Robotics and Computer-Integrated Manufacturing 26 (2010) 156–161

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

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

Open architecture dynamic manipulator design philosophy (DMD)

Syed Hassan [a,b,*], Naveed Anwer [b], Zafar Khattak [b], Jungwon Yoon [a]

[a] School of Mechanical and Aerospace Engineering and ReCAPT, Gyeongsang National University, Jinju, Korea
[b] Computer Science and IT Department, University of Gujrat, Gujrat, Pakistan

ARTICLE INFO

Article history:
Received 15 September 2008
Received in revised form
5 July 2009
Accepted 8 July 2009

Keywords:
Manipulator
Open architecture
Robotics
Dynamic controller
Design philosophy

ABSTRACT

This paper presents a generic and universal architecture design for robotic manipulators. A flexible approach is taken to develop the design philosophy throughout, resulting in a hardware architecture that is portable, can be integrated and enables the implementation of advanced control methods. A software kernel of management, supervision and control was developed in order to obtain an easy user interface to the robotic researcher. The application of many such controls has, traditionally, often been severely restricted in partial commercial robotic systems because of limitations associated with their controllers; rather than the arms themselves.

1. Introduction

Industrial robots are currently employed in a large number of applications, and are available with a wide range of configurations, drive systems, physical sizes and payloads. However, despite the perceived wide spread deployment of robots, recent surveys indicate that the number in service throughout the world are much less than predicted twenty, or even then, years ago.

In contrast, much academic research has been undertaken in recent years aimed at improving the performance of robots using a number of advanced techniques. They have included model based techniques for predictive and adaptive control, force and hybrid force position control schemes, and attempts to introduce intelligent control methods, especially using ANNs and fuzzy-logic based control Linkens and Nyoungesa [2].

Whilst varying degrees of success have been demonstrated, the application of many advanced methods has often been severely restricted in practical commercial robotic systems because of limitations associated with their controllers, rather than the arms themselves Kozlowski [3].

In order for the robotic device to perform a given task it is necessary to control it that is basically, to have the capabilities of, first, sending specific orders to the manipulator, in terms of positions or velocities of its final effector, and second, sensing the real obtained position or velocity. Other sensing capabilities may be important depending on the application, such as force feedback. So, in order to implement a given control algorithm for the manipulator to solve a task, it is necessary to control its basic functionalities (moving and sensing) which implies the development of a software platform.

2. Existing open architecture controller

The impact that 'open' system has had on the computer science culture has been remarkable.

From a general computing point of view, the term 'open architecture' has been attributed the following definition:

> An architecture whose specifications are public; this includes officially approved standards as well as privately designed architectures whose specifications are made public by the designers. The opposite of open is closed or proprietary

This definition is applicable to the general computer science community as a whole, but the phrase 'open architecture controller ', which has been partly plagiarized from this definition since the late 1980s Zeng G et al. [4], requires a slightly different definition.

The need for open architecture controllers has arisen because of a pressing need for more advanced flexible manufacturing system (FMS) in factory environments Proctor and Albus [21]. The NGC–SOSAS establishes a virtual controller infrastructure that integrates software-based 'services' under the aegis of a master controller, and is designed to provide interoperability, portability, scalability and interchangeability to a controller Anderson [1]. In Europe, the OSACA (open system architecture for control within automation) project aims to define hardware independent reference architecture for a range of industrial controllers Pritschow [15]. The OSEC project in Japan following the similar goals of open philosophy developed the

* Corresponding author.
E-mail address: wahab_kool@yahoo.com (S. Hassan).

0736-5845/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.rcim.2009.07.006

seven-tiered reference model the Kasashime et al. [14]. Naturally, each of these specifications and standards defines an 'open architecture controller' with a slightly different emphasis, but there are common themes throughout.

The demand for faster manufacturing processes with tighter quality control standards is pushing the limits of older generation robots Proctor and Albus [21]. Whilst many robot controllers features common elements in their hardware (CPUs, for examples the Intel 8088), adding performance upgrades is limited or even impossible. Even the most well adapted proprietary controllers are falling behind when compared with other devices in a FMS. To add new functionality in the controller without restoring or going for reverse engineering, the only option is using a communication port that will allow real- time path modification of the robot's trajectory. The slow communication rates manufacturers usually provide with these interfaces renders it impossible to effectively make compensations in a sensor based control loop with typical sampling requirements up to 1 kHz Proctor and Albus [21]. This limitation has negatively affected several attempts to implement sensor based control with a proprietary controller Bicker et al. [16].

