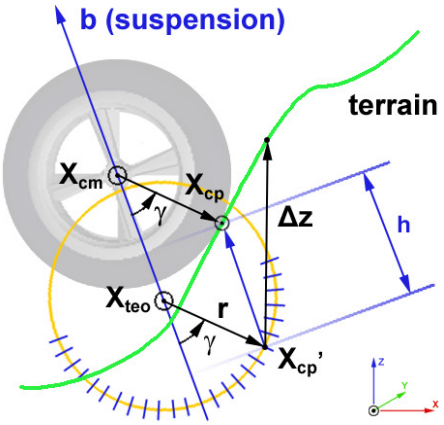

Figure 6. Scheme for the search algorithm of the contact point.

In Figure 6, $\boldsymbol{X_{teo}}$ is the theoretical position of the geometric center of the wheel without any deformation of the suspension; $\boldsymbol{X_{cm}}$ is the actual position of the geometric center of the wheel considering the displacement of the suspension; r is the wheel radius; $\boldsymbol{X_{cp}}$ is the actual wheel-terrain contact point; $\boldsymbol{X_{cp}'}$ is the theoretical wheel-terrain contact point based on the $\boldsymbol{X_{teo}}$ position; γ is the angle between the unit vector $\boldsymbol{b}$ and the normal force; and Δz is the positioning error that the algorithm tries to minimize, calculated by

$$\Delta z = Z_{cp}\left(h\right) - f\left(X_{cp} + h \cdot b_x, Y_{cp} + h \cdot b_y\right) \tag{8}$$

2.2.5. Dynamic Model of the Robotic Vehicle

For the force balance in each wheel (Figura 7), the magnitude of the impulse force $\boldsymbol{F_n}$ and the magnitude of the normal force $\boldsymbol{N}$ are obtained using $\boldsymbol{F_{susp}}$ and $\boldsymbol{F_x}$ through

$$F_n = \left(\frac{F_x - F_{susp}\sin\gamma}{\cos\gamma}\right) \tag{9}$$

$$N = \frac{F_{susp} - F_x \sin\gamma}{\cos\gamma} \tag{10}$$

where $\boldsymbol{F_x}$ is the traction force that the terrain exerts on the wheel and $\boldsymbol{F_{susp}}$ is the force that the suspension exerts on the wheel.

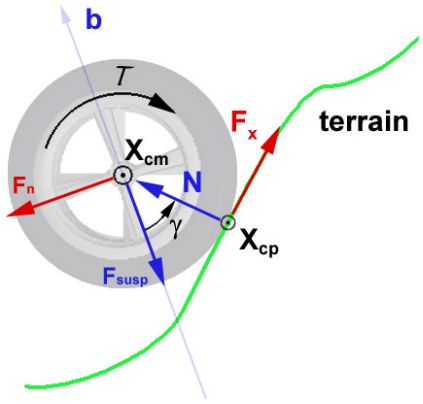

Figure 7. Forces and torques acting on a wheel.

The suspension force $\boldsymbol{F}_{susp}$ is calculated from its deformation and deformation rate. The direction of the normal force is given by the orientation of the wheels and the contact angle γ. The directions of the forces $\boldsymbol{F}_n$ and $\boldsymbol{F}_{susp}$ are also defined by the wheel orientation. The traction force $\boldsymbol{F}_x$ is always orthogonal to the normal force $\boldsymbol{N}$.

2.3. Wheel-Terrain Contact Force Model

First, a simplified model of the contact force between the tire and the terrain was implemented, proportional to the normal force ($\boldsymbol{F} = \mu\boldsymbol{N}$). However, this model, when used to represent the lateral force on the wheel, leads to an unstable and unrealistic simulator, causing oscillations even when the vehicle is not moving. This is due to the fact that such simplistic model can only assume three magnitudes $-\mu N$, 0 or μN, without enough equations to calculate the values in-between.

