U. Battista, F. Cepolina, S. Costo, M. Zoppi “Videogame for safe flights”, CLAWAR/EURON Workshop on Robots in Entertainment, Leisure and Hobby- ELH '04, Wien 2-4 December

Videogame for safe flights

Umberto Battista, Francesco Cepolina, Sandro Costo, Matteo Zoppi
PMAR laboratory of Design and Measurement for Automation and Robotics
Via All’Opera Pia 15 A - 16145 Genova, Italy
E-mail: {battista, cepolina, costo, zoppi}@dimec.unige.it http://www.cepolina.com

Abstract

Terrorism is a very actual problem. Although terrorist attacks came in succession from the early seventies and eighties, recent events have raised the technological scale and the dimension of the actions. High level of security is required and bomb-disposal experts are frequently exposed to risks in mission. The effects on the society are blameworthy. Media like television and radio, tend to scare people; travels and flights become less attractive. This trend risks to flow into cultural and religious forms of racism. Entertainment and leisure are means that can help to fight against this fear. The paper describes the development of a videogame that simulates the rescue of a potential bomb from an airplane by means of a mobile robot. The user can control real-time the robot, using the keyboard. The device, actuated by two crawlers, carries a 7 DoF arm and moves inside a virtual airplane. The videogame, based on Open Dynamics Engine (ODE), supports dynamics and collision detection. The graphic engine Open Scene Graph (OSG) enhances the reality of the game. The system has been also designed for training of bomb-disposal experts that will maneuver explosive ordnance disposal robots.

1 Introduction

A simulation game is composed of three main elements: a scenery, one ore more characters and some rules. The characters can be real or fictional. Users control, for example, airplanes, helicopters or robots and try to accomplish missions. These games allow to analyze the behavior of certain characters in a constrained environment. Videogames can have different levels of abstraction. A simulator may reduce the complexity of a real system or extrapolate only some main aspects, highlighting some details not completely clear by the trend or data analysis.
Usually, for detailed/realistic simulations, the user needs a 2D/3D mouse in combination with several keys combinations. Often many players can interact in the same environment; some examples are surgery theatre simulation, racing or role-playing games. Simulators can be used for amusement, training or to predict likely outputs from a real system.
A remotely controlled robot is able to enhance the safety of the artificers [11] and, at the same time, to execute successfully the task. Since water cannons and micro-charges, typical equipment of robots that operate outdoor, cannot be used inside an airplane, any other technique that allows to identify potential explosive devices is extremely valuable: cameras, X rays, etc.
The robots existing on the market can be classified depending on “what” they are able to handle (mass and size of the object) and “where” they should operate (indoor, outdoor, structured environment). The mass and the size of the object are crucial for the correct design of the arm and of the grasping device.
Commercial EOD (explosive ordnance disposal) robots derived from military robots cannot operate in narrow spaces such as planes and busses.
Ideally a complete equipment for EOD tasks is composed of:

- a big size robot, for open space missions, equipped with a water cannon
- a small size robot to work in restricted spaces
- a miniature robot for the inspection of vehicles.

Commercially available EOD robots are derived from devices used for the treatment of dangerous substances or the achievement of military tasks. Robots specially suitable for EOD have been designed only later. These robots can be classified by size.
Small size robots, about 10 kg, can easily accomplish inspection tasks. Bigger size robots, up to 300 kg or more, can work outdoor, can lift heavy objects, but due to their size, they have a low dexterity. Middle size robots show intermediate performances.
The robots can have wheels or crawlers. Crawlers, despite their limited speed, offer several advantages compared to wheels: high torque, capability to avoid obstacles, zero curvature ray and low center of gravity.
Vanguard, from *EOD Performance*, is the only robot commercially available equipped with a really dexterous arm and that could be employed, in principle, in an airplane; the other EOD robots, suffer lack of arm dexterity, and generally have a cart too cumbersome.

2 The PMAR-EOD Robot

The mission environment, for EOD robots operating in airplanes, includes narrow corridors between the banks of seats and along the plane, and places difficult to reach, like the inside of the baggage rooms above the seats or the service cabinets. The EOD robot designed at the PMAR-lab of the University of Genova, named PMAR-EOD (Fig. 1), is a mobile system tailored for this complex environment. It is composed by a crawled base and an arm. The 7R arm is agile and light.
The design process of the PMAR-EOD availed of fully implemented digital mock-ups and extended virtual test campaigns.

