USING ACTUAL INDUSTRIAL ROBOT MANIPULATORS WITH CONSTRUCTION TASKS

J. Norberto Pires

Mechanical Engineering Department, University of Coimbra

3030 Coimbra, Portugal, norberto@robotics.dem.uc.pt

Abstract

Actual industrial robot manipulators are sophisticated machines that work essentially as position and motion controllers. They have sufficiently powerful programming environments and good communication devices that, considering also their ability to perform human-like tasks, make them a typical case of flexible manufacturing equipment. Force control can also be used if no more than passive and/or indirect force control is required. This is roughly the actual state-of-the-art of industrial robot manipulators. Is this useful or interesting for construction tasks? In this paper we review the main characteristics of actual manipulators, in a way to show that technically the actual state-of-the-art is sufficient to cope with the requirements of many of the construction tasks. Beside of that, and using a typical industrial manipulator, we present a software interface that can be used to program, monitor and control those tasks. The main applications of the presented software are with off-site construction tasks, although on-site tasks can also benefit from using it. Off-site tasks include manufacturing prefabricated panels, adding ceramic covers to other prefabricated panels, polishing, painting, window assembly, etc.

Keywords: Industrial Robotics, Robots for construction, software, man-machine interfaces.

1. Introduction

Robot Technology evolved enormously in the last 25 years [1,2], although only a few selected technologies are available on commercial industrial robots. Many other technologies, currently common on research and development environments, didn't reach the market yet mainly due to the conservative nature of the robot industry. Things like force control, visual servoing, flexible robots, advanced programming leading to intelligent and autonomous robots, etc, are still not available for the general user. Actual robot state of the art, if we take commercially available robots, present us with excellent position controlled machines, with interesting programming environments and several interfaces with other machines (IO, serial communication, fieldbuses, TCP/IP connections, etc). How did we get this far, and what should be done to go further is part of the subject of this paper. Also, what is now possible with commercial robots and what can be expected in the next few years, considering in particular civil construction tasks will be considered. Robotics is not a science of your century or of the near future, as is commonly mentioned. In fact, this idea of designing and building robots (capable of human like tasks and obedients) is not new and was part of the thoughts of many of the great thinkers of our common history. Briefly, the first works on robotics may be traced back until 270 BC, in the ancient Greece, to the water clocks with mobile figures designed by the Civil Engineer Ctesibius [1]. His work was followed by Phylo of Byzantium (author of the marvellous book "Mechanical Collection", 200 BC), Hero of Alexandria (85 BC) and Marcus Vitruvius (25 BC) [1]. Several hundred years later, the Arabians documented (the three Banu Musa working for the Kalifa of Baghdad, 786-833 AC) and developed (Badías-Zaman Isma'Il bin ar-Razzaz al-Jazari in the book "The science of the Ingenious Devices", 1150-1220 AC) the Greek designs to be used on their own creations. Leonardo Da Vinci also spent some time on robotics, when he was working for the Sforza family. By the same time he painted "The last supper", he was also involved with building the "Salle delle Asse" of the Sforza Castle, where he planned to put a human-like robot in the form of a XV century knight [1,3,4]. Somehow, the plans and drawings were never found, although some pages of his famous book "Codex Atlanticus" are missing precisely in the point where it seams that he was preparing the robot project. But Leonard didn't have

at the time the sufficient conditions to develop efficient robots: namely a permanent power source and the possibility to build parts with high precision. Nicola Tesla did another outstanding contribution to robotics, in the turn to our century. He was thinking about automatons and how he could command them or "embody" intelligence on them. At the time, there was a German scientist (Hertz) claiming that an electromagnetic excitation generates radiation of the same type that can be detected far from the excitation. Tesla thought about using this to command an automaton: the term "tele-automatics" appeared. In its own words [5]:

"… But this element I could easily embody in it by conveying to it my own intelligence, my own understanding. So this invention was evolved, and so a new art came into existence, for which the name "teleautomatics" has been suggested, which means the art of controlling movements and operations of distant automatons."

Modern robotics started in the late fifties with the Goertz Master-Slave robots [6], designed to handle dangerous materials. After that the evolution was very fast, mainly after around 1970, when the first industrial robots appeared. In this paper we'll focus mainly on industrial robot manipulators since those are the ones presenting good potentialities for construction applications. Also, they've been used intensively in several industrial applications, which further enhanced their capabilities and flexibility as a way to meet the requirements of today manufacturing platforms. For those types of robots, we'll briefly present in section 2 a brief state of the art. Section 3 presents a software environment designed for industrial and automation equipment. The idea is to show applicability to the present case of Civil Engineering tasks. Finally, some conclusions are drawn in section 4.