To solve such problem, the semi-empirical model called "Magic Formula" is implemented in this simulator (Pacejka, 2006). Pacejka (2006) states that this approach (fitted from experimental data) is more accurate than other existing methods. The model equations are

$$y = D\sin\left[C\arctan\left\{Bx - E\left(Bx - \arctan Bx\right)\right\}\right] \tag{11}$$

$$Y(X) = y(x) + S_V \tag{12}$$

$$x = X + S_H \tag{13}$$

where:

Y – output variable F_x (longitudinal force) or F_y (lateral force);
X – input variable $tan(\alpha)$ (lateral slip) or κ (longitudinal slip);
B – stiffness factor;
C – shape factor;
D – peak value;
E – curvature factor;
S_H – horizontal shift;
S_V – vertical shift;

2.4. Actuator

The actuators implemented in the simulator (Morrot, 2010) are brushed DC motors. To simulate the VIVI robot, the parameters from the motors Magmotor model S28-150 that it uses are used in the model. The simulator allows the configuration of any other brushed DC motor just by setting its parameters in the software script file.

Friction torque on the motor shaft is implemented using a modified version of LuGre's friction model, originally proposed by Canudas de Wit et al. (1995). The continuous approximation of the LuGre friction model (Sobczyk et al., 2009) eliminates the discontinuities of the original model, allowing its use for any desired speed value. The model equation is

$$\tau_a = \sigma_0 z + \sigma_1 \dot{z} + \sigma_2 \dot{y} \tag{14}$$

where σ_0 is the stiffness factor, σ_1 is the damping factor, σ_2 is the viscosity coefficients, z is a state variable, $\dot{z}$ is its variation through time, and $\dot{y}$ is the system speed (in this research, the angular speed of the motor shaft).

2.5. Numerical Integration Method

Given the unpredictability of the terrain profile, it is not possible to use implicit integration methods in the simulator such as Newmark-Beta's, since it would require the terrain profile to be represented by a continuously differentiable function. However, the terrain profile is in practice imported through height maps or elevation matrix files, obtained from surface scans. Smooth functions could be fitted to such discrete height maps to allow the use of an implicit integration method, however the computational cost would be too high due to the high non-linearity of the wheel-terrain interaction, and the need to use iterative methods to find the wheel contact point. This makes it impossible (or too costly) to predict the influence of the terrain on the vehicle in the next iteration, required in implicit methods. Instead, it is found that an explicit integration method with a very refined step can result in much faster computations with a similar accuracy. Therefore, the simulator currently uses the Euler integration method with a small integration step, typically in the order of only 100ns.

3. CONTROL

Stability control for robotic vehicles in rough terrain has been developed and implemented both in simulations and experiments (Iagnemma and Dubowsky, 2004; Silva, 2007; Santos, 2007). The developed algorithms aim at stability through controlling the traction force at each wheel using the dynamic equations of the system to predict its behavior, while compensating for any destabilizing tendencies.

This study implemented the stability control algorithm CDTA proposed by Silva (2010) for rough terrain. This algorithm aims to keep the wheel traction force lower than the limit from the friction cone, while trying to keep the normal forces on each wheel always positive (to keep the wheel in contact with the terrain). The CDTA also avoids saturating the motors, and minimizes the power dissipated by them. The traction force of wheel i ($\boldsymbol{F_{xi}}$) must meet the following criteria: 1) $\left|F_{x_i}\right| < T_{sat} r_i$; 2) $\left|F_{x_i}\right| < F_{n_i}$; and 3) $\left|F_{x_i}\right| < F_{t\max_i}$, where T_{sat} is the saturation torque of the motor; r_i is the radius of wheel i; and $\boldsymbol{F_{ni}}$ is the maximum value of the traction force that allows all normal forces to remain positive. The maximum friction force used by the author is μN, where μ is the coefficient of friction and N is the magnitude of the normal force.

To improve the model for the wheel-terrain interaction, it is necessary to use, instead of μN, the coefficient D from Eq. (11) to represent the maximum traction force, from the "Magic Formula" model. The juxtaposition of these constraints forms an admissible range of traction forces for the wheels among which the solution to the stability control can be sought.

4. SOFTWARE VALIDATION

To ensure that the results of this study are reliable, a validation of the software is needed. For the sake of simplicity during the tests, the robotic vehicle has its center of mass $\boldsymbol{X_c}$ (represented in a global inertial system) coincident with the geometric center, although the software allows its location at any point.

Two cases where analytical solutions exist are tested, so that the software results can be compared. For both cases, no control algorithm is used so that constant accelerations can be applied to the chassis through constant torques to the wheels. The first case (Figure 8) consists of a forward acceleration through constant torques over a horizontal plane, where the normal forces at each wheel and the pitch angle are monitored. In the second case (Figure 9), an initial condition for the vehicle speed is set so that it loses ground contact from all wheels while jumping over a bump in the terrain, while the landing distances are predicted. An integration time step of 0.65μs is used for both cases. Data from the vehicle produced by the simulator during the tests, as well as all the results of the analytical solutions, can be seen in Morrot (2010). For both cases, four different settings of thrust and initial conditions (respectively) are tested so that the software is better evaluated.

