Real-Time Integration of Industrial Robot with MATLAB

Megha G. Krishnan, Abhilash T. Vijayan, and S. Ashok

Abstract The prevalence of robots in manufacturing industries is growing due to the increasingly automated industrial operations. The research in robotics and automation is well supported by simulation platforms like MATLAB, which provides a scientific tool with an algebraic base and various toolboxes. Dedicated systems can be built after testing the innovative algorithms developed. The growth of robotic solutions becomes sluggish at the system interface phase, which is critical and challenging. This paper discusses a simple and cost-effective way of interfacing MATLAB with an industrial robot, ABB IRB 1200 in real-time by establishing file transfer protocol (FTP) client–server communication using Transmission Control Protocol/Internet Protocol (TCP/IP). The method benefits researchers to understand and explore the possibilities of robot communication for validating any modern control. A positioning experiment is conducted to demonstrate how the robotic system is communicated with the MATLAB.

Keywords Real-time integration · Industrial robot · ABB IRB 1200 robot · MATLAB interface

1 Introduction

Industries benefit from intelligent or decision-making robots in production lines and applications rather than collaborative robots. Numerous research has been carried out to develop smart robots to reproduce human senses by gathering information about the environment and making decisions for themselves. When the results are to

M. G. Krishnan (✉) · S. Ashok
National Institute of Technology, Calicut, India
e-mail: megha_p160007ee@nitc.ac.in

S. Ashok
e-mail: ashoks@nitc.ac.in

A. T. Vijayan
Rajiv Gandhi Institute of Technology, Kottayam, India
e-mail: abhilash@rit.ac.in

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2022

A. Kumar and S. Mozar (eds.), *ICCCE 2021*, Lecture Notes in Electrical Engineering 828, https://doi.org/10.1007/978-981-16-7985-8_13

be implemented in real-time systems, the interface options become critical in determining the system's performance. Since most of the software solutions of robots are proprietary in nature, addition or integration of any third-party software or hardware is not stress-free. In this paper, an effective communication between the control system and the ABB robot is established for the validation of a suggested strategy in a real-time system. Both modelling and simulation are possible for ABB robots with various toolbox developed for MATLAB. The programs can be translated into the corresponding RAPID code using inbuilt functions and the IRC5 controller can receive the RAPID file using a primary FTP client or through RobotStudio for experimental evaluation. The ABB robot controllers are capable of reading external information from sensors through customer I/O devices and influence the program structure in RAPID. Section 3 explains the stepwise procedure for the interface of ABB robot with MATLAB through simple programming.

2 Literature Survey

Numerous platforms are available for the simulation studies of robotic systems, some of which can be extended to hardware-in-loop applications. MATLAB, a widely used software platform for research and teaching purposes in robotics and automation, mainly because of the availability of a collection of toolboxes and specific third-party solution packages [1]. Different manufacturers have dedicated robot controllers and proprietary robot languages; most of them have similar structures. It would require either the knowledge of additional software or paid add-on installations or both for effective communication between the robot controller and the computation software [2]. Effective integration of the robotic system with research software platforms is necessary to implement innovative algorithms developed but presents significant challenges [3]. One of the most common application development frameworks for robots is the Robot Operating System (ROS) [4]. Robotics systems toolbox from Mathworks allows the connection of ROS with MATLAB/Simulink [5, 6]. Various toolboxes/software like Advanced Robotics Control and Operations Software (ARCOS) [7], Interfacing Toolbox for Robotic Arms (ITRA) [2], Mobile robot toolbox [8] etc., were developed to communicate between MATLAB and the pioneer family of robots. A client–server communication using TCP/IP protocol is developed in C/C++ for MATLAB compatible motion control units on the remote side [9], which is not used for industrial robots. The R&A equipment, including industrial robots are accessed from the MATLAB shell based on distributed software architecture [10]. This paper demonstrates a simple and efficient way of integrating an industrial robot with MATLAB in the server computer. This method requires only the basic knowledge of MATLAB, where most of the techniques discussed in the literature require knowledge of software programming languages like C, C++, and Java. Most of the researchers face difficulty while testing and implementing the developed algorithm in a real-time system. This method is advantageous for testing the algorithms in a real-time environment without procuring and mastering add-on packages or supplementary software platforms.