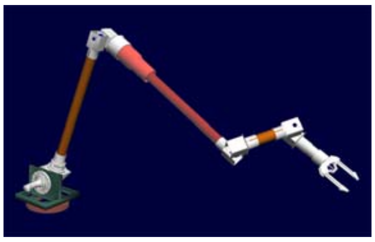

Figure 1. ***The PMAR-EOD robot***

A 3D mock-up of an airplane was prepared to simulate the robot in action (Fig. 2).

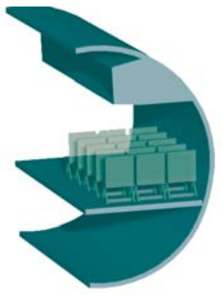

Figure 2. ***Cockpit model***

The robot is able to carry a payload up to 3 kg. The arm is sized to reach all "sensible" places inside of an airplane. Its frame is made of aluminum and composite hollow pipes that envelope the cables, the actuators and the gears. The overall weight is low compared to the payload.
The clamp (Fig. 3) has a compact design and its architecture and geometry are selected to grasp both big and small objects, from 4 mm to 100 mm diameter. The fingers use four-bar mechanisms assuring the parallelism of the tips and a linear force/displacement characteristic.

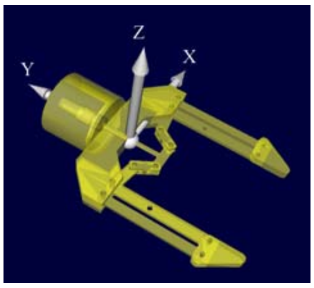

Figure 3. ***The clamp***

3 The videogame

Choice of the virtual simulation tool
For any system and machine, virtual simulation is much cheaper than any physical, partial or complete, prototype and the assessment of the system performances is faster.
Code libraries for the definition of environments for virtual simulation are an effective mean for custom made implementation of specific simulation software. The result is remarkably different from traditional CAD results, since kinematics/dynamics analyses, motion planning and collision check are enabled in the same environment. Several virtual simulation libraries and packages exist, some oriented to robotics, other general-purpose. Most of these code libraries, packages and environments are freeware and simply downloadable from the web. A rough subdivision is: freeware/open-source packages, university packages (e.g., developed in PhD theses or such and owned by a research team) and commercial software. After a comparison of the candidate simulation packages, it has been chosen to adopt the C library ODE.

ODE – Open Dynamics Engine
ODE is an open-source C/C++ library, implemented by Russel Smith. It can be used for real-time, multi-body, dynamic simulation of rigid bodies [14], with high level interaction facilities, such as motion commands and monitoring. The library supports a satisfactory set of joints and contact fixtures, also with friction. Hard contact conditions are implemented and a basic collision detection is provided, supporting elementary geometries such as cylinders, spheres, capsules, planes, lines and triangular meshes. Other more complex contact features was developed by ODE users and are available. The integration is performed by a first order (Euler) algorithm that is not accurate but intrinsically stable.
The main features of ODE are: - rigid bodies with arbitrary distributed mass properties; - spherical, revolute, prismatic, universal joints, weld connections and linear and rotational actuators; - Trinkle-Stewart [19-22] and Anitescu-Potra [1-4] algorithms for the solution of the velocity equations of the simulated systems; - LCP Dantzig solver for the modeling of contacts with friction [5-10] and a faster Coulomb friction modeling; - C++ interface on the native C interface; - OpenGL graphic library for real time representation of the simulation results; - run in single and double precision floats.
The steps of a typical simulation are:
1. implementation of a dynamic environment;
2. generation of bodies in the virtual dynamic environment;
3. definition of the pose of all bodies (orientation and location);
4. generation of joints in the virtual dynamic environment;
5. definition of the connections between bodies and joints;
6. setup of the joint parameters;
7. implementation of a collision environment and of body shadow geometries for the contact simulation;
8. setup of a joint group for the contact simulation;
9. loop (this section of the code is the body of the simulation to be run):
   - forces are applied to the bodies
   - the joint coordinates are controlled
   - the collision detection runs
   - virtual contact joints are generated, one per each active contact point; these joints are inserted in the joint group for the contact simulation
   - the equation of motion are integrated numerically
   - the contact joints are removed from the joint group;
10. the dynamic and the collision environments, together with the joint group, are cleared.

