Dynamic simulator for humanoids using constraint-based method with static friction

J-R. Chardonnet — *LIRMM - JRL CNRS*
S. Miossec — *JRL - AIST*
A. Kheddar — *JRL - CNRS*
H. Arisumi — *JRL - AIST*
H. Hirukawa — *HRG - AIST*
F. Pierrot — *LIRMM*
K. Yokoi — *JRL - AIST*

Abstract— **A dynamic simulator using constraint-based method is proposed. It is the extension of the formalism previously introduced by Ruspini and Khatib by including static and dynamic friction without friction cone discretization. The main contribution of the paper is in efficiently combining the operational space formulation of the multi-body dynamics in the contact space and solving for contact forces, including friction, using an iterative Gauss-Seidel approach. Comparing to existing work in this domain, we illustrate our method with scenarios involving humanoid in manipulation tasks while contacting with the environment; an experiment validates our results. Technical details that allow an efficient implementation and problems with future orientation to improve the simulator are also discussed. This work is aiming to be a potential module of the next OpenHRP simulator generation.**

I. Introduction

Many humanoid robots that we can see nowadays like Toyota's one and Honda's robot Asimo have been presented mainly for entertainment. However, the main aim of a humanoid is to work and provide services in all kind of environments. People may request inevitably robots to perform some manipulating tasks, some of which can be hazardous for humans. Robots could thus be very helpful (Fig. 1).

Fig. 1. A collaborative task involving manipulation of various objects by humanoids.

But robots can not perform such tasks if they do not acquire some knowledge on the environment. Haptics is one of the modality which is important to perform physical tasks, which induces physical interaction with the environments and objects, hence involving to treat contacts. These kind of capabilities will be more and more developed in the next few years as new theories and models will be presented. Nevertheless, before testing on real platforms, simulation is needed to validate such models.

In particular, simulating the dynamics of a complex system such as humanoid robots is not simple as many unknown parameters intervene in the computation such as external forces and accelerations. Without any contact forces, accelerations can be computed quite easily if we consider as known some external parameters like torques, that can be measurable. If we include contacts, the computation of dynamics becomes more complicated as we do not know a priori reaction forces at contact points, which results in a non-linear form between contact forces and accelerations. This issue can be resolved by using a penalty method, which is widely used in actual simulators like OpenHRP[1] and many others. Although simple and fast, the penalty parameters of the method are difficult to tune which compromises robustness and even stability of the simulation and generally the way to determine them is not explicitly mentioned; static friction is also difficult to implement properly. Recently, Yamane and Nakamura [1] proposed to compute dynamic friction by weighting it, which allows to solve discrete-time integration.

Recently, other methods have been proposed, among them constraint-based methods, in which non-penetration constraints are explicitly integrated to the dynamic equation and hence accuracy highly increases. They become more and more used in simulators, even in video games [2]. Baraff [3] introduced constraint-based methods for contact force computation in rigid bodies simulation by formulating the problem as an LCP (Linear Complementarity Problem). As dynamics give accelerations, he expressed the problem in terms of acceleration, but there might be cases with no solution for friction forces. He applied impulse forces to allow discontinuity of velocities. Formulation of the dynamics as an LCP appears to be elegant for solving contact problems since a solution is guaranteed using pivoting methods like Lemke's algorithm. This formulation is kept even if friction is introduced, provided that friction cones need to be discretized into facets inducing additional constraints for each cone's facet. The price to pay for this benefit is the necessity of discretizing into many facets for each friction cone to obtain a decent accuracy. This arouses robustness and computation time matters, especially in matrices' computation as the size of the matrices relies on the desired accuracy.

