Bulletin of the *Transilvania* University of Braşov
Series I: Engineering Sciences • Vol. 5 (54) No. 1 - 2012

SIMULATION OF ARTICULATED ROBOTS FOR VIRTUAL PROTOTYPING IN DYNAMIC 3D ENVIRONMENTS

A. FRATU[1] **M. FRATU**[2]

Abstract: *This paper deals with the simulation of a dynamical system in which the motion of each rigid robot is subject to the influence of virtual forces induced by geometric constraints. These constraints may impose joint connectivity and angle limits for articulated robots, spatial relationships between multiple collaborative robots, or have a robot follow an estimated path to perform certain tasks in a cycle. In this paper the authors give a brief overview of a general simulation framework, describing the primary tasks which a simulator needs to implement. The robot behavioral simulation in the virtual environment enables us to predict the behavior of a given real manipulator into real environment.*

Key words: *virtual environment, virtual prototype, virtual assembly, estimated path.*

1. Introduction

The robot behavioral simulation enables us to predict the behavior of a given manipulator under given initial conditions, applied torques, and applied loads.

The ability of predicting this behaviors is important for several reasons: for example, in design the designers want to know whether with a given selection of actuators, the manipulator will be able to perform a certain typical task in a given time frame; in creating feedback control schemes, where stability is a major concern, the control engineer cannot risk a valuable piece of equipment by exposing it to untested control strategies. Therefore, a facility capable of predicting the behavior of a robotic manipulator, or of a system at whole, for that matter, becomes imperative.

In this paper, the authors present a new motion planning algorithm for virtual prototyping. This algorithmic structure is inspired by constrained dynamics in physically-based modeling.

The authors seek to deduce a virtual geometry of the objects, a 3D geometric realization of a collection of rigid bodies is visible in the drawing. The authors transform the motion planning problem into a dynamical system simulation by treating each robot as a rigid body or a collection of rigid bodies moving under the influence of all types of constraint forces in the virtual prototyping environment.

These may include constraints to enforce joint connectivity and angle limits for articulated robots, constraints to enforce a spatial relationship between multiple collaborative robots, constraints to avoid

[1] Dept. of Automatics, Electronics and Computers, *Transilvania* University of Braşov.
[2] Dept. of Installations for Constructions, *Transilvania* University of Braşov.

obstacles and self-collision, or constraints to have the robot follow an estimated path to perform certain tasks in a cycle.

Proposed constraint-based planning structure has the following characteristics:

- It can handle both static environments with complete geometric information or dynamic scenes with moving obstacles whose motion is not known a priori.
- It is applicable to both rigid and articulated robots of arbitrarily high degrees of freedom, as well as multiple collaborative agents.
- It allows specification of various types of geometric constraints.
- It runs in real time for modestly complex environments.

The authors demonstrate the effectiveness of this structure for the problem of virtual assembly prototyping with applications in assembly line planning.

2. Algorithm for Analytical Simulation

In simulation studies, the authors need to integrate the system of ordinary differential equations (ODE) describing the dynamics of a robotic mechanical system.

The authors use a model relating the state of the system with its external generalized forces of the form:

$$\dot{\mathbf{x}} = \boldsymbol{f}(\mathbf{x}, \mathbf{u}), \tag{1}$$

where $\mathbf{x}$ is the state vector, $\mathbf{u}$ is the input or control vector, $\mathbf{x}_0$ is the state vector at a certain time t_0, and $\mathbf{f}(\mathbf{x}, \mathbf{u})$ is a nonlinear function of $\mathbf{x}$ and $\mathbf{u}$, derived from the dynamics of the system.

The state of a dynamical system is defined, in turn, as the set of variables that separate the past from the future of the system. Thus, if we take t_0 as the present time, we can predict from Eq. (1) the future states of the system upon integration of the initial-value problem at hand, even if we do not know the complete past history of the system in full detail.