4 The implemented simulator

Each object is represented in the virtual environment as three separate virtual entities: a body, representing the mass properties, used for the dynamic simulation; a geometry used for the collisions detection; a rendered geometry to provide a virtual-reality representation of the simulation. Each of these three entities has its own location and orientation coordinates, which usually coincide. In this way, the three simulation layers are kept independent each other and the implementation of the model is simplified. Note that the simulation can be performed without being associated to any graphical virtual-reality representation, which is useful just for a faster understanding of the results.

Architecture of the simulator

The virtual simulation of the PMAR-EOD robot is performed in double precision. The simulator is implemented in C++ on eight levels:

1. kinematics model of the robotic arm
2. simple dynamic model of the robotic arm
3. introduction of collision detection of the arm links each other
4. accurate dynamic model of the arm
5. introduction of the environment and related collision detection
6. setup of the control system
7. graphic engine for the rendering
8. tuning and tests on the complete simulator

These levels are briefly described in the following sections.

Arm kinematics model

Featherstone's algorithm [13] is fast and can be used for chain-like structures, but problems are encountered when there are contacts or if the chain needs to be broken to add another object. Mirtich type methods [15], where one constraint at a time is satisfied, are difficult to make stable, and objects suffer from jitter.

The arm kinematics model (Fig. 4) is solved with the *RRG Kinematix* library, implemented by the Robotics Research Group of the University of Texas at Austin.

The geometry is described by the Denavit-Hartenberg parameters [12], measured on the Pro/ENGINEER virtual mock-up of the robot. The joint zeros chosen (arm fully deployed frontally and parallel to the ground) differ from the Denavit-Hartenberg representation, so a conversion routine is required.

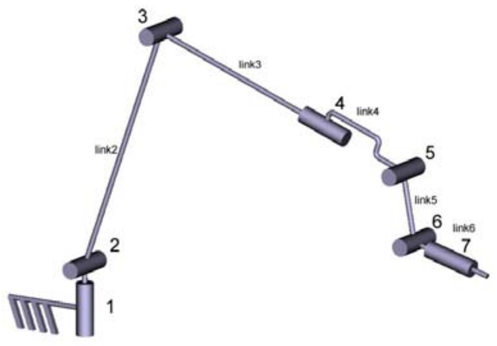

Figure 4. ***Kinematics architecture of the arm***

The simulation steps are:

1. Choice of a feasible set of joint values
2. Creation of a handling to control the robot; the D-H parameters are assigned
3. Run of the forward kinematics; the position of the end-effector is computed
4. Main loop:
    1. Wait for keyboard commands from the user. The current arm configuration is stored and changed according to the keyboard input;
    2. Solution of the inverse kinematics:
        a. If no feasible solution is obtained, the previous saved configuration is kept
        b. If the Measure Of Manipulability (MOM) is lower than a specified threshold [23, 24], then the inverse kinematics is solved by "Singularity-Robust Inverse" (SR-inverse) algorithm [16, 17] (SR-inverse computed with Matrix TCL Lite library). If some joint overcome the specified angular mobility, the previous saved configuration is kept
    3. Output of the solution

Simple arm dynamic model

A simple dynamic model of the arm is implemented using ODE functions (Fig. 5). The mass properties of the links are approximated. Each link is a constant-density cylinder without reaction force limitation. The inverse kinematics is solved again using *RRG Kinematix*. Additional joint angle conversions routines are used to link *RRG Kinematix* to ODE.

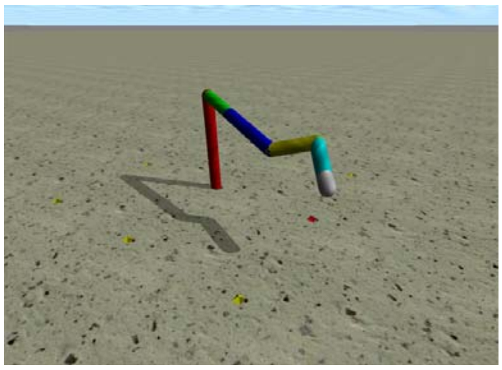

Figure 5. ***Links of simple arm dynamic model***

The pose of the gripper (end-effector) changes accordingly to the keyboard input coming from the user. The inverse kinematics routine provides the input for simple PID loops controlling the joints.