Anitescu [4] and Stewart [5] proposed a method for solving friction using an LCP formulation. Both presented an implicit

[1] http://www.is.aist.go.jp/humanoid/openhrp/

method for solving contact forces with friction, through the computation of estimated joint positions, which allows no to find explicitly the times of impact. However, this benefit reaches rapidly its limit as it requires small time steps to solve the problem in an acceptable way. Implementing these proposed algorithms in our context is not trivial. Robustness of the proposed approaches in face of high number of contacts and bilateral constraints can not be proved and the given examples are always simple case studies. Miller and Christensen [6] devised a grasping simulator based on Anitescu's approach. They integrated additional constraints for joint limits and joints' constraints reinforcement. The examples show grasped objects with different grippers without arm motion, the complexity of the algorithm is not given in terms of dof and the number of contacts.

Ruspini and Khatib [7] introduced a formalism for solving contact and impact using LCP implemented as an impressive simulator. The only missing feature is that they did not include friction. However, the main difference vis-a-vis other methods is that they use the operational space formulation of multi-body dynamics in the contact space, which allows a more efficient dynamics computation. We are using a similar approach as the basis of the simulator we propose.

Recently, an iterative Gauss-Seidel approach has been applied by Liu and Wang [8] to robotics. The interesting point with this method is that it allows to keep exact friction cones with a guaranteed convergence and a size of the system equal to the number of contacts thus reducing considerably computation time. Liu and Wang modeled surfaces as explicit equations so that the gradient is computed. Again, the illustrative examples do not demonstrate how the method performs in complex scenarios. In the other hand, Duriez et al. [9] also used an iterative Gauss-Seidel method to compute real-time force feedback in deformable objects. They showed that: the iterative Gauss-Seidel approach becomes much more faster than an LCP formulation for more than 10 contact points, and it is more precise since a minimum of 8 facets are required to have 10% accuracy when compared with an LCP.

Many other works dealing with contact problems illustrate examples with very simple scenarios and furthermore do not prove their validity by experiments. We think that such implementations hide actual problems of robustness, complexity, and stability that may recall into question hypotheses and proposed algorithms. In the ideal case, actual purpose implementations may require to consider additional aspects of the problem. We aim at realizing an interactive simulator for complex systems like humanoid which computes in a realistic manner, contact forces with friction while handling all related aspects of collision detection, numerical integration, etc. Our work is based on [7] because of its simplicity. The central contribution of this paper is in efficiently combining operational space formulation of dynamics, extending the model of contact to friction and solving it using a Gauss-Seidel approach, as in [9], illustrating complex scenarios with contact configurations using a 30-dof humanoid, and showing the effectiveness of our proposed simulator by experiments. In this work, we do not take into account joint friction.

II. Main algorithm

A. Dynamic model without contact

When torques are applied to a robot's joints, it moves according to the following differential equation:

$$\ddot{q} = A^{-1}(q)\left[\Gamma - b\left(q,\dot{q}\right) - g(q)\right] \tag{1}$$

where A is the inertia matrix of whole robot, Γ is the vector of joint torques, b is the centrifugal and Coriolis effects and g the gravity. There are several formulations for dynamics such as Newton-Euler, Lagrange, etc. Here we have chosen Newton-Euler formulation as it is more suited for programing. Many algorithms have been proposed, among them the Featherstone's algorithm [10], [11] which proposes to compute the forward dynamics model within three recursions:

- computation of the geometric and kinematic parameters,
- computation of the inertias in the base frame and external forces (articulated-body inertias and bias forces which are the forces that give a null acceleration),
- computation of the joint accelerations.

B. Equations for solving impact forces

Impact forces are solved in terms of velocities as the impact model assumes that velocity of collision points just after collision time should be zero to avoid penetration. Considering two bodies colliding in m points with corresponding relative velocities, Ruspini and Khatib showed that the impact force to be applied must satisfy the following equation:

$$0 \leq f_I \perp \Lambda_I^{-1} f_I + B_I \geq 0 \tag{2}$$

where $\Lambda_I^{-1} = J_I A^{-1} J_I^T$ and $B_I = (1+\epsilon) J_I \dot{q}^-$. J_I represents the Jacobian from the joint space to the contact space for impact. Joints' velocities are updated by:

$$\dot{q}^+ = A^{-1} J_I^T f_I + \dot{q}^- \tag{3}$$

Equation (2) can be solved using Lemke's algorithm, from where we get impact forces f_I.