Now, if we regard the vector $\boldsymbol{\theta}$ of independent joint variables and its time-rate of change, $\dot{\boldsymbol{\theta}}$ as the vectors of generalized coordinates and generalized speeds, then an obvious definition of $\mathbf{x}$ is:

$$\mathbf{x} = \left[\theta^T \dot{\theta}^T\right]^T. \tag{2}$$

The n generalized coordinates, $\boldsymbol{\theta}$ define the configuration of the system, while their time-derivatives determine its generalized momentum. Hence, knowing $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$ can predict the future values of these variables with the aid of Eq. (1).

The authors use the mathematical model, Eq. (1), explicitly, as pertaining to the serial manipulators, in terms of the kinematic structure of the system and its inertial properties, i.e., the mass, mass-center coordinates, and inertia matrix of each of its bodies. To this end, the authors first write the underlying system of dynamical equations for each link. We have n+1 links numbered from 0 to n, which are coupled by n kinematic pairs.

The following step of this derivation consists in representing the coupling between every two consecutive links as a linear homogeneous system of algebraic equations on the link twists. Moreover, all kinematic pairs allow a relative one-degree-of-freedom motion between the coupled bodies. It can then express the kinematic constraints of the system in linear homogeneous form [2], [6].

The procedure whereby the motion of the manipulator is determined from initial conditions and applied torques $\boldsymbol{\tau}(t)$ and loads, is known as simulation.

Since the authors start with a second-order n-dimensional nonlinear ODE system in the joint variables of the manipulator, the authors have to integrate this system in order to determine the time-histories of all joint variables grouped in the joint variables vector, $\boldsymbol{\theta}$.

With current software available, this task has become routine work, the user being freed from the quite demanding task of writing code for integrating systems of ODE. The implementation of the simulation-related algorithms is possible with the available commercial software packages.

As a rule, simulation code requires that the user supply a state-variable model of the form Eq. (1) of the robot dynamic model, with the state-variable vector, **x** and the input or control vector **u,** defined as:

$$\mathbf{u}\ (t) = \boldsymbol{\tau}\ (t). \tag{3}$$

With the above definitions, then the authors can write the state-variable equations, in the form of Eq. (1), with $\mathbf{f}\ (\mathbf{x}, \boldsymbol{\tau})$ thereby obtaining a system of $2n$ first-order ODE in the state-variable vector.

Various methods are available to solve the resulting initial-value problem, all of them being based on a discretization of the time variable. If the behavior of the system is desired in the interval $t_0 \leq t \leq t_F$, then the software implementing this algorithm provides approximations $\{y_k\}^N$ to the state-variable vector $\mathbf{x}(\mathbf{t}_k) = \mathbf{x}_k$, and the value of torques $\boldsymbol{\tau}\ (t_k)$ at a discrete set of instants $\{t_k\}$.

The variety of methods available to solve the underlying initial-value problem can be classified into two main categories, explicit methods and implicit methods. The former provide $\boldsymbol{y}_k$ explicitly in terms of previously computed values. On the contrary, implicit methods provide $\boldsymbol{y}_k$ in terms of previously computed values and itself.

Commercial software for scientific computations provides routines for both implicit and explicit methods, the user having to decide which method to invoke.

3. Simulation Robots' Motion

The robots' motion should be animated with the highest degree of realism possible using motion capture data or accurate full-body simulation, while the multitudes secondary details to the auxiliary elements (scene, cameras etc.) can be simulated at much lower fidelity.

The classic robot motion problem, also referred to as the Piano Mover's problem, can be stated as the following: given a robot R and a workspace W, find a path from an initial configuration I to a goal configuration G, such that R never collides with any obstacle O_i from a set of obstacles O along the path P, if such a path exists.

The path P is a continuous sequence of positions and orientations of R. Continuous sequences of positions and orientations of R are assimilated with the robot system animation on a virtual scene.

Despite the exciting progress in the field, simulating a dynamical system with many degrees of freedom remains a computational challenge. One of the central components of any control or simulation system for articulated bodies is forward dynamics [5].

Forward dynamics computes the acceleration and the resulting motion of each link, based on the given set of external forces and active joint forces. The known algorithms have a linear-time dependence of the number of degrees of freedom. This permits any object in a scene to behave in a physically-plausible way: they accelerate, recognize collisions, and respond to collisions much like one would expect it to respond.

