


Dynamic simulation of legged robots using a physics engine

Conference Paper**Author(s):**

Belter, Dominik; Skrzypczyński, Piotr; Walas, Krzysztof; Fankhauser, Péter; Gehring, Christian; [Hutter, Marco](#) ; Hoepflinger, Mark A.; Siegart, Roland

Publication date:

2014

Permanent link:

<https://doi.org/10.3929/ethz-a-010173667>

Rights / license:

[In Copyright - Non-Commercial Use Permitted](#)

Originally published in:

https://doi.org/10.1142/9789814623353_0066

DYNAMIC SIMULATION OF LEGGED ROBOTS USING A PHYSICS ENGINE

D. Belter, P. Skrzypczyński, K. Walas

*Institute of Control and Information Engineering, Poznan University of Technology,
Poznan, 60-965, Poland*

*E-mail: dominik.belter@put.poznan.pl
www.put.poznan.pl*

P. Fankhauser, C. Gehring, M. Hutter, M. A. Hoepflinger, R. Siegwart

*Autonomous Systems Lab, ETH Zurich
Zurich, 8092, Switzerland*

E-mail: pfankhauser@ethz.ch

This article presents an application for dynamic simulation of legged robots based on a physics engine. In the presented application an iterative solver is supported by analytical equations of the dynamics and software modules for collision detection, environment modeling and visualization. The presented application of the simulator allows for development and verification of control algorithms before their implementation on the real robot.

Keywords: Legged robot; dynamic simulation; motion control

1. Introduction

Legged robots are complex mechatronics systems and the development of motion control methods on a real walking robot requires significant effort. The behavior of a legged robot is difficult to control because predicting the interaction with the environment is notoriously challenging. Inaccurate models of the ground might cause the robot to slip and fall, which can damage the mechanics and electronics of the systems. The necessary maintenance effort is often very high and introduces significant delays in the development process.

In another aspect, a well functioning control system requires reliable information feedback from a multitude of sensors. The higher the quality of the acquired map of the environment is, the better the control strategy can perform. Additionally, the robot is required to localize itself in the

environment in order to generate suitable motion. Both, localization and mapping, are complex and still open research problems. We can avoid these difficulties on the real robot by using an external motion capture system which, however, only works in limited areas. By using a simulator of the dynamics, one can focus on the development of control methods and mitigate the problems of self-localization and mapping⁶. Moreover, it is possible to develop mapping algorithms with the simulator since the ground truth data is easily available for the simulated system.

The advantages of working with a simulation of the dynamics are multifold. Besides protecting the real robot from damages during experiments and providing ground truth data whenever required, a sufficiently fast simulator enables to run multiple simulation experiments in a short amount of time to verify research hypotheses before the implementation on the real robot. This can tremendously accelerate the research and development phase of the robot control system.

The basis for simulating the dynamics of the system are mathematical models, which for legged robots are generally nonlinear and discontinuous due to the kinematic setup, intermittent contacts with the ground, and gear backlash. These equations of motion cannot be solved analytically and therefore numerical methods are applied. The numerical integration methods used by physics engines are based on the approximation of the area under the derivative curves. Thus, the dynamic behavior of a robot obtained from a physics engine is always a simplified model.

Physics engines originally found their main application as game engines, animation and modeling software. They are able to simulate rigid and soft body dynamics, particles and detect collisions between objects⁴. As the physics engines have become more sophisticated they have also found their application in science and engineering. In our work, we use an open source physics engine to obtain a simulation of the dynamics of a legged robot in interaction with its environment. We support the simulation by providing the output of the controllers and behavior of passive elements.

This paper introduces our simulator for legged robots based on a physics engine and we present the techniques which allow us to obtain stable and reliable simulations.

1.1. *State of the Art*

Many commercial simulators that allow to model legged robots are available: Webots, Gazebo, Robotics Studio, USARSim⁹. Some of the most popular physics engines are the Open Dynamics Engine (ODE), PhysX,

MATLAB SimMechanics Toolbox and Bullet^{3,4}. Each of these solutions has specific advantages and disadvantages for certain applications. As we design our robots with specific mechanical and electronic structure, we have designed our own simulator to ensure better control on the implementation details.

We base our simulator on ODE¹⁰, as it performs well considering the stability of multi-body robotics systems³. ODE can be used to approximate the physical behavior of a robot at high fidelity⁶, as it can simulate multi-body dynamics and contacts with a soft contact model. Furthermore, a collision detection tool and a visualization are provided within the software. There is no Graphical User Interface (GUI) from ODE, which increases its efficiency.