2. State of the art

Industrial robot manipulators are currently position/motion-controlled machines. With them users can define a set of positions, define trajectories and the motion parameters between those positions, and execute them continuously. Basically, this is all that they can do. Robot controllers offer additionally PLC like capabilities to control IO signals (digital and analog), several communication interfaces (profibus, can, Ethernet, serial channels, etc, are common) and a programming or scripting language to access all this resources. Actual robot manipulators main characteristics are resumed in table I.

Table I – Main characteristics of actual robot manipulators

Repeatability	up to 0.03 mm (0.1 mm is common)
Velocity	up to 5 m/s
Acceleration	up to around 25 m/s2
Payload	from 2-3 kg up to 350 kg
Weight/payload	around 30-40
Axis	6
Communications	Profibus, Can, Ethernet and serial channels (RS 232, 485)
IO capabilities	PLC like capabilities to handle digital and analog IO.

In conclusion, actual robot manipulators are excellent motion controllers, with sufficient but somehow limited programming environments and closed controllers (even to the advanced user).

In the near future, robots must evolve to reduce weight leading to flexible robots. Also, some effort should be done to improve actuator efficiency. Intelligent sensors, including data processing, filtering and packaging should also be improved to get more distributed resources in a robot. But the main advances must be done at the system controller level. Robots are still very complex machines to use and program, i.e., although they are the ultimate example of a flexible machine, its flexibility is only barely used. And that is so due to the fact that robot controllers are closed systems, using different types of hardware and operating systems, different programming languages, complex developing processes requiring too many details, very deficient high level programming and workstation connection, etc. Robots will give a "quantum leap" when standardization finally arrives to the robotics industry. The adoption of standards, both on hardware and software, enabling user access to robot controllers and the introduction of new features will certainly accelerate steps further.

One of the required steps is force control [11-14]. When robots interact with parts and surfaces, and the contact forces are important to successfully accomplish the task goal, then the robot needs some efficient way to actively control those contact forces. That also means adding force/torque sensors and/or tactile sensors, which are commonly available from several manufacturers. Visual servoing, using CCD cameras, laser cameras or other visual sensors, is also fundamental for parts handling but also for complex tasks like robotic welding.

This is industrial robotics today. Is it sufficient for construction tasks? In the next section we give an example of a software environment developed at our laboratory to be used with actual industrial robots. By using it, and showing its capabilities, we'll demonstrate that many of them can be used with construction tasks, namely off-site construction tasks.

3. Exploring an Industrial Robot

The key factor about industrial robots is its flexibility, i.e., the possibility to perform different tasks just by reprogramming and retooling. If we take actual market conditions (leading to small batch manufacturing of products with increasing complexity and parts density), it is very easy to understand why robots are so important for manufacturing platforms. They represent, when integrated into Flexible Manufacturing Systems (FMS), the possibility of a fast response to market needs and product enhancement. Nevertheless, if we consider all the possible equipments of a FMS system, with a lot of different robots, control systems, PLC, etc, then we easily conclude that their potential flexibility is only barely used. Any change, even if small, will require a specialist to handle it. And that is generally not easily available or very expensive. If robots are to be used as a general tool with complex tasks into flexible environments (like constructions tasks), then they need to improve their programming environments (leading to distributed and object oriented high level programming), remote access capabilities facilitating integration with other equipments, etc, leading to machines more easier to use by regular operators [9,10,15]. This means, more intelligent robots capable of receiving complex requests from user computers, execute them and delivering results. Also, means connectivity with regular computers in a way to improve human-machine interfaces.

Basically, when we want to use some kind of equipment from a computer we need to write code and define data structures to handle all its functionality. We can then pack the software into libraries, which are not very easy to distribute being language dependant, or build a software control using one of the several standard architectures available (preferably ActiveX or JAVA [9]). Using a software control means implementing methods and data structures that hide from the user all the tricky parts about how to have things done with some equipment, focusing only on using its functions in a easy way. Beside that, those components are easily integrated into new projects built with programming tools that can act as containers of that type of software controls, i.e., they can be added to new projects in a "visual" way. We built software components to handle industrial robots (any from ABB), force/torque sensors (any from JR3 Inc.), a CCD camera (VS710 from Siemens), and other equipment. With them users can build applications exploring their functionality using tools like: any Microsoft Visual Studio Tool, Matlab from Mathworks, LabView from National Instruments, any DDE client tool and any ActiveX container tool [9] (for example, any Microsoft Office Tool can be used to access robot services; we've done that to recollect production information directly into Excel spreadsheets).

In this paper, we choose to demonstrate using Matlab, since it is a well-known package and is basically an interpreted language, which suits our demonstration purposes. The interested reader can find more application details in [9,10], or in our web site http://www.dem.uc.pt/norberto/. Components can be included in Matlab as MEX files. We built a toolbox named MATROBCOM [10] that includes modules for all the equipment mentioned above. One of the modules uses Remote Procedure Calls (RPC) services available from ABB robots, enabling users to control a robot from the PC. In fact this module integrates an ActiveX control built with the same purpose [10]. Table II lists the functions available on that module.