Several techniques have been proposed for accelerating various types of dynamic simulation. Yet, there exists no known general algorithm for automatic simulation of articulated body dynamics.

3.1. Plausible motion simulation

In [1] Barzel introduced the idea of "plausible" motion, i.e. motion that could happen and look physically plausible to the viewers. For many visual applications or real-time interaction, accurately simulating all the details of the real environment is not necessary [3].

In fact, it is often sufficient to provide effective motion to make the scene appear more realistic, without committing much computational resources.

In an environment with uncertainty, we generally expect a constrained problem to have multiple solutions. It is difficult to know before what solutions are available.

Hence, it is bad to use a solution strategy that seeks a single answer; rather, it prefers a technique that produces many solutions that reflect the range of possible outcomes. While for feature animation a user is expected to choose the one animation they prefer, other applications benefit directly from multiple solutions:

- Computer simulator designers can use different animations each time a simulation is on stage, making it less predictable and potentially more entertaining.
- Training environments can present trainees with multiple physically consistent scenarios that reflect the physics and variety of the real world.

The authors generate multiple animations that satisfy constraints by applying an original algorithm to trial from a randomized model [7]. The algorithm needs the model of the environment, including the sources of uncertainty and the simulator that will generate an animation in the virtual environment. The algorithm described in this paper generates an arbitrarily sequence of animations in which “good” animations are expected to appear.

3.2. Simulation loops

The simulation loops fix robots’ components algorithmically, in order to generate physically plausible motion. Note that not all the pieces are put together into a unified system.

Many related works describes a simulation core or a simulation loop to achieve this [4]. The following are the general steps in a simulation core:

1. *Clear the force accumulators*: Each body or its components maintains a total of all forces on it. At the beginning of each step, we clear the forces from the previous step.
2. *Detect collisions*: Loop over all bodies and determine any contacts or collisions in the scene and prepare them to be resolved.
3. *Compute external forces*: Loop over all external forces, including contact or collision resolution forces, and add them to the force accumulators.
4. *Compute constraint forces*: At this point, each body or component knows the total force acting upon it (it’s in force accumulator). To process the constraints, first add any soft constraints to the accumulators. Finally, compute and apply the hard constraint forces.
5. *Compute derivatives*: Gather derivatives as needed to prepare to update the system.
6. *Integrate and update the state of the simulation*: Use a numerical integration method to update the state of the system by some small time increment.

4. Application to Prototyping

Below the authors discuss a few issues pertaining to the implementation of the simulation-related algorithms available in commercial software packages.

4.1. Implementation

Our system was implemented with DELPHI object-oriented programming language. The authors used in-house library ANIMATION-VIEW for collision detection by generating of the distance fields for surface repulsion constraints.

Platform’ toolbox offers the Delphi functions for the implementation of the virtual system prototypes.

For discrete set $\{t_k\}$ of instants, Delphi system generates an images sequence of the virtual robot system.

4.2. System demonstration

The authors have tested the proposed motion planning system in the following virtual prototyping application: *Assembly Line Planning*. An animation generated from this type of scenario is shown in Figure 1.

The robot arms avoid the moving object to reach a moving part passing on the conveyer belt. In this scenario the aim is to animate all actors to realize the assembly of the automobile. In this example, shown in Figure 1, the robot arms from scene must access a part moving and past it on a conveyer belt. The factory floor contains a transfer structure that is moving over the conveyer belt in the opposite direction to the part's movement. The moving obstruction causes the robots to reactively modify its path to avoid collision.

In this example, constraints can be defined for any aspect of the object's 3D state at any point in time. Initial conditions for the object are specified by constraining its state at the start of the simulation.

