Mechanism and Machine Theory 53 (2012) 86–98

Contents lists available at SciVerse ScienceDirect

Mechanism and Machine Theory

journal homepage: www.elsevier.com/locate/mechmt

An efficient parallel dynamics algorithm for simulation of large articulated robotic systems

Kishor D. Bhalerao [a,*], James Critchley [b], Kurt Anderson [c]

[a] Mechanical Engineering, The University of Melbourne, Melbourne 3010, Australia
[b] Continental Automotive, One Continental Drive, Auburn Hills, MI 48326, USA
[c] Mechanical Engineering, Rensselaer Polytechnic Institute, 110 8th street, Troy, NY 12180, USA

ARTICLE INFO

Article history:
Received 9 November 2011
Received in revised form 2 March 2012
Accepted 5 March 2012
Available online 23 March 2012

Keywords:
Parallel algorithm
Robot dynamics
Parallel kinematics
Multibody systems
Divide-and-Conquer

ABSTRACT

This paper presents a new parallel computer algorithm for the simulation of large articulated robotic systems. The method employs the *Divide and Conquer Algorithm* (DCA) multibody methodology and achieves significant increases in speed by using a variation of the *Articulated Body Algorithm* (ABA) to efficiently construct the DCA subsystems. The method outperforms contemporary parallel dynamics algorithms for a serial topology and degenerates to the ABA in the absence of parallel resources. The implementation and comparison with the ABA demonstrate useful speed increases with as few as two parallel processors and speed ups by a factor of 2.25 in the presence of four processors for serial manipulators. The paper also addresses the practical limitations on obtainable speedups due to the interprocessor communication costs and its consequences on choosing optimal DCA subsystems.

1. Introduction

Developing efficient dynamics algorithms has long been the subject of robotics research with the recent efforts being focused on methods to exploit multi-core architectures. There is a lot to consider when choosing a parallel computer implementation to solve a dynamics problem. Ultimately, the end user is interested in speed and resource consumption. However, scalability, adaptability, and portability are also important factors when authoring optimized software.

From a resource perspective (CPU cycles), the most efficient algorithms are the sequential linear complexity $O(n)$ propagation methods [1–3] (where n is the number of degrees of freedom). The efficiency and the relative ease of implementation have resulted in the $O(n)$ methods finding applications in a wide range of areas beyond the traditional domain of robotics [4,5].

This family of $O(n)$ methods performs recursive propagations from body to body and therefore is highly sequential in the computation of serial topologies. This means they have little native concurrency and are not amenable to general purpose parallel computer implementations. However, the independent branches of articulated systems offer significant concurrency which is trivially exploited in parallel computer methods. Ref. [6] describes an early effort deploying this type of basic $O(n)$ parallelism to a real-time vehicle simulation.

In 1995 Fijany et al. introduced the *Constraint Force Algorithm* (CFA) [7] to address the lack of efficient concurrency in chain systems. While other researchers where deploying parallel resources to speed-up traditional $O(n^3)$ solutions, the authors demonstrated that only optimal order resource algorithms would provide scalable benefits and merit general purpose implementation. The CFA was the first such method achieving $O(log(n))$ time complexity with $O(n)$ parallel processors for serial chains (the most difficult case). The CFA relies heavily on the block diagonal nature of the multibody chain system's linear algebra problem to achieve logarithmic binary tree type concurrency. The notion of block diagonal systems allowed the CFA to later be generalized to accommodate all system

* Corresponding author. Tel.: +61 383446720; fax: +61 383444290.
E-mail addresses: kishorb@unimelb.edu.au (K.D. Bhalerao), James.Critchley@continental-corporation.com (J. Critchley), anderk5@rpi.edu (K. Anderson).

0094-114X/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mechmachtheory.2012.03.001

topologies [8] but the general algorithm could only accommodate *short* branches and loops before degrading concurrency and losing logarithmic complexity.

To address the lack of general system solutions in logarithmic complexity algorithms, Anderson and Duan developed the *Hybrid Direct Iterative Algorithm* (HDIA) [9,10]. The method partitions large multibody systems into subsystems which run the efficient $O(n)$ solution with the addition of unknown constraint forces which are solved in exactly the same manner as the common $O(n)$ kinematic loop solution. However, the method requires the solution of a global constraint equation which scales with desired concurrency. Rather than accept an $O(n_c^3)$ performance (where n_c represents the number of constraints which have been "cut" in order to achieve the desired level of parallelism), a parallel iterative pre-conditioned conjugate gradient method was applied. This demands an iterative solution for larger problems and the method does not achieve strictly logarithmic performance.