The ESA's CDM utilizes a functional reference model (FRM) for the activities of robot control systems. The top hierarchy, the MISSION layer, attempts to describe the activities that the robot is responsible for very abstract terms, for example SERVICE a satellite, and REPAIR a platform. The TASK layer decomposes the high level activities into tasks, which are defined as the highest level of activity that can be performed on a single subject/object (OPEN a door, WELD a seam).

The NEXUS open software system has a similar methodology, whereby modules are defined by the services they provide Fernández-Madrigal and González [17]. Data and event flows between different modules are handled by an Internal Communications Manager (ICM) and an Architecture Information Manager (AIM). Zhou and deSilva have retrofitted a PUMA 560 with an open-structure controller Zhou and deSilva [20]. The host computer is an intel 80486 processor with AT system bus. This choice of OS is not specified, but the real-time support provides a GUI. Brambones and Etxebarria have reverse engineered a controller for the Tecquipment Ltd. MA2000 laboratory manipulator Barambones and Etxebarria [18]. This is a six-DOF manipulator actuated by DC motors with potentiometer position feedback. The existing controllers surveyed had a drawback with controller that when operates in free motion, the system is controlled in open loop. Precompiled code cannot be modified and to link more libraries the software requires updating which again push them to closed end besides the controller is still an open architecture.

In general, robot controllers can be broadly classified into three different types Fu et al. [6]:

(a) *Proprietary*: the controller structure is effectively closed. Integration of external or new hardware (including sensors) is either very difficult or impossible.
(b) *Hybrid*: the majority of the system is closed (control laws, etc.) but some aspects of the system remain open. It is possible to add new devices such as sensors.
(c) *Open*: the controller design is completely available to be changed or modified by a user. The hardware and software structures can be changed such that all elements (servo laws, sensors, GUIs) can be modified without difficulty.

3. Enabling technologies

When considering the implementation of an open architecture controller based on the architectural specifications and models discussed in Section 2, a particular hardware architectural must be committed to follow open standards and open source code. The high level, somewhat abstract architectural specification documents have lead developers of prototype systems to choose the following enabling technologies Fu et al. [6]; James and Graham [7]:

A. *Standard operating system* (OS) like DOS or Windows.
B. *Non-proprietary hardware* such as PC's or SUN workstations.
C. *Standard bus systems* such as PCI or VME.
D. *Use of standard control languages* such as C or C++ or Java.

4. OS for open controllers

The operating system provides a software interface to enable the user to run application program and performs tasks such as port I/O, updating the screen display and communicating with peripheral device. In general, the tasks that an open architecture controller has to manage can be split into two different categories:

A. *Direct machine control*: this encompasses drive interfacing, signal conditioning, trajectory generation, servo-control (or other joint control algorithms), sensor/transducer interfacing and coordinating asynchronous events.
B. *Non-machine control*: this encompasses tasks such as interpreting instruction sequences (CNC codes/robot program files), higher level communications to other systems and providing user interfaces.

We can also classify these two sets of task into real-time and non-real-time. The definition of real-time, which relates to the computing control systems, is given accurately by Microsoft Corporation [8]:

> A real time system is one in which the correctness of the computation not only depends upon the logical correctness of the computation, but also upon the time at which the result is produced. If the timing constraints of the system are not met, system failure is said to have occurred.

5. Motion controllers

From the wide variety of industrial motor control equipment available a servo controller of this nature requires a minimum, for each axis under control, 0–10 VDC analogue output channel and an encoder input channel Perry and Wolf [9]. A wide variety of products exist, each with differing feature and option.

The PMAC (programmable multi-axis controller) card manufactured by Delta Tau Systems is a PC expansion card for most common local bus systems (PCI, VME, etc.) and is equipped with onboard A/D and D/A converters, digital I/O, encoder inputs and PLC emulation. Many accessories are available to provide further expansion. It has a Motorola DSP 56001 Digital Signal Processor running at a clock speed of 20 MHz (standard), and up to 60 MHz (enhanced). It is highly flexible system that allows many advanced custom features to be incorporated Lee and Lee [13]. A diverse array of options and control features is available as standard, making the card highly suited to research, robotics and machine tool applications.

Communications between the host processor and the PMAC servo controller take place via the common system bus, using ASCII character format. This is generally the case for these types of controller. The interpretation of the meaning of the