SIMULATION OF THE ROBOT SYSTEMS FOR VIRTUAL PROTOTYPING IN ASSEMBLY OPERATIONS

Aurel FRATU
Transilvania University of Brasov, Romania

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 author give a brief overview of a general simulation framework, describing the primary tasks which a simulator needs to implement. The robot behavioural simulation in the virtual environment enables us to predict the behaviour of a given real manipulator into real environment.

Keywords: virtual prototype, virtual assembly, estimated path, multi-body systems

1. Introduction

The main specifically properties in the motion control of the robots systems are the complexity of the dynamics and uncertainties, both parametric and dynamic. Parametric uncertainties arise from imprecise knowledge of kinematics parameters and inertia parameters, while dynamic uncertainties arise from joint and link flexibility, actuator dynamics, friction, sensor noise and unknown environment dynamics.

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

The author seeks to deduce a virtual geometry of the objects; as such a 3D geometric realization of a collection of rigid bodies is visible in the drawing. One transforms 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 obstacles and self-collision, or constraints to have the robot follow an estimated path to perform certain tasks in a cycle.

The author demonstrates 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 author need to integrate the system of ordinary differential equations (ODE) describing the dynamics of a robotic mechanical system.

The author uses a model relating the state of the system with its external generalized forces of the form:

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

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 $f(\mathbf{x}, \mathbf{u})$ 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 one take t_0 as the present time, one van predicts from eq. (1) the future states of the system upon integration of the initial-value problem at hand, even if one do not know the complete past history of the system in full detail.

Now, if one regards 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:

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

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 author 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-centre coordinates, and inertia matrix of each of its bodies. To this end, the author first writes 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 [1].

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 author 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, $\mathbf{x}$ and the input or control vector $\mathbf{u}$, 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 $f(\mathbf{x}, \boldsymbol{\tau})$ 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 discrimination of the time variable. If the behaviour 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 $\mathbf{x}(t_k) = \mathbf{x}_k$, 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. Plausible robot's motion simulation

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 [2].

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.

In [3] 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 [4].

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, one generally expects that a constrained problem to have multiple solutions. It is difficult to know before what solutions are available. 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.

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 author generates multiple animations that satisfy constraints by applying an original algorithm to trial from a randomized model. 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.

Generating motions for real or virtual agents, which are coupled to a goal or task, is usually a complicated task. A wide variety of approaches and methodologies have been proposed to attempt the problem [5]. In most cases, these solutions can be viewed as "search" algorithms, where one tray to find a way to the goal through some space representative for the problem. Each search space is tailored to the problem itself, which allows for an extremely wide variety of extensions, applications, and even interpretations of the basic motion planning problem [6]. The solution is to utilize the simulation to relax the requirement of precise control of many degrees of freedom, and instead allow the agents or their parts to move toward the goal through the use of artificial forces acting on the agents.

Control, when needed, is gained through integration of these forces with simplified paths through the search domain [7]. One can shows that an animation framework can overcome many of the limitations of prior approaches and show how it can be used in a variety of application including hyper-redundant robots [8]. For example, consider a team of robotic arms on a manufacturing assembly line whose task is to simultaneously assembly an automobile.

Each arm would need several joints in order to effectively articulate itself to be able to cover any part of the vehicle. The individual arms must move themselves to precise locations along the automobile's body while also avoiding collisions with other arms, the car and other parts of the assembly line.

However, in many cases where the goal is only to have a mass that moves around, motion planning and simulation algorithms could be used to determine this locomotion and to provide goals for each agent. They share a set of rules govern their motion.

Equations for robot dynamics, can model the agent's physical properties, how it interacts with its neighbours, and in which direction it should proceed next. Furthermore, if the vehicle or its parts are moving along the assembly belt, then, their end-effectors must accurately move with the item in a prescribed manner. To automate this process, algorithms must be able to quickly determine the sequence, or collectively a path, of joint angles that each arm must follow to both reach the piece. Finally, it needs a motion controller to execute that sequences.

4. Application to prototyping

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

The algorithm was implemented with DELPHI object-oriented programming language. The author 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.

The author has tested the proposed motion planning system for a robotic assembly line. An animation generated from this type of scenario is shown in Figure 1.