Featherstone's *Divide and Conquer Algorithm* (DCA) [11,12] was the first algorithm which provides a time optimal $O(log(n))$ speed per integration time step with a resource optimal $O(n)$ processors for a very broad class of problems. The DCA is approximately 4 times slower than the fastest sequential $O(n)$ algorithm. This implies that for a theoretically optimal parallel implementation, the DCA requires 4 processors to break even with the $O(n)$ approach. The DCA is very popular and has been extended in numerous ways to include the *Assembly Disassembly Algorithm* (ADA) [13] and the *Orthogonal Complement Based Divide and Conquer Algorithm* (ODCA) [14]. These two methods perform similar operations as that of the DCA and as such they have comparable computational speed [15].

In recognition of the disconnect between algorithms of low computational order and those which could provide useful results (for practical systems and common computers), Critchley and Anderson developed two logarithmic complexity methods which outperform the fastest sequential algorithms with just two parallel processors. These are *Recursive Coordinate Reduction Parallelism* (RCRP) [16] and *Efficient DCA* (DCAe) [17]. The RCRP is based on a recursive solution for loop closure constraints and requires a complicated accounting system to avoid formulation singularities and as such might never be used for a general dynamics engine. The DCAe on the other hand uses an $O(n)$ algorithm to more efficiently construct DCA subsystems at the requisite level of concurrency. The DCAe requires one propagation sweep per handle to construct the DCA subsystems and achieves an improvement over the $O(n)$ algorithm by a factor of 1.20 and 1.80 for serial chains with 2 and 4 processors, respectively [18] (and continues to scale in this manner).

The method [19] presented in this paper derives the idea of using an $O(n)$ algorithm to construct the DCA subsystems from the DCAe. The method applies the *Articulated Body Algorithm* (ABA) to calculate the handle equations of a DCA subsystem and as such is referred to as the DCA–ABA. Unlike DCAe, excess propagation sweeps are avoided. Additionally, any existing general purpose sequential dynamics engines driven by the ABA (or any of the $O(n)$ methods) can be readily modified to use the DCA–ABA.

To support the development of this new algorithm involving the combination of the DCA and ABA algorithms, both methods are briefly reviewed in Section 2. Section 3 then derives and describes the DCA–ABA and offers additional insight into applying the method. A discussion of the parallel implementation and the associated numerical and parallel timing results follows in Section 4. Finally, the work is summarized with conclusions in Section 5.

2. Underlying dynamics algorithms

The method presented in this paper relies on the DCA and the ABA which are briefly reviewed in this section.

2.1. Divide-and-Conquer-Algorithm

Here, the details of the DCA which are relevant to the current formulation are reproduced. Interested readers may refer to [11–14] for a more detailed discussion. The DCA consists of two process *assembly* and *disassembly* which must be performed twice. The first *assembly–disassembly* sweep is for the system kinematics while the second is for the system kinetics. This section discusses the parallel kinetics sweep while the parallel kinematics sweep is discussed in Section 3.3.1.

The *assembly* operation between two kinematically connected subsystems involves expressing the dynamics of the *assembled* subsystem in terms of spatial quantities associated with the mutually exclusive kinematic joints or handles. On the other hand, the spatial quantities and the time-derivatives of the generalized speeds associated with the handles across the connecting joint are computed during the *disassembly* operation.

The *assembly–disassembly* process is described in Fig. 1. Bodies k and $k+1$ are the *leaf* nodes and form a part of a larger articulated system. H_1^k and H_2^k are the two handles associated with body k. J^k denotes the joint connecting bodies k and $k+1$.

Let $\mathcal{A}_i^k = \left[\left(\boldsymbol{\alpha}^k\right)^T \left(\boldsymbol{a}_i^k\right)^T \right]^T$ and $\mathcal{F}_{ic}^k = \left[\left(\boldsymbol{\tau}_{ic}^k\right)^T \left(\boldsymbol{f}_{ic}^k\right)^T \right]^T$ denote the spatial acceleration of handle H_i^k and the spatial constraint force acting on it, respectively. Unless mentioned otherwise, all kinematic quantities are assumed to be relative to the inertial frame. The two-handle equations of motion of body k can be written as

$$\mathcal{A}_1^k = \zeta_{11}^k \mathcal{F}_{1c}^k + \zeta_{12}^k \mathcal{F}_{2c}^k + \zeta_{13}^k, \tag{1a}$$

$$\mathcal{A}_2^k = \zeta_{21}^k \mathcal{F}_{1c}^k + \zeta_{22}^k \mathcal{F}_{2c}^k + \zeta_{23}^k. \tag{1b}$$

In Eq. (1), the inverse of the spatial mass matrix of body k is embedded in the terms ζ_{ij}^k. The known external forces acting on body k along with the gyroscopic and Coriolis terms are absorbed in ζ_{13}^k and ζ_{23}^k. A similar expression can be written for each body