3 Robot MATLAB Communication

The robot system under study includes 6 DOF ABB IRB 1200 robot manipulator with 7 kg payload and 703 mm reach and IRC5 controller [11]. RobotWare is the robot controller software that communicates RAPID, a flexible high-level programming language [12]. The robot manipulator can be positioned by running the MATLAB programs in the personal computer (PC) and RAPID program in the FlexPendant (ABB's Human Machine Interface) in parallel. The controller has numerous Ethernet channels, which can be used at 10 Mbit/s or 100 Mbit/s. The speed of communication is set automatically. The programmer can send or receive data/information between the PC and the robot by establishing TCP/IP communication with network file system access using FTP client and server (see Fig. 1).

To establish secure communication between robot and PC, connect the robot controller and PC through a permanent Ethernet port. Then log on to the controller

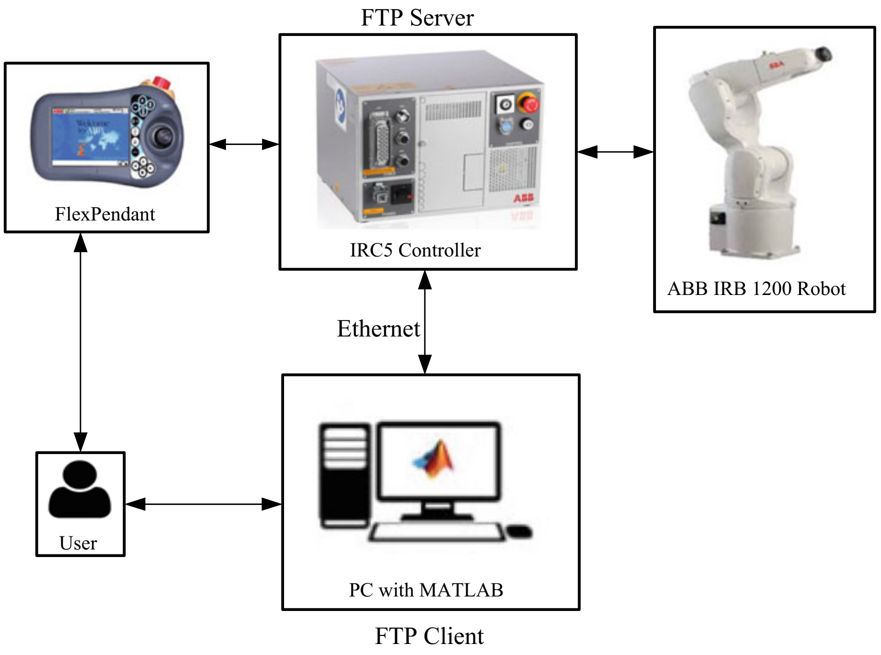

Fig. 1 System framework

using the user authorization system (UAS) in the FlexPendant. UAS can limit available functions to the users. Configure the PC to use Dynamic Host Configuration Protocol (DHCP) to obtain an automatic Internet Protocol (IP) address and open the command prompt window in the PC and renew the IP configuration by executing ipconfig/renew command. Command prompt window displays the current TCP/IP network configuration of the PC. The FTP server on the IRC5 controller will assign an IP configuration for the client PC. As the connection is established, the files with data/information are transferred between the client and the server.

3.1 Creating FTP Object and Files for Data Transfer in MATLAB

MATLAB can be connected to the FTP server by calling ftp function as

```
robotftp = ftp('192.168.125.1'.'username','password');
```

The username and password are to access a particular FTP account on the server, which corresponds to that for logging on the controller. Open files for the position, orientation and configuration data using `fopen` function and keep them in the MATLAB directory for data transfer. Upload these files to the robot controller using `mput` function and then close and delete them from the MATLAB directory using `fclose` and `delete` functions to prevent duplication and replacement of intended data.