The ODE environment is often used to simulate legged robots (e.g. Sony AIBO⁷). The representation of a robot in simulation often consists of boxes, spheres, and capsules which are connected by rotational joints. Modeling friction in the joints increases the accuracy of the simulation. To model a servomotor in a joint, P and PID controller implementations are provided, which makes the behavior of the simulated robot more realistic. The physics simulation is separated from the visualization. In spite of the simplified model used for physics modeling, an independent, detailed model can be used for the visualization.

Some issues are specific to ODE's simulation approach. It is known for performing poorly when simulating certain types of contacts and friction. Precise tuning of the parameters of the environment, like friction and ground softness, are necessary. Better accuracy can be obtained by decreasing the simulation time-step, which on the other hand decreases the simulation speed. It was already shown by Tanev et al.¹¹ that appropriate tuning of the model parameters in ODE allows to simulate a system whose movement is friction-based, such as that of a snake-like robot.

2. The Simulator

The simulator is characterized by the modular design as depicted in Fig. 1. The physics simulation (Simulated World) is separated from the control system of the robot (Robot Controller). This setup allows to send the output from the robot controller to the simulated or/and to the real robot without modifications. The robot controller contains all modules which are required to control the real robot. This includes the mapping algorithm, motion planning, low-level gait control and modules which support the control and motion planning such as stability and collision checkers.

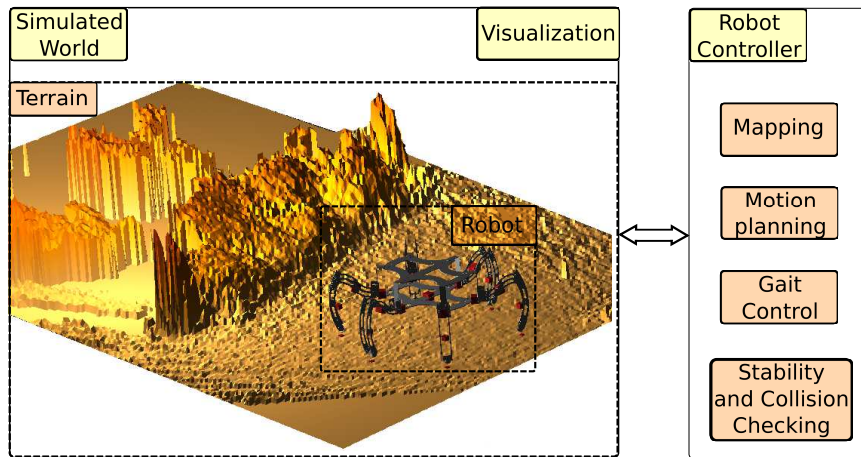


Fig. 1. Software modules of the simulator

The visualization of the robot model and the dynamic world is implemented with OpenGL routines. Appropriate visualization eases the observation of the robot's behavior. The robots as well as the ground are drawn with triangle mesh models (cf. Fig. 1 and Fig. 3A and B).

An advantage of the simulator is the possibility to know perfectly and instantaneously the full state of the robot during the experiments. The same ground-truth is difficult to obtain on the real robot in real environment. A custom library records and visualizes the state of the robot (position and orientation of the platform, leg trajectories, planned path, etc.). The recorded data can also be plotted during execution of the simulation and written to a file for further analysis.

2.1. *Environment Model*

The environment model is created using an elevation (grid) map. We use maps which were created during experiments on the real robot or we provide noise-free maps with artificial obstacles such as steps, bumps, and stairs. The elevation map is converted to a triangle mesh model and used by visualization, dynamics engine and collision detection module. The robot controller uses the original elevation map to plan the motion of the robot.

2.2. Robot Model

The geometry of the simulated robot is created by using simple box primitives, which keeps the computation cost low. Simplification of the robot model makes the simulation faster by reducing time consumed for collision checking. Despite the reduced complexity of the model, the simulation results are still reliable.