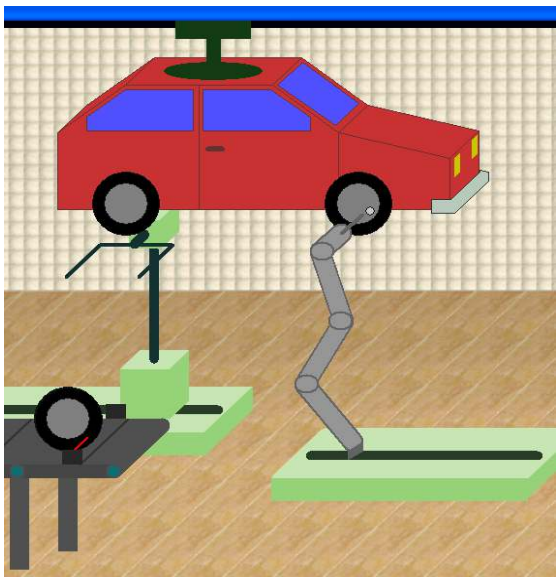

Figure 1. Car manufacturing plant - robotic assembly line

With respect to dynamic obstacles, fast moving obstacles may still collide with a robot since the robots may not always have enough time to react. A velocity-bias also helps with this situation, by essentially projecting the moving obstacle forward in time to a position which a robot can reason about. This situation can also usually be overcome by placing velocity-limits on the obstacles or otherwise correcting the local avoidance parameters.

Regarding parameters, many of the simulation components require correction of parameters to have a realistic and stable simulation. Some algorithms exist which can auto-time some of these parameters, but to the best of our knowledge no approach does this well in a generic manner.

5. Conclusions

The author has reformulated 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. Virtual reality developers can populate their environments with many agents each of which have their own goal, task, or behaviours.

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. An animator could solve this task by carefully moving each agent through the scene.

An algorithm can precisely control all aspects of the situation, or it can employ a predetermined set of laws or rules which influence the motion and behaviour. It is known that there are a wide variety of technical challenges at both ends of this spectrum, resulting in consideration attention from the related research communities. With precise control, it has been shown that the time required finding a solution grows intractably as the complexity of the planning problem, the underlying robot, or the scene increases.

The models proposed by author 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. Yamane, K., Nakamura, Y. (2003) *Natural motion animation through constraining and de-constraining at will.* Journal IEEE Transactions on Visualization and Computer Graphics, DOI: 10.1109/TVCG.2003.1207443, vol. 9, no. 3, p. 352-360
2. Weinstein, R., Teran, J., Fedkiw, R. (2005) *Dynamic simulation of articulated rigid bodies with contact and collision.* Journal IEEE Transactions on Visualization and Computer Graphics, **ISSN:** 1077-2626, vol. 12, no. 3, p. 365-374
3. Barzel, R., et al. (1996) *Plausible motion simulation for computer graphics animation*. Proceedings of the Euro-graphics Workshop on Computer Animation and Simulation, Springer-Verlag, ISBN: 3-211-82885-0, New York, December 1996, p. 183-197
4. Venture, G., Ripert, P.J., Khalil, W., Gautier, M., Bodson, P.: *Modeling and identification of passenger car dynamics using robotics formalism.* Journal IEEE Trans. on Intelligent Transportation Systems, DOI: 10.1109/TITS. 2006.880620, vol. 7, no. 3, p. 349-359
5. Moll, M., Kavraki, L.: *Path planning for variable resolution minimal energy curves of constant length*. Proceedings of International Conference on Robotics and Automation, DOI: 10.1109/ROBOT.2005.1570428, vol. 7, p. 2130-2135
6. Rodrigues, L., How, J. (2003) *Observer-based control of piecewise-affine systems.* International Journal of Control, DOI: 10.1080/0020717031000091432, vol. 76, no. 5, p. 459-477
7. Laumond, J.P. (1998) *Robot Motion Planning and Control*. Springer, ISBN 3-540-76219-1
8. LaValle, S.M. (2006) *Planning Algorithms*. Available for downloading at http://planning.cs.uiuc.edu/, Published by Cambridge University Press

Received in November 2012