Available online at www.sciencedirect.com
ScienceDirect
Computers in Industry 58 (2007) 255–264
COMPUTERS IN INDUSTRY
www.elsevier.com/locate/compind

Application of genetic programming to the calibration of industrial robots

J.U. Dolinsky [a], I.D. Jenkinson [b,*], G.J. Colquhoun [b]

[a] *Codeplay Ltd., Edinburgh, UK*
[b] *School of Engineering, Liverpool John Moores University, Byrom Street, Liverpool L3 3AF, UK*

Received 6 March 2004; accepted 23 June 2006
Available online 8 September 2006

Abstract

Robot calibration is a widely studied area for which a variety of solutions have been generated. Most of the methods proposed address the calibration problem by establishing a model structure followed by indirect, often ill-conditioned numeric parameter identification. This paper introduces a new inverse static kinematic calibration technique based on genetic programming, which is used to establish and identify model structure and parameters. The technique has the potential to identify the true calibration model avoiding the problems of conventional methods. The fundamentals of this approach are described and experimental results provided.

Keywords: Inverse static kinematic calibration; Distal supervised learning; Co-evolution

1. Introduction

The range of applications of industrial robots has expanded in recent years in part due to advances in programming capabilities resulting from the development of offline-programming (OLP) systems [11,7,5,1], which enable program development to take place in a virtual environment. However, the critical factor in the application of OLP systems is the accuracy with which they are able to model the physical robot. Deviations between the idealized simulation model and the real world cause the OLP system to generate robot poses with large positional errors. Any successful off-line programming procedure must therefore include a method of compensating for the errors between the simulation and the actual robot.

Robot calibration techniques [4,24,28] are designed to improve the software model of the robot so that it is able to more closely represent the behaviour of the actual robot. These techniques can be classified into two types either static or dynamic calibration. This paper addresses the problem of static [10] calibration, more specifically inverse static calibration. The aim is to improve the kinematic model used to relate the position of the robot end-effector with the joint sensor readings so that it more accurately represents its actual position.

Established methods of robot calibration are based on classical mathematical regression techniques. They typically fit a non-linear, parametric regression model, which is pre-specified by the user to a measured data set. The approach adopted in this work is radically different. The model relating system input to output is not pre-specified but developed in symbolic form using the method of Genetic programming [20,21]. The main advantage of the technique is that it is not limited to a pre-defined model structure and therefore has the potential to produce a complete and more accurate calibration model than conventional methods.

1.1. Conventional modeling techniques

The largest body of research into static robot calibration is based on parametric models, which are designed to represent the true relationship between joint configurations and end-effector poses. Most work in this field has been conducted on forward calibration methods [8,12,14,18] where calibration is applied to the forward kinematic model $\mathbf{P} = \mathbf{f}(\theta, \phi)$. This computes the end-effector pose $\mathbf{P}$ from the joint configuration θ using the equations of the model $\mathbf{f}$, which depend on parameter vectorϕ to be calibrated.

The calibration process is typically carried out using following steps:

* Corresponding author. Tel.: +44 151 231 2011; fax: +44 151 207 3460.
E-mail address: i.d.jenkinson@livjm.ac.uk (I.D. Jenkinson).

0166-3615/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2006.06.003

1. A suitable model is developed usually based on prior engineering knowledge, which provides a model structure and nominal parameter values.
2. A set of end-effector locations is created and their actual position measured.
3. The model parameters are identified based on the measurements using a numerical method based on a least squares fit.
4. The identified model is implemented.

The important issues in the development of accurate parametric based robot calibration models are those of *completeness* and *proportionality* [10] of the kinematic model, and the inclusion of non-geometric effects. A kinematic model is defined as proportional if small changes in the physical geometry can be represented by correspondingly small changes of related model parameters. In order for a model to be complete it must be able to sufficiently describe all spatial geometries of successive joints in the kinematic chain. Models based for example on the Denavit and Hartenberg transformations lack completeness and proportionality, which leads to instability when attempting parameter identification using non-linear optimisation methods. If the model contains insufficient parameters to accurately describe the manipulator's geometry, the parameters then account for both modelled and un-modelled effects. Hence, there is no proper relationship between physical and model parameters. An identification algorithm might be able to find optimal parameter values, but these values are optimised for the particular incomplete model and will not reflect the physical properties of the robot.

Variations in the geometric parameters such as link length, joint orientation, encoder offsets, usually make the largest contribution to positional error and are assumed to be time-invariant which is convenient for setting up a compensation scheme to improve positional accuracy. Non-geometric effects [30] such as joint eccentricity and compliance, static deflection, and thermal effects make smaller but significant contributions to error. However, their effect is difficult to model parametrically and often depends on the manipulator pose and payload.

1.2. Alternative modeling techniques

The difficulty in attempting to model both geometric and non-geometric errors has led to a variety of alternative techniques in which the positional error is approximated using a non-parametric model. Functional approximation theory provides a variety of approximation models and methods are based on uni- and multivariate polynomials [29], splines, Bezier curves, wavelets, Fourier series, artificial neural networks [16,31]. An approximation model is chosen according to the characteristics of the data to be approximated in a certain interval and finally fitted or trained to this data. However, since these approximation models are non-parametric there is no relationship between model parameters and physical properties of the data to be approximated. Hence, they are only valid within the interval in which they have been trained and are weak or incapable of extrapolating or generalizing beyond these interval boundaries. However, for most practical applications they are adequate.

2. Robot error model identification using genetic programming

Genetic programming involves the evolutionary *synthesis* of computer programs from primitive entities. The results are evolved symbolic computer programs capable of solving a problem. The main principles of genetic programming and its terminology (based on the LISP programming language) were developed by Koza [20]. The technique has been used for a range of applications, which include structured process modeling [26], parameter identification in non-linear systems [23], machine learning [25] and artificial intelligence [13,20,21], but its application in the field of robot calibration is believed by the authors to be entirely new.

2.1. The structures undergoing adaptation

Genetic algorithms are designed to modify or adapt solutions encoded usually as string-like and often fixed-length structures. Those strings may correspond, for example, to numerical values of a particular variable or a set of variables. In genetic programming the individual structures, which undergo adaptation, are hierarchically structured computer programs. Thus, since it is possible to write a computer program to solve any problem, the programs evolved have the capacity to span an infinitely large search space. The evolved programs, termed s-expressions by Koza, represent rooted labeled trees made up of terminal and non-terminal nodes. The terminal and non-terminal sets are defined by the problem and are the only resources a genetic programming algorithm uses to compose programs. The terminal set typically contains constants, variables or other primitive problem domain dependent entities, e.g. functions or commands without arguments. With symbolic regression [19], which is used in this paper, the terminal set may also contain an ephemeral random constant $\Re$, which returns a random number in a specified range every time the terminal is chosen. The non-terminal set may contain arithmetic; trigonometric, Boolean functions, conditional operators, and functions that depend on the problem domain, e.g. motion statements in robotics.

It is important for both sets to have all the components necessary to induce a program that solves the problem. In addition, all nodes, in particular function nodes, are required to adhere to the *closure* property; so that the evaluation result of any node is always well defined and valid for any combination of arguments [20]. Since genetic programming is typically unconstrained in the way in which programs are constructed (except that, for example, a maximum tree depth is defined), possible invalid operations need to be prevented by redefining all operators that may return undefined results in certain situations. For example when using the division operator the protected division $a\%b$ must be introduced to prevent invalid results arising from a possible division by zero. Similarly protected versions for square root $\mathrm{SQRTp}(x)$ and logarithm $\log_p(x)$ may be required.