3.2 Write and Send Current Pose Data in RAPID

Meanwhile, the robot controller is ready to receive the files uploaded by the PC. The RAPID program code written on the FlexPendant should be running and waiting for these files. The text files for the position, orientation and configuration data are declared as string variables `string1, string2` and `string3` in the `MainModule`. The program code to be executed is written in the procedure called `main`. There can be several procedures and a procedure is declared in the program with `PROC`. Then the existence of file is checked using the following code snippet and the maximum wait time for the robot is set as 240 s.

```
IF object THEN
object:=FALSE;
ELSE
WaitUntil IsFile(string1)\MaxTime:=240;
obect:=FALSE;
ENDIF
```

Table 1 `robtarget` components

S. No.	Reference	Data	Data type	Description
1	Translation	[x, y, z]	pos	Position of the tool in mm
2	Rotation	[q1, q2, q3, q4]	orient	Orientation of the tool in quaternions
3	Robot configuration	[cf1, cf4, cf6, cfx]	confdata	Axis configuration of the robot
4	External axes	[eax_a, eax_b, eax_c, eax_d, eax_e, eax_f]	extjoint	Position of the external axes

Once the files are received at the controller, the current pose of the robot has to be written and send back to the PC for further movement. The pose data in RAPID [1] is defined using a data type `robtarget`, which can be expressed as,

```
CONST robtarget pos1:=[[x, y, z],[q1, q2, q3, q4],[cf1, cf4, cf6,
cfx],[eax_a,eax_b,eax_c,eax_d,eax_e,eax_f]]
```

The robtarget consists of four components and each one is explained in Table 1. The current position of the robot can be read using `CRobT` function, which returns a robtarget value with position, orientation, axes configuration, and external axes position. Open a file from `diskhome` for writing using the open function and set the file pointer to the beginning of the file using `Rewind` instruction. The following code snippet shows how the current position is written on file and keep ready for sending to the PC.

```
Open diskhome\File:=string6, selfposedata\Write;
Rewind selfposedata;
posnow := cur_pose.trans;
Write selfposedata, ""\Pos:=posnow;
Close selfposedata;
```

For convenience, the configuration data `cf1, cf4, cf6` and `cfx` are initially transferred to registers and then write to the configuration data file. When the data file is moved to the MATLAB directory, the robot can either wait for the next instruction or end the procedure depending on the program needs.

3.3 Read Current Pose Data in MATLAB

Once the files uploaded to the controller is written with the current pose of the robot, download those files using `mget` function and load data into an array using `importdata` function. The orientation data in quaternion format is converted into rotational matrix using `quat2rotm` function in Corke's Robotics Toolbox (RTB) [14]. The next pose for robot positioning can be obtained using the programmer's control law written in MATLAB and can be sent to the controller for robot manipulation.

3.4 Send New Pose in MATLAB

The new pose data has to be converted in to `robtarget` format for uploading to the robot controller. Open new text files using `fopen` function and write new position, orientation and configuration data using `fprintf` function. Then upload the data files to the FTP server using `mput` function.

3.5 Read New Pose Data and Move the Robot

The file checking code is run in the RAPID editor to check whether the new files with updated pose data are reached the controller or not. Once the files are received, open the files from `diskhome` in read mode and read the string from the file using `ReadStr` function. Hence the updated data are stored `in text_pos, text_orient,` and `text_config` files as string, which can be converted to values using `StrToVal` function as follows.

```
FUNC robtarget robo_target (string postext1,string orient-
text,string robconfigtext);
bool1 := StrToVal(postext1,temptarget.trans);
bool2 := StrToVal(orienttext,temptarget.rot);
bool3 := StrToVal(robconfigtext,temptarget.robconf);
ENDFUNC.
```

Finally, the updated pose is extracted using `robo_target` function and the robot is moved linearly to the new position using `MoveL` instruction. `v150` specifies the speed of the robot (150 mm/s). `z100` defines the corner zone defined by the datatype `zonedata`. `GRIPPER` defines the tip of the tool attached that should move to the position specified.

```
Posenew := robo_target(text_pos,text_orient,text_config);
MoveL Posenew, v150, z100, GRIPPER;
```

With these basic operations, Sect. 4 demonstrates a positioning application on an industrial robot.