C. Equations for solving contact forces

From Ruspini and Khatib's approach, new constraints appear for solving contact problems. These constraints are written in terms of acceleration: relative normal accelerations should be zero at contact points and positive normal force f (reaction force) should be applied:

$$0 \leq f \perp \Lambda_c^{-1} f + B_c \geq 0 \tag{4}$$

where $\Lambda_c^{-1} = J_c A^{-1} J_c^T$ and $B_c = J_c A^{-1}\left[\Gamma_{joint} - b - g\right] + \dot{J}_c \dot{q}$. Γ_{joint} represents the applied joint torques and J_c is the same as J_I for contact. This equation is written into an LCP form and thus can also be solved with Lemke's algorithm.

From now, we extend Ruspini and Khatib's approach by including friction. We will denote f as the contact force considering friction. We consider Coulomb's law as the friction model. Since friction introduces a non-linear condition ($||f_t|| \leq \mu f_n$), it can not be integrated as an additional constraint to the existing LCP (as for impact forces) unless discretization of friction cones is made [4], [5].

Moreover, equation (4) has been written in terms of accelerations. To avoid any problems described in [3], we formulated the problem in terms of velocity (which is the most common way) by integrating equation (4) with a simple Euler integration, that is:

$$a_c = \Lambda_c^{-1} f + B_c \tag{5}$$

we get

$$v_c^{k+1} = dt\Lambda_c^{-1} f + \left(dtB_c + v_c^k\right) \tag{6}$$

with dt the time step and k the step. As our goal is to provide exact contact forces, we solve this problem with an iterative Gauss-Seidel approach, combined with a Newton-Coulomb method [12].

Now, considering the equation of motion of the robot (1), we can add contact forces with $\Gamma = \Gamma_{joint} + J_c^T f$, so that we obtain:

$$\ddot{q} = \ddot{q}_{f=0} + \ddot{q}_{f\neq 0} \tag{7}$$

where

$$\ddot{q}_{f=0} = A^{-1}\left[\Gamma_{joint} - b - g\right] \text{ and } \ddot{q}_{f\neq 0} = A^{-1} J_c^T f \tag{8}$$

$\ddot{q}_{f=0}$ is already known as it is equation (1) computed by Featherstone's algorithm, without taking into account external force (except gravity). From $\ddot{q}_{f=0}$, we get free linear and angular accelerations by:

$$a_{f=0} = J\ddot{q}_{f=0} + \dot{J}\dot{q}_{f=0}. \tag{9}$$

It is worth noticing that as we solve contact problems in terms of velocities, we can update the dynamics directly in a velocity formulation which is:

$$\dot{q} = \dot{q}_{f=0} + \dot{q}_{f\neq 0} \tag{10}$$

where $\dot{q}_{f=0}$ is the free velocity of the bodies given by integrating $\ddot{q}_{f=0}$ and

$$\dot{q}_{f\neq 0} = A^{-1} J_c^T dt f + \dot{q}^k \tag{11}$$

with $\dot{q}^k$ the joint velocities at one time step before. This way allows integrating just once to get joint positions.

D. Dynamic model with contact

As we do not know when a collision occurs or whether there is contact or not, we must check the contact state. On the contrary of Anitescu's and Stewart's approaches, we use an event-based approach. Contact forces are solved if any contact is present. Before solving equations (2) and (4), we have several parameters to find, which are Λ_c^{-1} and relative velocities and accelerations of contact points. From equation (9), we get B_c by making a projection to the contact space with respect to the three directions (normal and tangential).