In Figure 8, h_D is the displacement of the front suspension, h_T is the displacement of the back suspension, $\Delta h = h_D - h_T$, L is the distance between the front and back axis, L_1 and L_2 are respectively the distances from the back and front axis to the center of mass of the robotic vehicle, and θ is the pitch angle of the chassis.

Four trials are made for the first case, where in two of them the refinement of the geometrical representation of the wheels is explored. It is found that a greater level of detail (from 20 to 2000 slices to represent the lower half of the wheel's circunference) resulted in a decrease of the error of the front wheel normal forces from 0.34% to 0.03%. However, this improvement leads to an increase in processing time of 467%, suggesting that this improvement is not significant enough compared to the computational cost it represents. The other tests (evaluated with the original level of refinement) show that the error on the front wheel normal force does not exceed 1.23% with respect to the analytical solution. For the pitch angle comparisons, errors less than 0.01° were found. Those results show that the wheel-terrain interaction is well estimated since the error can be greatly decreased if a faster computer can be used.

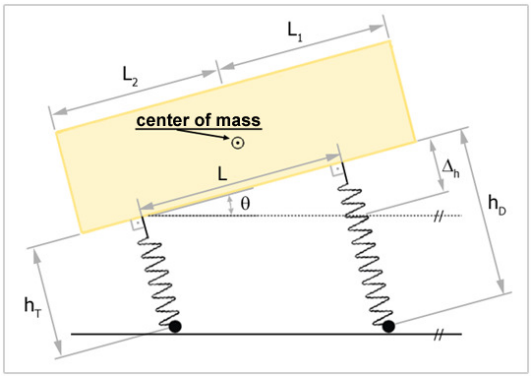

Figure 8. Parameters used in the calculation of the analytical solution for the acceleration test.

In Figure 9, V_y is the vertical component of the vehicle speed exiting a virtual ramp, V_x is its horizontal component, V is the speed of the vehicle and θ represents the angle of the virtual ramp. The angles used are 22.5º, 35º, and 45º. For this case of the simulated jump, four simulations are performed, where the maximum error of 0.02% is found over analytical results. Those results show that the implemented integration method is able to reproduce the classic scenario of a cannon shot.

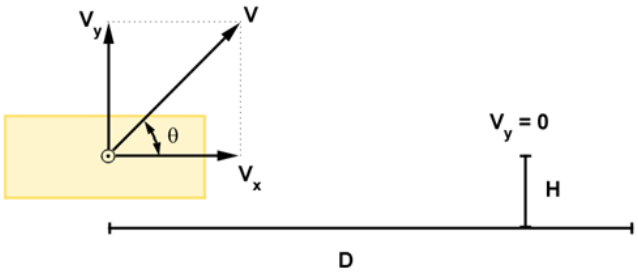

Figure 9. Scheme of the jump test.

5. SIMULATION RESULTS OF THE IMPLEMENTED STABILITY CONTROLS

The CDTA control and a proportional control with gravity compensation are used to control the vehicle speed. Their performance is compared in a Sinusoidal Terrain (Figure 10.a) and in a flat horizontal terrain with a ramp (Figure 10.b). The vehicle specifications used can be found in (Morrot, 2010).

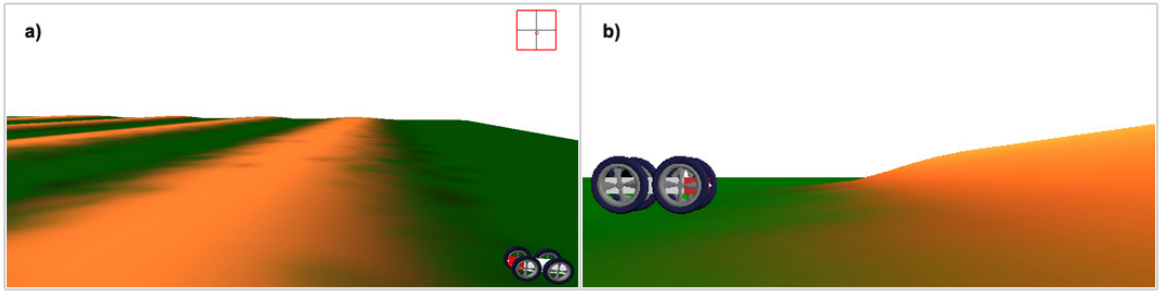

Figure 10. Simulations scenarios in a (a) sinusoidal terrain; (b) flat horizontal terrain with a ramp at the end.