Table II - Functions available in the MATABBS4 module.

Function	Brief Description
open	Opens a communication line with a robot (RPC client)
close	Closes a communication line.
motor_on	Go to Run State
motor_off	Go to Standby State
prog_stop	Stop running program
prog_run	Start loaded program
prog_load	Load named program
prog_del	Delete loaded program
prog_set_mode	Set program mode
prog_get_mode	Read actual program mode
prog_prep	Prepare Program to Run (Program Counter to begin)
pgmstate	Get Program Controller State
ctlstate	Get Controller State
oprstate	Get Operational State
sysstate	Get System State
ctlvers	Get Controller Version
ctlid	Get Controller ID
robpos	Get current robot position
read_xxxx	Read variable of type xxxx (there are calls for each type of variable defined in RAPID)
read_xdata	Read user defined variables
write_xxx	Write variable of type xxxx (there are calls for each type of variable defined in RAPID)
write_xdata	Write user defined variables
digin	Read digital input
digout	Set digital output
anain	Read analog input
anaout	Set analog output

The robot may be connected to the computer using a serial port or preferably an ethernet port, both using TCP/IP protocols (fig.1). If a local area network is available, several users/computers may be connected to the robot at the same time (with a line or channel

open), and then MATABBS4 keeps track of actual open lines/channels. Opening a line means starting a client connection to the RPC servers running on the robot.
Suppose that we have a robot program (written in RAPID [8], a robot programming language from ABB Robotics) running, which is switched by a variable named, let say, 'decision'. The basic structure of the program would be something like (using a C-type definition),

```
while never_end;
        switch decision
          case 1: call routine_1; break;
          case 2: call routine_2; break;
          …
          case n: call routine_n; break;
        end_switch;
end_while;
```

Then if we define complex routines to meet our special needs, it is very easy to write scripts to call a sequence of them [9]. In Matlab, that would be:
>> line = matabbs4 ('open', 'babylon', 'reserved')
If program not yet running,
>> matabbs4 ('program_load', 'flp1:\example.prg', line)
>> matabbs4 ('program_run', line)
Call routine_1,
>> matabbs4 ('write_num', 'decision', 1, line)
Note: When task is complete an RPC call is made to the PC with that information (event calls). We can check that just by reading if the variable "decision" reached its default value.

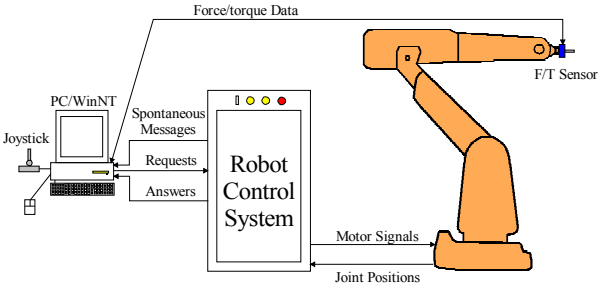


Fig.1 – Robot and sensor connections.

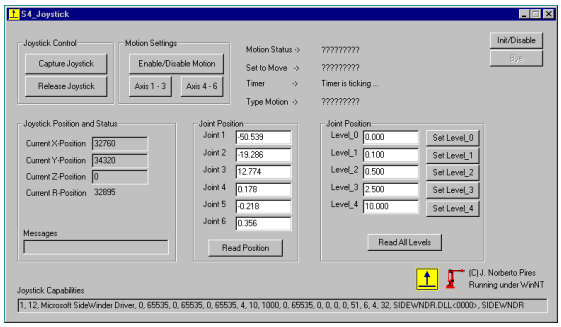
Acting on IO,
>> matabbs4 ('digout', 7, 1, line)
>> matabbs4 ('anaout', 1, 236, line)
Call routine_2,
>> matabbs4 ('write_num', 'decision', 2, line)
For example, suppose that routine_3 moves the robot from actual position to another one defined by a position variable named 'new_pos' of the type robtarget. Routine_1 should then be something like,

```
PROC routine_1
   MoveJ to new_pos;
   decision = -1;
ENDPROC
```

We can then send the robot to some position just by commanding,
>> matabbs4 ('write_robtarget', 'new_pos', new_position[1,:], line)
>> matabbs4 ('write_num', 'decision', 3, line)
The motion parameters (velocity, acceleration and positioning precision) can also be set before issuing the call to routine_3. That was not done just for simplicity.
Now, since the majority of the construction tasks require mainly positioning and IO control, these demonstrations show the possibilities we can have just by being able to control the robot from a PC, where the civil engineer tools are. By tools we mean CAD and simulation tools, where parts and things are designed. This makes the above-mentioned software useful for a vast majority of off-site construction tasks [16]. We've been using this software on several industrial tasks requiring databases and PC software to define the task completely, or that require monitoring of the production site [17]. Tele-operation is also possible, since new positions can be fed to the robot at high rates (a new position can be commanded in ~15 ms). In fact, we built a small C++ application that enable users to use any game joystick to jog the robot from any computer on the network (fig.2). We are now experiencing with force feedback joysticks (Microsoft Sidewinder Force Feedback Pro). That will integrate information from a force/torque sensor mounted on the tip of the robot. Finally, the all collection of tools is presented as ActiveX controls which means that they can be used with web applications, i.e., using the robot from the web as if we were on-site.