To compute Λ_c^{-1}, we need $A^{-1}J_c^T$. Let consider that there is no gravity ($g = 0$), no torques ($\Gamma = 0$), no joint velocities ($\dot{q} = 0$). The free joint acceleration $\ddot{q}_{f=0}$ becomes then zero, so that in equation (7) only the term $A^{-1}J_c^T f$ remains.

Algorithm 1: Resolution of contact forces with friction

Data: $\ddot{q}_{f=0}$ (or $\dot{q}_{f=0}$), Λ_c^{-1}, v_{ci}, B_c
Result: f
begin
  for *each contact* $i \leftarrow 1$ **to** m **do**
    $v_{ci}^{relative} \leftarrow B_c T_e + v_{ci}$
  while *(not convergence)* **do**
    for *each contact* $i \leftarrow 1$ **to** m **do**
      $v_{ci} \leftarrow$ GaussSeidelSolver()
    if $v_{ci}^{normal} = 0$ **then**
      check $\|f_{ti}\| < \mu f_{ni}$
      $f_i \leftarrow$ NewtonContactSolver()
  for *each body* $i = 1$ **to** n **do**
    $\ddot{q}_i \leftarrow \ddot{q}_{f=0i} + A^{-1}J_c^T f_i$ or
    $\dot{q}_i \leftarrow \dot{q}_{f=0i} + A^{-1}J_c^T dt f_i + \dot{q}^k$
end

We compute again the second and the third recursion of Featherstone's algorithm for each contact points, setting f to unit forces with respect to the normal and tangential directions in the contact space as external forces. We obtain from the third recursion unit joint accelerations:

$$\ddot{q}_{f=1} = A^{-1} J_c^T \tag{12}$$

and linear accelerations:

$$a_{f=1} = J_c \ddot{q}_{f=1} = J_c A^{-1} J_c^T \tag{13}$$

We obtain Λ_c^{-1}. For Λ_I^{-1}, as we do not consider friction for impact, we make a projection of Λ_c^{-1} to the normal direction.

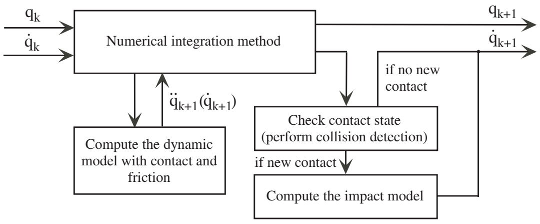
B_I is obtained by integrating relative accelerations of collision (contact) points obtained from $\ddot{q}_{f=0}$ and multiplying it by $(1+\epsilon)$.

In the case one wants to use LCP formulation to solve contact forces, slight modifications have to be made which are the computation of unit joint accelerations with respect to the directions of the discretized friction cones, that means as many directions as facets.

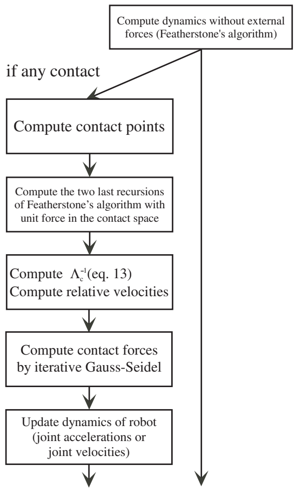
Regarding algorithm complexity, Chang and Khatib [13] proposed a $\mathcal{O}(nm+m^3)$ algorithm for computing Λ_c through the computation of Λ_c^{-1}, with n the number of links and m the number of contacts. Actually, their algorithm for computing Λ_c^{-1} is in the worst case (which means all bodies in contact) $\mathcal{O}(nm + m^2)$ because they manage to avoid unnecessary computation processes, for example they update the dynamics of only bodies in contact, whereas with our method, we update the dynamics of the whole system. Our method is in the best case as efficient as Chang and Khatib's ($\mathcal{O}(nm + m^2)$), but is easier to implement. In the future, we plan to thoroughly investigate this issue. The general algorithm is presented in Fig. 2.