Collisions

A geometry is associated to each link (Fig. 6); the simulator is provided with a collision detection routine. This routine considers the interaction between non-subsequent links, and between links and the virtual environment.

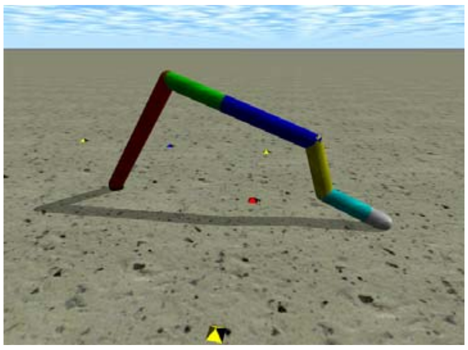

Figure 6. ***Arm collision detection***

Accurate dynamic model

The arm and the mobile platform of the robot are modeled with basic solids (Fig. 7). The mobile platform moves front and back and steers. The crawlers are replaced by four cylindrical wheels; the platform steers due to different rotation velocities of the left and right wheels.

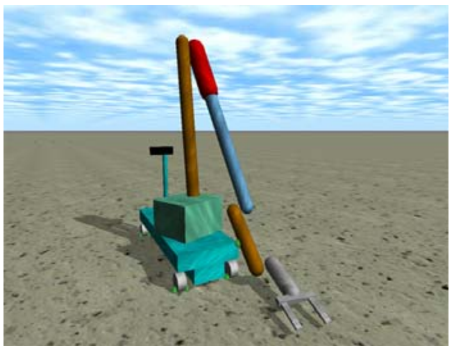

Figure 7. ***Accurate dynamic model of the system***

The tangential component of the wheel-ground contact force depends on the orthogonal component (by the friction factor) and on the actuation torque applied to the wheel.
In the final implementation of the algorithm, each simulation loop corresponds 0.01 s advance. On an Athlon XP 2000+ with 512 MB SDRAM, the simulation runs at 43 frames per second, i.e., 1 s run corresponds to 0.43 s simulated. Real time simulation is possible by adapting the integration step at each step; this solution has been abandoned due to convergence problems.

Environment
An accurate modeling of the environment has fundamental importance to get significant results from the simulation. The aircraft interior has been created with Pro/ENGINEER. The model has been exported to AC3D format and further filtered; each surface has been represented with a triangular mesh. Finally the robot is inserted in the new environment (Fig. 8).

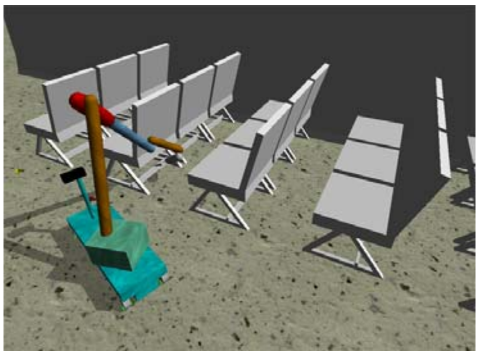

Figure 8. ***Modeling of the aircraft environment***

Control system
The user can select one of three control schemes developed for the PMAR-EOD robot:

- *Direct dynamics*
- *Inverse dynamic - Global mode*
- *Inverse dynamic - Local mode*

During the simulation, it is possible to switch from one mode to another. The passage from one to any another is always possible during the simulation. PID controllers are used for all joints in all three cases [17-18].
The control of the mobile platform is independent from the control of the arm, thus the pose of the arm gripper refers to the platform and it is not possible to keep the gripper fixed while the platform moves.
The initial control imposed torque thresholds to the motors together with a torque control scheme, but this led to instability. Then the velocity control scheme natively supported by ODE has been adopted.
In the case of *Direct dynamics* scheme (Fig. 9), each joint is controlled separately from the others in the joint space. This scheme is used for large arm displacements while reaching the workspace region containing the target object to be disposed, e.g., in the bottom-front of a seat.

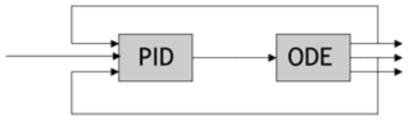

Figure 9. ***Direct dynamics***

The *Inverse dynamics* scheme (Fig. 10), allows the control of the robot in the end-effector space. In *Global mode* the pose of the gripper is controlled with respect to the mobile platform reference frame, while in *Local mode* the gripper is controlled with respect to its own reference frame.