The desired speed for the simulations is 10m/s, and a proportional gain of 100Hz is used on both controllers. The simulation uses an integration step of 65μs and the controllers are simulated at a frequency of 10kHz. A model for a 36V battery capable of delivering up to 80A is implemented to simulate the power limitations of the motors. In these simulations, the same torque is sent to both rear wheels, and another torque to both front wheels.

In Figure 11, one can see that by using the CDTA control the vehicle takes the advantage of higher traction forces to achieve the desired speed faster than the proportional control, while maintaining it throughout most of the simulation. Figure 11 also shows that the simulation prematurely ends for the proportional control after 8s. It happens due to a roll over after hitting the ramp, indicating the effectiveness of the CDTA approach to prevent roll-over.

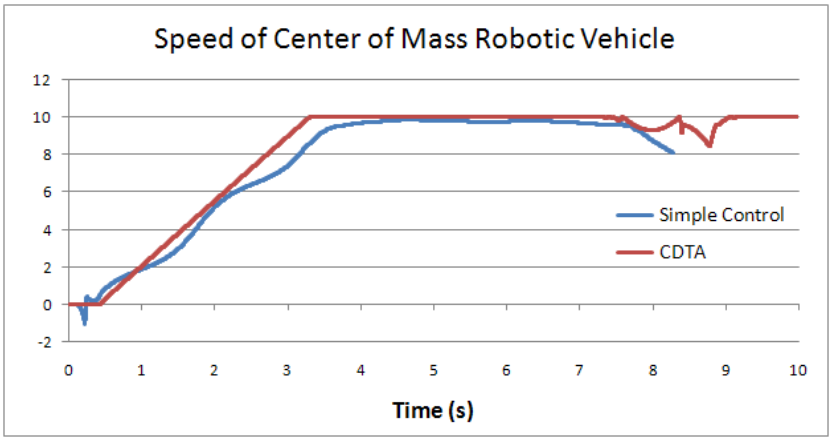

Figure 11. Robotic vehicle speed (in its center of mass) with a proportional controller with gravity compensation and the CDTA control, in the horizontal flat terrain with a ramp.

The limited torque allowed by the CDTA control (so that the maximum traction force is not exceeded) helps the vehicle to waste less energy than the proportional control, since the power consumed by the system (showed in Figure 12) is lower. This happens because, with the traction force limited, the longitudinal slip is minimized, not wasting energy by excessively spinning the wheels. Those behaviors are in agreement with the ones described by Silva et al. (2010), showing ability of the VirtualBotz 3D Simulator to predict its behavior.

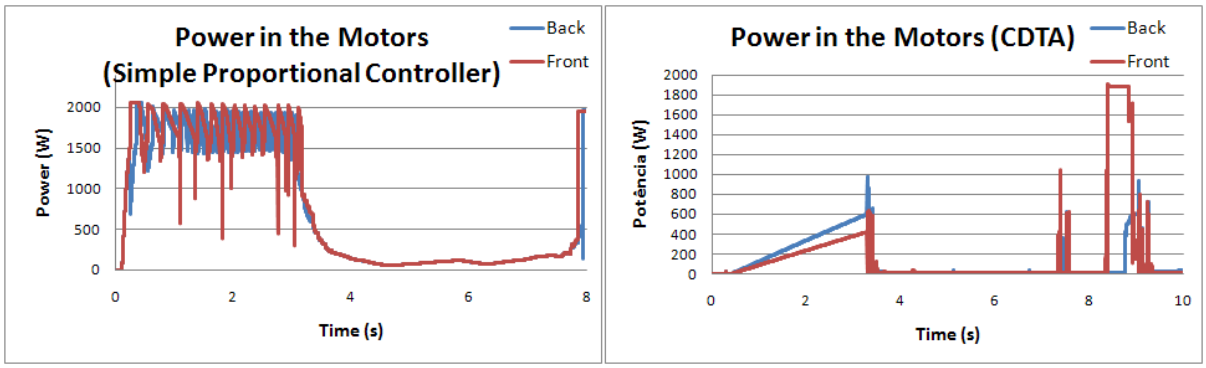

Figure 12. Power dissipated by the motors in the horizontal flat terrain with a ramp.