III. IMPLEMENTATION

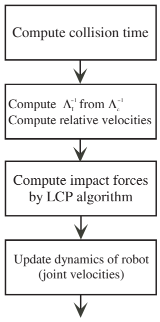
In problems of contact with friction, collision detection is an important issue to avoid bodies penetrating each other.



(a) General overview of one step computation of dynamics



(b) Computation of the dynamic model with contact and friction



(c) Computation of the impact model

Fig. 2. Algorithm for dynamics computation with contact and friction.

Various collision detection libraries have been proposed and for our purpose, we chose PQP[2]. Given the geometry of the bodies, we can either request the pairs of colliding triangles of two bodies, and the minimal distance between these two bodies. The main interesting point with this library lies in the proximity query feature as it allows preventing bodies to penetrate each other by setting all points located in a bounded area around each body as contact points, whereas requesting the pairs of colliding triangles implies the bodies to overlap and meanwhile the computation of contact points with an appropriate algorithm such as Möller's one [14]. It is worth noticing that the last method may result in redundant contact points implying to remove them, thus slowing down the simulation. Detecting overlapping bodies is a common feature of PQP and previously implemented Pierre Terdiman's library, OPCODE[3], which does not integrate proximity query feature. We made a comparison between these two libraries on simple objects using overlapping tests and we saw that using PQP resulted in more acceptable contact points than OPCODE, without any gain of computation time.

Dealing with impact, since dynamic simulation is computed in discrete domain and with a constant time step, our method requires contact states to be considered at two successive times t_0 and t_0+step. As a matter of fact, collision may occur between these two times, at a time t_i that has to be determined, and so the impact force has to be applied at this time on the collision points. Since the exact trajectory of the body is unknown, the impact time may be found using an interpolation between the two positions of the body. For sufficient small time step (0.001sec) one may take the time t_0 as the time of impact.

[2]http://www.cs.unc.edu/ geom/SSV/
[3]http://www.codercorner.com/Opcode.htm

IV. Illustrative examples

To illustrate our work, two demonstration examples have been simulated and another has been conducted also by experiment. All the simulations have been performed using a bi-AMD™64 2.5Ghz processor, with a time step of 0.001sec. We took the HRP-2 humanoid robot (30 dof) as character for our simulations and experiment.

A. Simulations

The first example consists in grasping an object on a table by reaching it. For this purpose, the robot bends forward with its chest and uses one of its hand as a support on the table so that it can reach the object without falling. We make the robot bend sufficiently so that it leans forward. The coefficient of friction has been set to 0.4 between each object of the scene. As the snapshots in Fig. 3 show, when the robot bends forward, the hind parts of the feet start getting off the platform and slipping on it. The table retains the robot falling as the legs of the robot collide the table. The distance between the robot and the table is sufficiently small so that the feet can not slide anymore. By using its left hand as a support on the table, the robot does not rely only on its legs for its global stability and does not tip out to the side.

The second example is a quite similar task as depicted in Fig. 1. We ask the robot to push a cart on which we put a 3kg box. The cart is pushed for some distance, then the robot lifts it up intentionally and release it so that the box falls down. In this scenario, we show that our simulator can handle dynamics computation of many objects in the environment besides the robot, toward a highly multi-contact resolution. Snapshots of the simulation are shown in Fig. 4. The box bounces a bit as the cart moves. As in the previous example, friction cones and contact forces have been illustrated.

Besides these two examples, we performed several tests involving more than 50 contacts and we made a comparison between the Gauss-Seidel approach and the LCP formulation in terms of size of matrices. In the second example there are at most 67 contact points. Using our algorithm, we get for Λ_c^{-1} a square matrix of dimension 201, whereas if we used algorithms with LCP formulation, like for example Miller, with 8-sided friction cones and without taking into account joint constraints, we would get for a square matrix of dimension at least 1734.