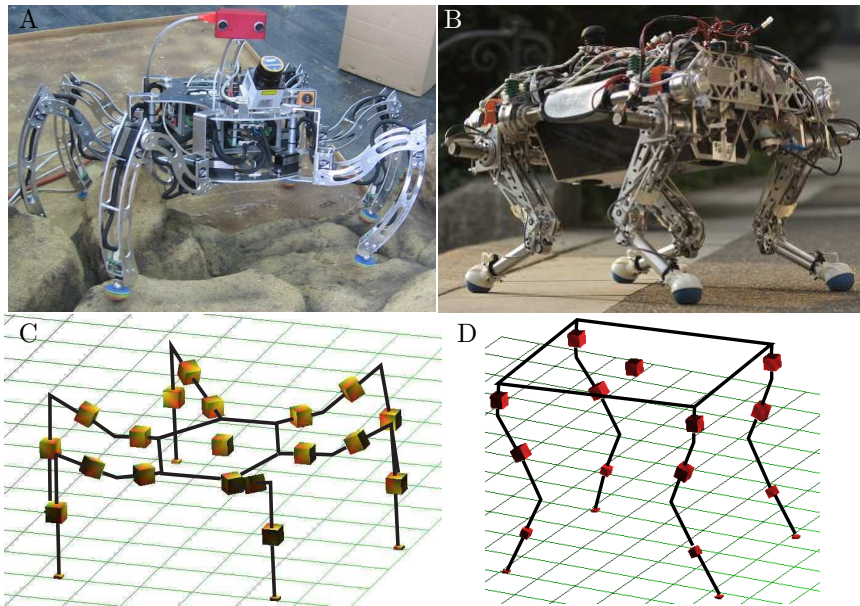


Fig. 2. Messor and StarLETH robot models

The ODE model for two legged robots, namely the robots Messor¹² and StarLETH⁵, are presented in Fig. 2. Each link of the robots is represented by a cube with proper mass and inertia and the links are connected by rotary joints. Each actuator is modeled as a proportional controller, with the output u defined as

$$u = k \cdot (\phi_{\text{ref}} - \phi_{\text{curr}}), \quad (1)$$

where ϕ_{ref} is the desired joint position, ϕ_{curr} is the current joint position, and k is the controller gain. The controller gain values are determined experimentally in such a way that a speed similar to that of the real robot

motor is achieved. The maximal torque value is limited according to the real system.

We use the mathematical description of the controller instead of dedicated servomotors model included in ODE because it allows for higher flexibility. The simple control rule (1) is sufficient for the Messor robot which is build using simple servomotors. StarLETH as well as the new Messor II have more complex drives and we are going to update the controllers to better approximate the real actuators.

2.3. Collision Detection Module

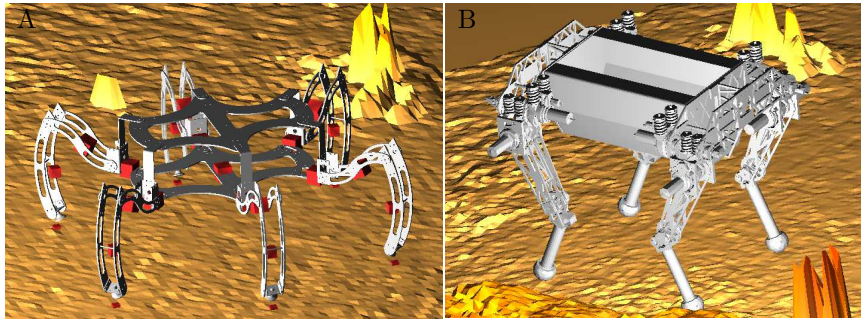


Fig. 3. CAD models of the robots used for visualization and collision detection

The simulator is equipped with two collision checking modules. The first one is the embedded ODE collision detector and the second is a separate collision checker used by the robot controller. The collision detection module used by the robot controller uses a CAD model of the robot's frame, which is shown in Fig. 3. The same CAD model is used for the visualization of the robots.

The CAD model is converted to a triangle mesh. When two bodies with triangle mesh collide, there exists a line which is common for two triangles. To detect a collision, the triangle-triangle intersection test from the Oriented Bounding Box (OBB) algorithm is used⁸. It is fast and reliable when used to detect collisions between parts of the robot (the model is fixed, only its configuration may vary). However, collision detection between the robot and the ground is significantly slower, because the ground model changes as the robot walks over the terrain. Thus, the collision detection module is used only to prevent self-collisions of the robot.

3. Applications of the Simulator

3.1. *Evolutionary Gait Optimization*

We applied the simulator to a method which allows to evolutionary optimize the gait of a six-legged robot¹. The simulator is used to test various solutions (trajectories of the robot) found by the algorithm. Each optimization trial consists of thousands of simulations. In this case the simulation speed plays an important role, as with the default configuration a single optimization would have lasted for several weeks.