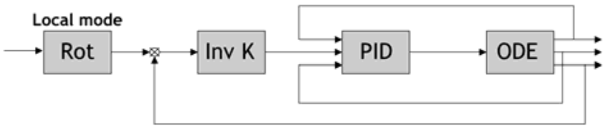

Figure 10. ***Inverse dynamics***

The normal robot operation sequence is:

- *Direct dynamics* for fast approach of the workspace region where the target object relies
- *Inverse dynamics – Global mode* to make the gripper closer to the target
- *Inverse dynamics – Local mode* to reach the target

Graphic engine for rendering
A realistic representation of the robot in the environment, together with an impressing rendering is a useful option to make better advantage of the PMAR-EOD simulator (Fig. 11).
The graphic engine used is Open Scene Graph (OSG), an open source library for advanced 3D graphics that is normally used for videogames, virtual reality and modeling. It is written in C++ and uses OpenGL.
The pose of each body is computed by ODE and transferred to OSG, which place the bodies in the simulation space. The 3D models of every link and component of the system come from Pro/ENGINEER, from which they are converted in AC3D format, directly supported by OSG. During the simulation ODE computes the poses of the bodies and transmits them to OSG. The real-time data visualization is provided by a suitable HUD, switched on/off by the user through the keyboard.
The rendering process slows remarkably the simulation, up to 16 frames per second on an Athlon XP 2000+ with 512 MB SDRAM and GeForce 2 Mx 32 MB video card (Fig. 11). Although the simulation is not real time, the delay is small and the user can easily perceive and com-

mand the robot. With an improved hardware, the simulation could also be performed real time.

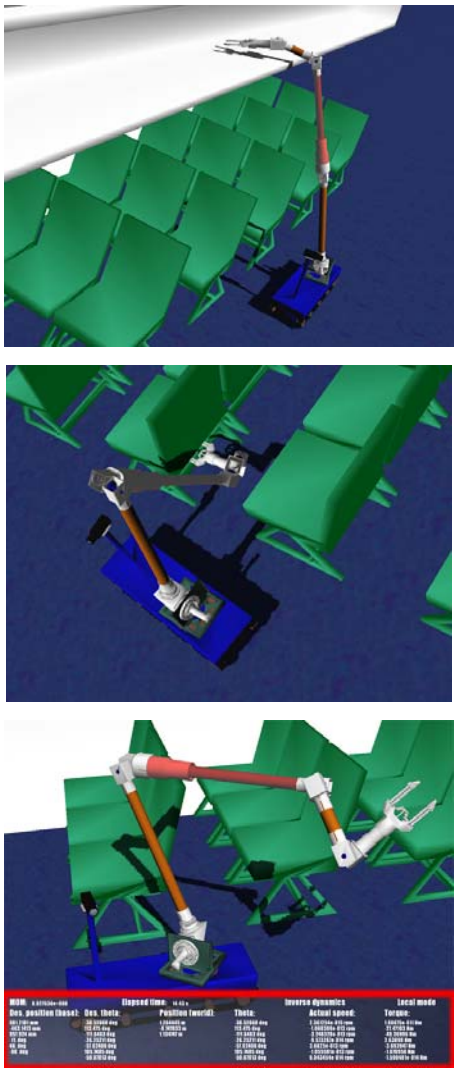

Figure 11. ***Graphic representation of the robot***

User interface

The user can interact with the simulation through the mouse and the keyboard. The former controls the camera, while the latter commands the robot and some parameters of the simulation.

The keyboard is managed by a keyboard handler class of OSG. At each time step, the keyboard handler class checks for a key press. If a key is currently pressed, then the action associated to it is performed, otherwise the robot remain still, waiting for an input.

A key press leads to an increase of the variable associated to it. A decreasing is obtained by pressing the Shift button as well.

In *Direct dynamics* each joint has one key associated to it. In *Inverse dynamics* three keys move the gripper along the x, y and z axes of the actual reference frame, while three other keys control its rotation around the same axes.

The crawled base is controlled by the arrow keys. They suitably increase/decrease the speeds of the two crawlers.

The user can also disable/enable gravity and show/hide the data on the HUD.