In Fig. 5, we show the average CPU time obtained for our simulator from several tests involving many contact points. We can observe that the computation time grows roughly in m^2, with m the number of contact, which corresponds to what we said in section II.D.

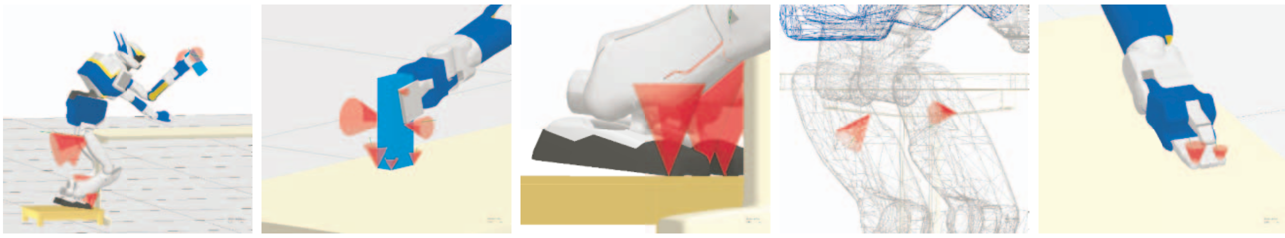


Fig. 3. HRP-2 reaching an object on a table (the last four snapshots show contact points at gripper, feet, legs and hand).

Fig. 4. HRP-2 pushing a cart, then lifting it up and finally releasing it (the last two snapshots show contact points at hands and box).

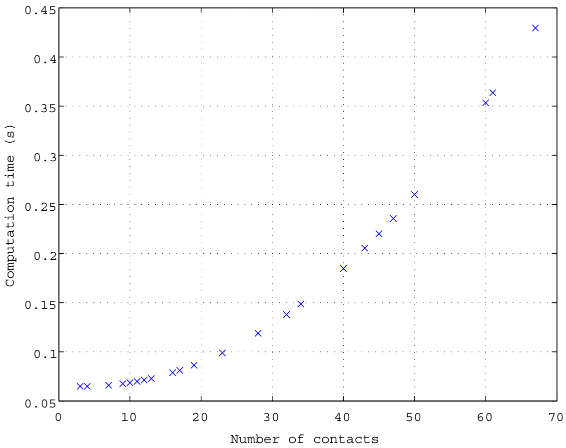


Fig. 5. Average CPU time obtained with Gauss-Seidel approach.

B. Experiment

In order to validate both simulations, we decided to make experiments on the real platform and compare what we get with the simulations. In one experiment we ask the robot to lean forward and make an object fall on a table by pushing it. Because the robot collides soon the table, its feet do not slide. Fig. 6 shows the comparison between the simulation and the experiment. As the snapshots show, in simulation, the robot and the object move in a similar way as in the experiment, especially the object slides while falling, the robot's legs and feet collide the table and the platform respectively without apparent rebound. We measured the sliding distance of the object on both simulation and experiment, and we found a difference of about 2mm after sliding, which means the simulation matches well the experiment and thus our simulator can handle contact problems with friction in a realistic manner, assuming that there may be small visual errors in measurements and we did not take exactly the same parameters like coefficient of friction and model of the object. We are planning to perform exact measurements of position and forces at hand, legs and feet and compute precisely the gap between simulation and experiment. If we take exactly the same parameters and we improve dynamics computation, we expect that the gap between simulation and experiment will nearly disappear.

V. Conclusion

This paper presented a new dynamics simulator for humanoid robots with contact and friction and solving them by using operational space and constraint-based method. We showed that we can consider friction without cone discretization which results in an efficient algorithm. We also showed the effectiveness of our proposed algorithm by experiment.

We should have a closer look to the computation time to allow haptic interaction. We plan to implement a unified approach for impact and resting force. We should also optimize the code for dynamics computation and test other integration methods. We will devise a dedicated module for torque control.