4. Conclusion

In this paper, actual state of the art of robot technology was briefly presented and discussed. Needs for future robots were also briefly presented and forecasted. Civil Engineering is considered to be an area where actual robotic technology can be applied with success in the improvement of actual construction processes. Robots are currently accurate and powerful for those types of tasks. It is only a question of integration and adaptation to construction environments. Off-site construction tasks are the ones that can easily benefit from actual robots.
Finally, we demonstrated how to include actual robots into distributed environments, and also how important is to integrate robots into our computer environments. Robots should be used from our normal working tools, and should be simple to do that. That means, high-level software, standardization and integration to standard environments, i.e., object-oriented software, adoption of standard technologies both for hardware and software, and the effort to integrate them into our working tools. We demonstrated that using an object oriented language (ActiveX), one control-engineering tool (Matlab), under a very common operating system (Microsoft win32 operating systems: Windows 98/NT/2000).



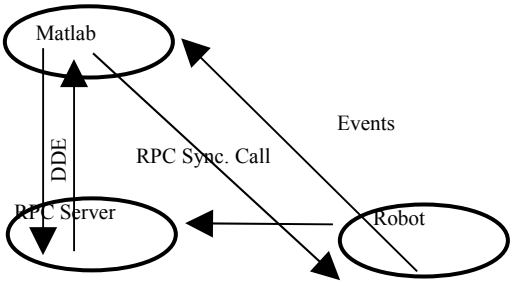


Fig.2 – Joystick demonstration application and matlab interface.

5. References

[1] M. Rosheim, "Robot Evolution: The Development of Anthrobots", New York: John Willey & Sons, 1994.

[2] L. Westerlund, "The extended arm of man: a history of the industrial robot", ISBN: 91 7736 4676-8, Sweden, 2000.

[3] M. Rosheim, "In the Footsteps of Leonardo", IEEE Robotics and Automation Magazine, June 1997.

[4] C. Pedrettii, "Leonardo Architect", New York:Rizzoli International Publications, 1981.

[5] N. Tesla, "My Inventions: Autobiography of Nicola Tesla", Willinston, VT: Hart Brothers, 1983.

[6] R.C. Goertz, "Fundamentals of General Purpose Remote Manipulators", Nucleonics - Vol. 10, Novembro de 1952.

[7] United Nations e International Federation of Robots, "World Industrial Robots 1996: Statistics and Forecasts", New York: ONU, 1996.

[8] ABB Rapid Users Manual, ABB Flexible Automation, 1997.

[9] J.N. Pires, J.M.G. Sá da Costa, "Object-Oriented and Distributed Approach for Programming Robotic Manufacturing Cells", IFAC Journal Robotics and Computer Integrated Manufacturing, Volume 16, Number 1, pp. 29-42, March 2000.

[10] J.N. Pires, "Using Matlab to Interface Industrial Robotic & Automation Equipment" Accepted to IEEE Robotics and Automation Magazine, to appear, (expected September 2000).

[11] JN Pires, "Force Control on Industrial Robotics", Ph.D. Thesis, University of Coimbra, June, 1999.

[12] B. Siciliano, L. Villani, *Robot Force Control*, Kluwer Academic Publishers International Series in Engineering and Computer Science, Boston, MA, 1999.

[13] J. De Schutter, H. Bruyninckx and M. Spong, Force control: a bird's eye, *in* 'Proceedings of IEEE CSS/RAS International Workshop on Control problems in robotics and Automation: Future directions', San Diego, USA, 1997.

[14] R. Volpe and P. Khosla, "A theoretical and Experimental Investigation of Explicit Force Control Strategies for Manipulators", IEEE Transactions on Automatic Control, 1993;38(11):1634-1650.

[15] K. Nilsson, "Industrial Robot Programming", Ph.D. Thesis, Department of Automatic Control, Lund Institute of Technology, May of 1996.

[16] L.F. Penin et all, "Robotized Spraying of Prefabricated Panels", IEEE Robotics and Automation Magazine, pp.18-28, September 1998.

[17] JN Pires, "Programming Industrial Robotic and Automation Equipment", Industrial Robot, An International Journal, MCB University Press, to appear, (expected July 2000).