5 Tuning and tests

The tuning of the simulator was a high time-consuming phase of the work. High accuracy was required together with a high stability of the simulator. These are opposite requirements, so a compromise had to be found.

The main parameters to be tuned in ODE are:

- Error Reduction Parameter (ERP)
- Constraint Force Mixing (CFM)
- integration step

Error reduction parameter (ERP)

This parameter modifies a special force applied by the physics engine to reduce joint error and maintain the bodies in the correct pose. ERP has an influence on how much of this error will be accepted at the actual step and compensated at the further step.

Constraint force mixing (CFM)

The contact constraint is as stiff as lower the CFM. This parameter is used to tune the surface properties of the bodies with respect to contact.

Main modeling parameters

- ERP = 0.8 and CFM = 10^{-10} for high accuracy
- timestep = 0.01 s
- the integration function is *dWorldStep()*, as it provides the best accuracy and stability
- the gains used for the PID controller of all the joints of the arm are kp = 20, kd = 0.1 and k = 0.01. They guarantee a quick and stable response. A better tuning of the gains for each joint could be made, but the gains for the real robot would probably be different
- μ = 10 is the friction coefficient used to model the friction between the crawler tracks and the floor. Smaller values reduce the rotation velocity of the base. So, even if this value is not really realistic, the result is good
- joints are without stops. In this way, it is possible to measure the span required to perform the different task. This is a useful information to validate the design of the robot
- joints' CFM is set 10^{-10} in order to introduce some elasticity

Tests

Two main test campaigns were performed in order to check both the quality of the virtual reality simulation, on one hand, and the performance of the real robot that will be realized. The tests were planned and the results assessed with the help of bomb-squad agents.

A set of significant missions was selected and the corresponding simulations run. The mass of the considered payload for these simulations was 3 kg. The actuation torques was measured at every simulation step (Fig. 12 and 13) and the obtained plots used to validate the choice of the actuation motors.

The robot shows able to reach all regions of the inside aircraft, in particular the bottom of the window seats and the hat-boxes of the queue seats. The level of reality was judged satisfactory for training of bomb squad agents as well as to play true-like missions.

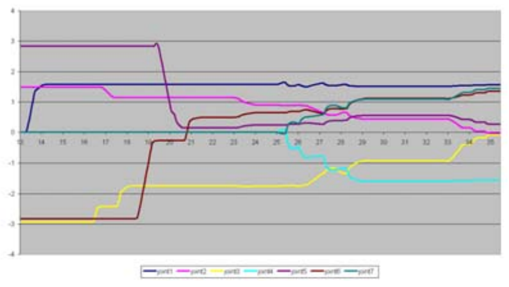

Figure 12. ***Joint's angles vs. time***

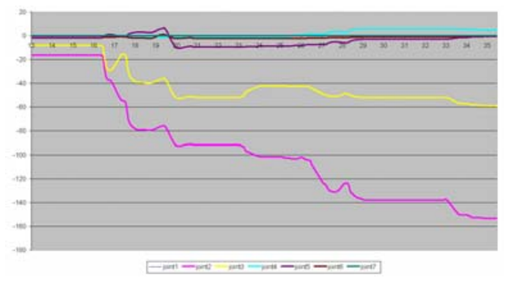

Figure 13. ***Gravity torque vs. time***

6 Conclusions and future work

A dynamic 3D videogame is realized: the mission is to find and retrieve a bomb placed inside an airplane. The player controls, through the keyboard, a mobile robot; the security robot can pick the bomb by means of a gripper positioned on the tip of its redundant arm. The PMAR-EOD simulator is suitable either for training of the airport bomb squads either as a core of a videogame. The game may be inspired by some recent events: terrorist attack, bombs, sabotages etc. with the aim of making people rationally conscious instead of unconscious or irrationally sensible.

Both the left side and the right side of the cockpit are simulated. Crucial data like maximum angular range of each joint and maximum torque were computed. The simulation environment and the model of the PMAR-EOD were used both to check the correctness of the design of the robot and to validate the selected geometry with respect to the missions.

The simulation speed is limited only by the CPU performance. For this reason, more powerful computers will allow to execute real time simulators of complex systems with arbitrary temporal steps and at the same time high stability and accuracy. Moreover, some simplifications would fast the simulation speed in order to make the software compatible with less powerful hardware.