References

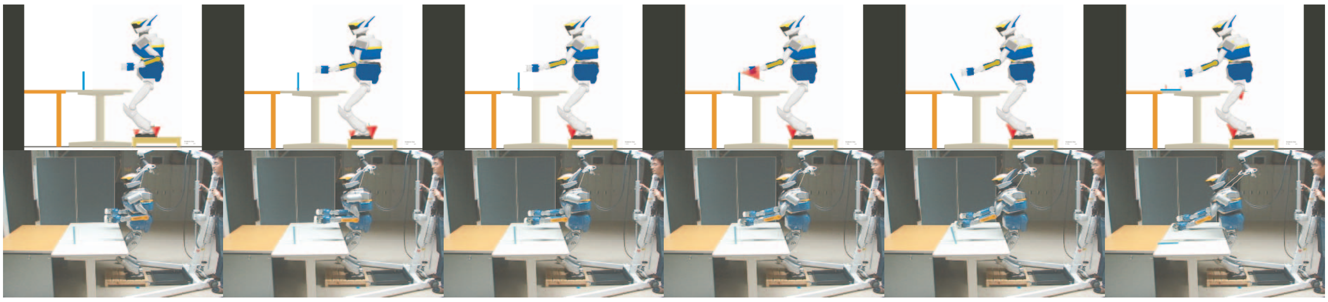
[1] K. Yamane and Y. Nakamura, "Stable penalty-based model of frictional contacts," in *IEEE International Conference on Robotics and Automation*, 2006.

[2] E. Kokkevis, "Practical physics for articulated characters," in *Game Developers Conference*, 2004.

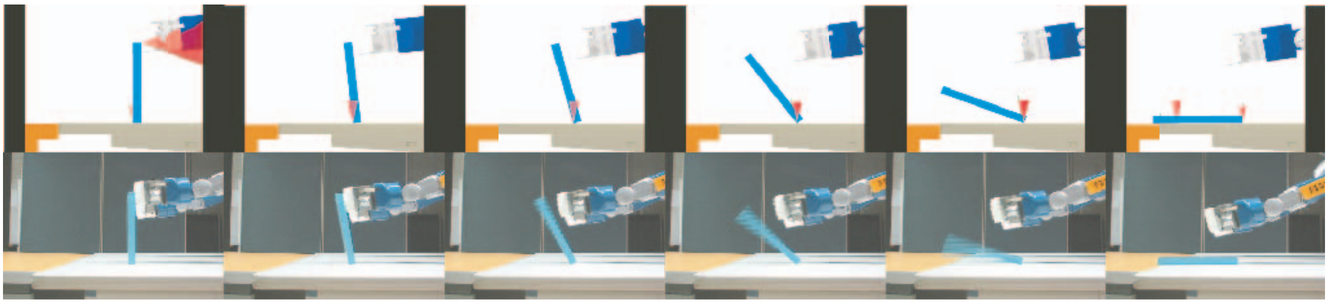
[3] D. Baraff, "Fast contact force computation for nonpenetrating rigid bodies," in *SIGGRAPH*, 1994.

[4] M. Anitescu and F. A. Potra, "A time-stepping method for stiff multibody dynamics with contact and friction," *International Journal for Numerical Methods in Engineering*, 2002.