4 Positioning Experiment

To show the performance of the client–server communication strategy developed, a positioning experiment is conducted to move the robot based on the MATLAB commands. The experiment setup for positioning application is shown in Fig. 2 [13]. The robot controller, IRC5 compact is connected to PC using the Ethernet cable. The procedure explained in Sect. 3 is followed for establishing connection. The robot controller send the current pose to the PC as files. Hence the initial pose `(Ti)`

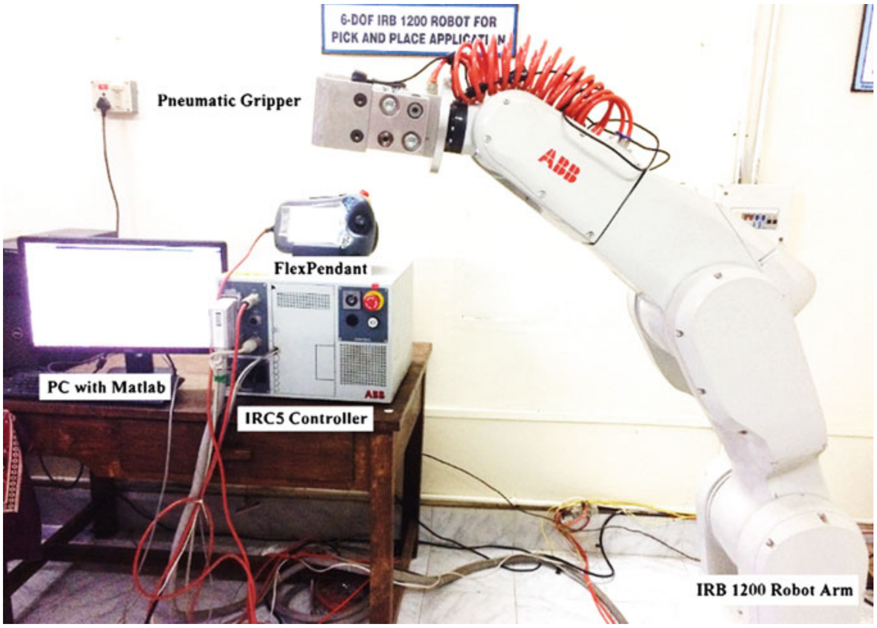

Fig. 2 Experiment setup

of robot tool center point (TCP) is obtained from the robot controller. MATLAB generates a Cartesian trajectory from pose `Ti` to `Tf` with 35 points using `ctraj` command from RTB [14]. The pose of the generated trajectory is sent to the robot controller one after another as files and the robot moves accordingly. The elapsed time to get a current pose from the controller is about 2–3 s. Thus, the robot manipulator successfully communicates with the robot controller for positioning experiment. The position (meters) and orientation (quaternion) obtained during the robot movement is plotted in MATLAB (see Fig. 3).

5 Conclusion

Various software tools are existing in the robotics and automation field of research for simulating the innovations, ideas and algorithms developed. The implementation in real-time is highly inevitable to validate the effectiveness of algorithms developed. It requires a solid interface between the software and hardware platforms, which often creates a hurdle with an incompatible system or add-on paid packages. Once secure communication between the robot and the software platform is established, dedicated systems can be structured for any applications like tracking, pick and place, gesture following, visual servoing, etc. This paper provides a simple and cost-effective method for interfacing an ABB robot with the widespread technical software