The modular and object oriented structure of the simulator allows to upgrade and modify the code; new features can be implemented with relatively low effort. Moreover ODE is continuously improved and upgraded; in the future new program features will allow to enhance the accuracy, stability and speed of the simulations. Similarly OSG is continuously upgraded in order to offer always more realistic simulations.

References

[1] Anitescu M., Cremer J., Potra F.A., 1995, Formulating 3D Contact Dynamics Problems, Mechanics of Structures and Machines, 22/4: 405-437

[2] Anitescu M., Potra F.A., 1997, Formulating dynamic multi-rigid-body problems with friction as solvable linear complementarity problems, Reports on Computational Mathematics, 93

[3] Anitescu M., Potra F.A., Stewart D.E., 1998, Time-Stepping for Three-Dimensional Rigid Body Dynamics, Comp. Methods Appl. Mech. Eng., 177/3-4: 183-197

[4] Anitescu M., Potra F.A., 2001, A Time-Stepping Method for Stiff Multibody Dynamics with Contact and Friction, Argonne National Laboratory, Argonne, IL

[5] Baraff D., 1989, Analytical Methods for Dynamic Simulation of Non-Penetrating Rigid Bodies, Computer Graphics, 23/3: 223-232

[6] Baraff D., 1990, Curved Surfaces and Coherence for Non-Penetrating Rigid Body Simulation, Computer Graphics, 24/4: 19-28

[7] Baraff D., 1990, Coping with Friction for Non-Penetrating Rigid Body Simulation, Computer Graphics, 24/4

[8] Baraff D., 1992, Dynamic Simulation of Non-Penetrating Rigid Bodies, PhD Thesis

[9] Baraff D., 1993, Issues in Computing Contact Forces for Non-Penetrating Rigid Bodies, Algorithmica, 10: 292-352

[10] Baraff D., 1994, Fast Contact Force Computation for Non-Penetrating Rigid Bodies, Computer Graphics Proceedings, Annual Conference Series, Orlando: 23-34

[11] Bowen I.G., Fletcher E.R., Richmond D.R., 1968, Estimate of Man's Tolerance to Direct Effects of Air Blast, Technical Progress Report, DASA-2113, Defense Atomic Support Agency, Department of Defense, Washington, D.C.

[12] Craig J., 1989, Introduction to Robotics: Mechanics and Control, Addison-Wesley, Reading, MA

[13] Featherstone R., 1987, Robot Dynamics Algorithm, Kluwer Academic Publishers, Norwell

[14] Haug E.J., 1989, Computer Aided Kinematics and Dynamics of Mechanical Systems, Allyn and Bacon, Needham Heights

[15] Mirtich B.V., Canny J., 1995, Impulse-Based Dynamic Simulation of Rigid Body Systems, PhD Thesis, University of California, Berkeley

[16] Nakamura Y., 1991, Advanced Robotics: Redundancy and Optimization, Addison-Wesley, Boston, MA

[17] Sciavicco L., Siciliano B., 2000, Robotica Industriale: Modellistica e Controllo di Manipolatori, McGraw-Hill, Milano

[18] Spong M., Vidyasagar M., 1989, Robot Dynamics and Control, Willey, New York

[19] Stewart D.E., Trinkle J. C., 1997, Dynamics, Friction, and Complementarity Problems, Complementarity and Variational Problems: State of the Art, Proc. 1995 International Conference on Complementarity Problems, Philadelphia: 425-439

[20] Stewart D.E., 2000, Time-Stepping Methods and the Mathematics of Rigid Body Dynamics, Chap. 9 in Impact and Friction of Solids, Structures and Machines, Vol. 1, Birkhäuser, Boston

[21] Stewart D.E., Trinkle J.C., 2000, An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Inelastic Collisions and Coulomb Friction, Proceedings of the International Conference on Robotics and Automation (ICRA)

[22] Trinkle J.C., Pang J.-S., Sudarsk S., Lo G., 1997, On Dynamic Multi-Rigid-Body Contact Problems with Coulomb Friction, Zeithschrift fur Angewandte Mathematik und Mechanik

[23] Yoshikawa T., 1984, Analysis and Control of Robot Manipulators with Redundancy, Robotics Research, eds. M. Brady and R. Paul, MIT Press, Cambridge, MA: 735-747,

[24] Yoshikawa T., 1985, Manipulability of Robotic Mechanisms, Int. J. Robotics Res. 4/2: 3-9