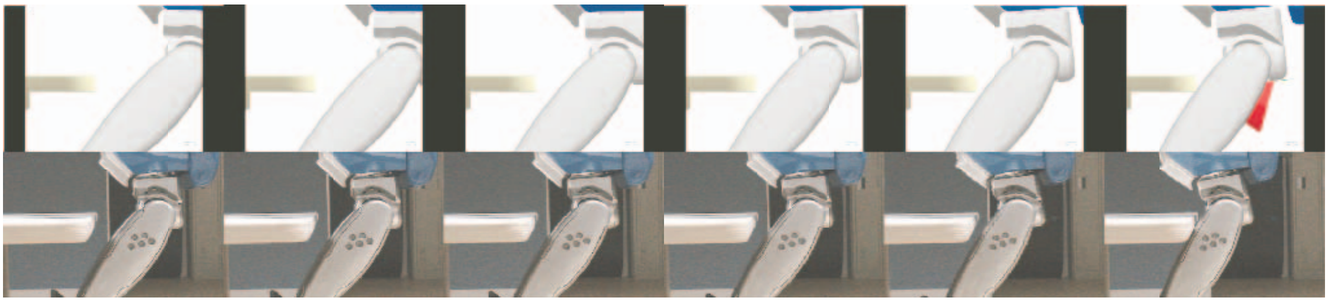
[5] D. Stewart and J. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with coulomb friction," in *IEEE International Conference on Robotics and Automation*, 2000.



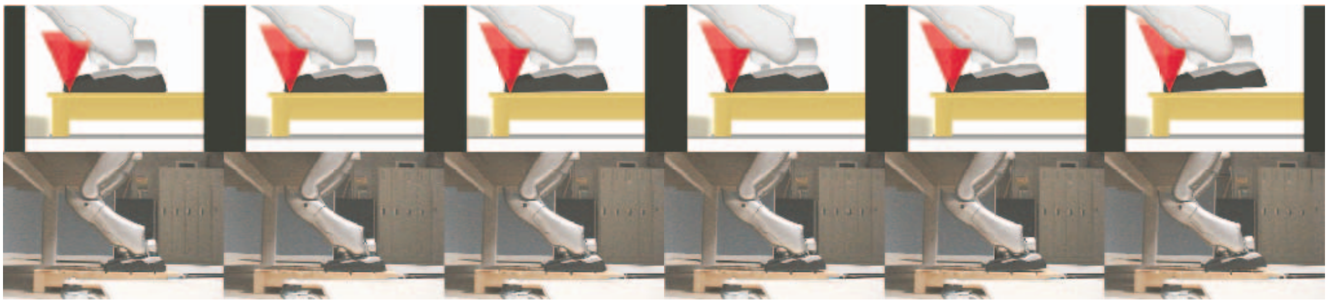
(a) General overview.



(b) The robot's hand collides the object making it fall and slide.



(c) The robot's legs collide the table without rebounding.



(d) The hind part of the feet heels up as the robot leans forward but the feet do not slide.

Fig. 6. HRP-2 leaning forward and pushing an object (top is simulation and bottom is experiment).

[6] H. I. C. Andrew T. Miller, "Implementation of multi-rigid-body dynamics within a robotic grasping simulator," in *IEEE International Conference on Robotics and Automation*, 2003.

[7] D. C. Ruspini and O. Khatib, "Collision/contact models for dynamics simulation and haptic interaction," in *International Symposium of Robotics Research*, 1999.

[8] T. Liu and M. Y. Wang, "Computation of three-dimensional rigid-body dynamics with multiple unilateral contacts using time-stepping and gauss-seidel methods," *IEEE Transactions on Automation Science and Engineering*, 2005.

[9] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot, "Realistic haptic rendering of interacting deformable objects in virtual environments," *IEEE Transactions on Vizualization and Computer Graphics*, 2006.

[10] R. Featherstone, *Robot dynamics algorithms*. Kluwer Academic Publishers, 1987.

[11] B. Mirtich, "Impulse-based dynamic simulation of rigid body systems," Ph.D. dissertation, University of California at Berkeley, 1996.

[12] M. Jean, "Numerical methods for three dimensional dynamical problems," in *Conference Contact Mechanics, Southampton*, 1993.

[13] K.-S. Chang and O. Khatib, "Operational space dynamics: Efficient algorithms for modeling and control of branching mechanisms," in *IEEE International Conference on Robotics and Automation*, 2000.

[14] T. Möller, "A fast triangle-triangle intersection test," *Journal of Graphics Tools*, 1997.