6. CONCLUSIONS

In this work, a simulation program was developed in C++ to predict the behavior of mobile robots in rough terrain, even at high speeds. The simulator allows the import and definition of different types of terrain profiles and robotic vehicles, as well as the definition of the DC drive motor characteristics. Finite power batteries were implemented in the model, so that the control algorithms were not allowed to generate unrealistic actuations during the simulations. A continuous approximation of LuGre's friction model was implemented in the motor shafts to simulate energy dissipation. An intersection algorithm to find the wheel-terrain contact point was developed, avoiding the need to use commercial third-party libraries for collision detection. The contact force between the tire and the terrain was modeled using the "Magic Formula" that considers both lateral and longitudinal slips combined and the normal force on the wheels. To validate the results of the simulator, comparisons with analytical solutions of an acceleration scenario and a jump scenario were made. The first one showed that a larger level of refinement of the wheel model (used in the wheel-terrain contact point search algorithm) causes an improvement in the accuracy of the predicted forces on the front wheel from 0.34% to 0.03% inducing, however, an increase of the processing time of 467%. Those results prove the effectiveness of the developed simulator. The Dynamic Control for Rough Terrain (CDTA) proposed by Silva (2010) was also implemented in the simulator and compared to a proportional velocity control with gravity compensation. The comparison between their performances showed an agreement with Silva's results, confirming the ability of the developed simulator to reproduce its behavior.

7. REFERENCES

Canudas de Wit, C., Olsson, H., Astrom, K.J. and Lischinsky, P., 1995, "A New Model for Control Systems with Friction", IEEE Transactions on Automatic Control, Vol. 40, n. 3, pp.419-425.

Chee Shin Yeo, Buyya, R., Pourreza, H., Eskicioglu, R., Graham, P., and Sommers, F., 2006, "Cluster Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers", Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies, chapter 16, Pages: 1-24.

Iagnemma, K. and Dubowsky,S., 2004, "Mobile Robots in Rough Terrain: Estimation, Motion Planning, and Control with Application to Planetary Rovers". Series: Springer Tracts in Advanced Robotics, August 17 2004, Vol. 12, 1st ed., Springer.

Iagnemma, K. and Dubowsky,S., 2004, "Traction Control of Wheeled Robotic Vehicles in Rough Terrain with Application to Planetary Rovers". The International Journal of Robotics Research, October 2004, vol. 23, no. 10-11 1029-1040 .

Morrot, R., 2010, "Simulação Tridimensional em Tempo-Real de Veículos Robóticos em Terrenos Acidentados",M.Sc. Thesis, Mech. Eng. Dept., Pontifical Catholic University of Rio de Janeiro, Brazil (In Portuguese).

Pacejka, H.B., 2006, "Tire and Vehicle Dynamics", SAE – Society of Automotive Engineers, Inc., Second Edition, pp. 156-215.

Peters, S.C. and Iagnemma, K., 2009, "Stability measurement of high-speed vehicles.", Vehicle System Dynamics, Vol. 47 Issue 6, pp.701-720.

Santos, A.V., 2007, "Controle de Capotagem e Deslizamento de Sistemas Robóticos Móveis em Terrenos Acidentados", M.Sc. Thesis, Mech. Eng. Dept., Pontifical Catholic University of Rio de Janeiro, Brazil (In Portuguese).

Silva, A.F.B., 2007, "Modelagem de Sistemas Robóticos Móveis para Controle de Tração em Terrenos Acidentados", M.Sc. Thesis, Mech. Eng. Dept., Pontifical Catholic University of Rio de Janeiro, Brazil (In Portuguese).

Silva, A.F.B., Santos, A.V., Meggiolaro, M.A. and Neto, M.S., 2010, "A Rough Terrain Traction Control Technique For All-Wheel-Drive Mobile Robots", Journal of the Brazilian Society of Mechanical Sciences and Engineering, 2010, ABCM.

SobcZyk, S., Mario, R., Perondi, E.A. and Cunha, M.A.B., 2009, "A Continuous Approximation Of The LuGre Friction Model", 20th International Congress Of Mechanical Engineering – COBEM 2009, November 15-20, Gramado, RS, Brazil.

8. RESPONSIBILITY NOTICE

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