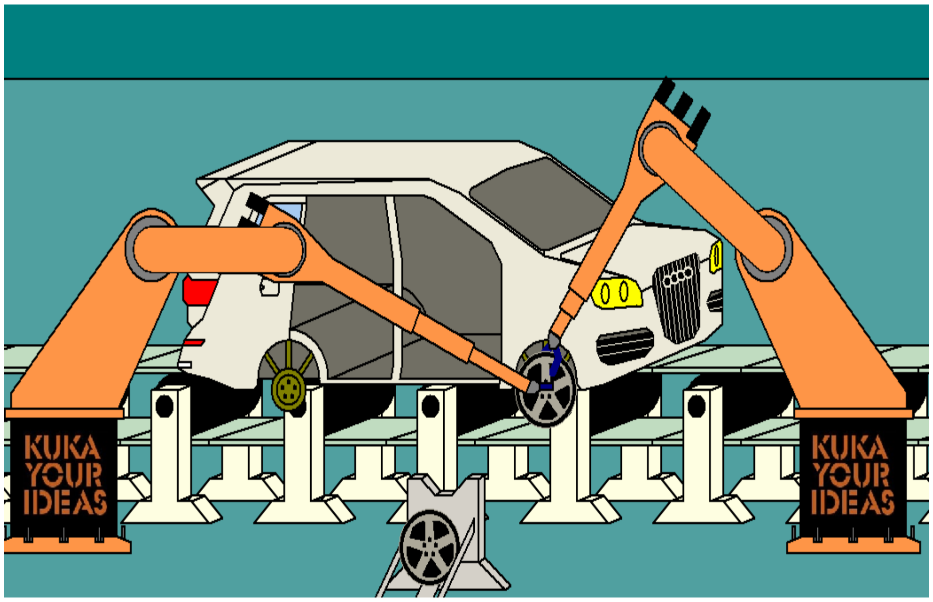

Fig. 1. *Assembly Line Planning Scene*

The simulator, used in this example, simulates all objects at a time to manage the large number of control points required for a specific scene.

5. Conclusions

The authors have presented a novel framework for motion planning in virtual prototyping applications. Thy have reformulate the motion planning problem into a virtual simulation problem where constraints on the robot's motion guide it from its starting configuration to its target, on the virtual scene. These constraints can't impose penetration constraints among objects, the angle limits and connectivity of articulated robot joints. The avoidance of collision, the following of estimated paths, and many other possible relationships between the cooperative robots and objects on the scene, are feasible in the virtual environment.

The models proposed by authors arise naturally in the virtual environment and provide a means of verifying the plausibility of the motion in the real environment. With further work it should be possible to experimentally obtain more accurate robot dynamically models who require finding good animation.

References

1. Barzel, R., et al.: *Plausible Motion Simulation for Computer Graphics Animation*. In: Proceedings of the Eurographics Workshop on Computer Animation and Simulation, December 1996, New York. Springer-Verlag Publisher, p. 183-197.
2. Sugihara, T., Nakamura, Y.: *A Fast Online Gait Planning with Boundary Condition Relaxation for Humanoid Robots.* In: Proceedings of IEEE International Conference on Robotics and Automation, Barcelona, Spain, April 18-22, 2005, p. 305-310.
3. Varley, P.A.C., Martin, R.R.: *Estimating Depth from Line Drawing.* In: Proceedings of the seventh ACM Symposium on Solid Modeling and Applications, Saarbrucken, Germany, June 17-21, 2002, p. 180-191.
4. Venture, G., Ripert, P.J., Khalil, W., Gautier, M., Bodson, P.: *Modeling and Identification of Passenger Car Dynamics Using Robotics Formalism.* In: Journal IEEE Transaction on Intelligent Transportation Systems **7** (2006) No. 3, p. 349-359.
5. Weinstein, R., Teran, J., Fedkiw, R.: *Dynamic Simulation of Articulated Rigid Bodies with Contact and Collision.* In: Journal IEEE Transactions on Visualization and Computer Graphics **12** (2005) No. 3, p. 365-374.
6. Yamane, K., Nakamura, Y.: *Natural Motion Animation through Constraining and Deconstraining at Will.* In: Journal IEEE Transactions on Visualization and Computer Graphics **9** (2003) No. 3, p. 352-360.
7. Zhang, L., Young, J., Kim, Y.J., Manocha, D: *A Hybrid Approach for Complete Motion Planning*. In: Proceedings of IEEE International Conference on Intelligent Robots and Systems, San Diego, California, October 29 - Nov. 02, 2007, p. 7-14.