To reduce the simulation time we increase the integration step. However, increasing the integration time step decreases the accuracy of the simulation and can even render the simulation unstable. To solve the stability problem and preserve accuracy of the simulation, we first optimize for the parameters of the simulation, and moreover, for the physical parameters of the robot¹. As a result, the parameters of the simulated world and robot model differ from the real parameters. However, the simulation is fast and still sufficiently accurate such that the results obtained in the simulator can be applied to the real robot.

3.2. *Motion Planning and Navigation*

The simulator is also used to develop motion planning and navigation methods for walking robots². In this case, the reliability of the simulation is of high importance. We obtain sufficient accuracy for an integration step of 0.1 ms. This decreases the simulation speed, but for this application the simulator needs to be run only once to execute the planned path.

One of the modules used for motion planning is a foothold selection module. We use the simulator to learn the behavior of the robot during contact with terrain templates². Such experiments are difficult to perform and time-consuming on the real robot. To increase the interaction between the foot and the terrain template, we decrease the friction coefficient of the ground. As a result, the robot learns a conservative behavior which acts as a safety margin on terrains with higher friction.

4. Conclusions

This paper presents a simulator of legged robots, which allows to model the behavior of robots with different number of legs and different morphologies. We demonstrated our approach for a six and a four-legged robot and for different applications. We presented various techniques which enable effi-

cient modeling of the system dynamics. We use our simulator to develop new control methods dedicated to legged robots.

Currently, we are improving the models of the StarLETH and Messor II robots in the simulator and continue to develop and verify different control strategies. Our long-term goal is to develop motion planning algorithms which take the dynamic properties of the robot into account.

References

1. Belter D, Skrzypczyński P, A Biologically inspired approach to feasible gait learning for a hexapod robot, *International Journal of Applied Mathematics and Computer Science*, Vol. 20(1), pp. 69-84, 2010
2. Belter D, Skrzypczyński P, Posture Optimization Strategy for a Statically Stable Robot Traversing Rough Terrain, *IEEE/RSJ 2012 International Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, pp. 2204–2209, 2012
3. Boeing A, Braünl T, Evaluation of real-time physics simulation systems. In *Proc. of 5th Int. Conf. on Computer Graphics and Interactive Techniques*, pp. 281–288, 2007
4. Hummel J, Wolff R, Stein T, Gerndt A, Kuhlen T, An Evaluation of Open Source Physics Engines for Use in Virtual Reality Assembly Simulations, *Advances in Visual Computing, Lecture Notes in Computer Science*, Vol. 7432, pp. 346-357, 2012
5. M. Hutter, C. Gehring, M. Bloesch, M. A. Hoepflinger, C. D. Remy, R. Siegwart, StarLETH: A compliant quadrupedal robot for fast, efficient, and versatile locomotion, *Adaptive Mobile Robotics* (N. Cowan et al., eds.), Singapore, World Scientific, pp. 483–490, 2012,
6. Kutter O, Hilker C, Simon A, Mertsching B, Modeling and Simulating Mobile Robot Environments, *Proc. 3rd Int. Conf. on Computer Graphics Theory and Applications*, Funchal, Portugal, pp. 335–341, 2008
7. Laue T, Spiess K, Röfer T, Simrobot - a general physical robot simulator and its application in RoboCup, *RoboCup 2005: Robot Soccer World Cup IX. Lecture Notes in Artificial Intelligence*, pp. 173–183, Springer
8. Möller T, A Fast Triangle-Triangle Intersection Test, *Journal of Graphics Tools*, vol. 2, pp. 25–30, 1997
9. E. Van Noort, A. Visser, Extending virtual robots towards robocup soccer simulation and @home, In: *Proceedings of the 16th RoboCup Symposium. Lecture Notes on Artificial Intelligence*, Springer, pp. 332–343, 2013
10. Smith R, Open Dynamics Engine, <http://www.ode.org>, January, 2014
11. Tanev I, Ray T, Buller A, Automated Evolutionary Design, Robustness, and Adaptation of Sidewinding Locomotion of a Simulated Snake-Like Robot, *IEEE Transactions on Robotics*, vol. 21(4), pp. 632–645, 2005
12. Walas K, Belter D, Messor – Versatile walking robot for search and rescue missions, *Journal of Automation, Mobile Robotics & Intelligent Systems*, vol. 5(2), pp. 28–34, 2011