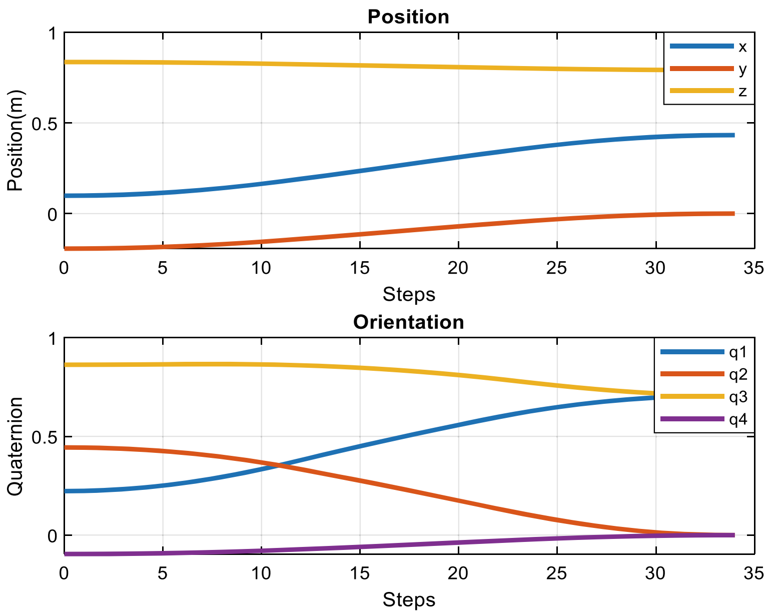

Fig. 3 Position and orientation of TCP

MATLAB and its real-time experimental validation. Even though the system seems to be a little sluggish in response, the overall performance of the system is inspiring with the scope of testing any proven algorithm in a real robot.

References

1. MATLAB (2015) MATLAB 2015b users guide. The MathWorks Inc., Natick
2. Krishnan MG, Vijayan AT, Ashok S (2020) Interfacing an industrial robot and MATLAB for predictive visual servoing. Industrial Robot. https://doi.org/10.1108/IR-05-2020-0100
3. Mineo C, Wong C, Vasilev M, Cowan B, MacLeod CN, Pierce SG, Yang E (2019) Interfacing toolbox for robotic arms with real-time adaptive behavior capabilities. University of Strathclyde, Glasgow, pp 1–12. https://doi.org/10.17868/70008
4. O'Kane JM (2013) A gentle introduction to ROS. CreateSpace Independent Publishing Platform, Scotts Valley
5. Corke P (2015) Integrating ROS and MATLAB. IEEE Robot Autom Mag 22:18–20. https://doi.org/10.1109/MRA.2015.2418513
6. Galli M, Barber R, Garrido S, Moreno L (2017) Path planning using Matlab-ROS integration applied to mobile robots. In: 2017 IEEE international conference on autonomous robot systems and competitions (ICARSC), Coimbra, pp 98–103. https://doi.org/10.1109/ICARSC.2017.7964059
7. Calusdian J, Yun X (2019) A simple and highly portable MATLAB interface for learning robotics. SN Appl Sci 1:890. https://doi.org/10.1007/s42452-019-0941-2
8. Karakaya S, Kucukyildiz G, Ocak H (2017) A new mobile robot toolbox for matlab. J Intell Robot Syst 87:125–140. https://doi.org/10.1007/s10846-017-0480-2

9. Turan A, Bogosyan S, Gokasan M (2006) Development of a client-server communication method for Matlab/Simulink based remote robotics experiments. IEEE international symposium on industrial electronics, Montreal, Que, pp 3201–3206
10. Pires JN (2000) Interfacing Industrial R&A equipment using Matlab. IEEE Robot Autom Mag 7(3):32–41. https://doi.org/10.1109/100.876909
11. ABB (2019) Product specification IRB 1200, Document ID: 3HAC046982-001
12. ABB (2004) Technical reference manual RAPID instructions, functions and data types, Document ID: 3HAC 16,581-1
13. Krishnan MG, Ashok S (2019) Kinematic analysis and validation of an industrial robot manipulator IEEE region 10 conference (TENCON), Kochi, pp 1393–1399. https://doi.org/10.1109/TENCON.2019.8929712
14. Corke PI (2011) Robotics, vision and control: fundamental algorithms in MATLAB. Springer, Berlin