Proceedings of 2008 IEEE International Conference on
Mechatronics and Automation

FA1-1

A Robot Manipulator Communications and Control Framework

Christian Kohrt, Anthony Pipe, Gudrun Schiedermeier, Richard Stamp, Janice Kiely

Abstract— **The use of industrial scale experimental machinery robot systems such as the Mitsubishi RV-2AJ manipulator in research to experimentally prove new theories is a great opportunity. The robot manipulator communications and control framework written in Java simplifies the use of Mitsubishi robot manipulators and provides communication between a personal computer and the robot. Connecting a personal computer leads to different communication modes each with specific properties, explained in detail. Integration of the framework for scientific use is shown in conjunction with a graphical user-interface and within Simulink as a Simulink block. An example application for assisted robot program generation is described.**

Index Terms—**Manipulator, communication, robot programming, manipulator motion-planning**

I. Introduction

PATH-PLANNING for industrial robots in complex environments where collision avoidance, in cooperation with the presence of a human worker in the robot work space is a research area which merits significant attention. The levels of automation within the automotive industry are expected to increase in future, so as to enhance the economics of manufacture. It is to be expected that in a future factory, human employees will co-exist alongside active industrial robots, to perform such tasks as body-part assembly and sealant application. For example, in car industry a moving conveyor is often used and separation of human employees and robot systems can hardly be realized. An increase in productivity can only be gained with either shorter production cycles or longer production times. Manufacturing industry must have a flexible production to offer highly diversified product mixes in a short delivery time, based on just-in-time small batched production [13]. With changing products production robots must also be programmed more often while the overall production time must be maximized to guarantee a high productivity.

The proposed framework is used in an ongoing project leading to foundations and algorithms for the industrial path-planning task which is the creation of a robot program within static industrial surroundings. The programming task will change from explicit to implicit programming.

A system overview in the next section gives a brief summary of physical devices involved. Section III describes the assisted robot program generation application, which makes use of this framework. Subsection IV.A gives detailed information of the communication modes possible with a Mitsubishi robot system connected to a CR 1 controller. An extension to the built-in communication modes is the data link control mode explained in subsection B. An overview of communication modes and their use is given in subsection C. The next subsection shows how the framework interacts with Matlab/Simulink followed by the *Visualization* component with collision detection and a visual servo control example. A conclusion is given in section V, showing the main use of the presented framework and important results.

II. System Overview

The equipment is shown in Fig. 1. External devices are the pointing device, vision system, robot controller, Teachpendant, robot and personal computer.

Fig. 1. System overview.

The robot manipulator communications and control framework is executed on the personal computer, which has an Ethernet and serial port connection to the robot controller. The Teachpendant and the robot are connected to the controller. The vision system and the pointing device are plugged in to the personal computer. The framework is

Manuscript received May 31, 2008. This work was supported in part by the Bavarian Research Foundation.

Christian Kohrt is with Berata GmbH, Munich, Germany (phone: +49-179-2921307; e-mail: christian.kohrt@berata.com).

Anthony G. Pipe is with UWE - University of the West of England, Bristol, UK (e-mail: anthony.pipe@uwe.ac.uk).

Gudrun Schiedermeier is with UASL – University of Applied Sciences Landshut, Germany (e-mail: gschied@fh-landshut.de).

Richard Stamp is with UWE - University of the West of England, Bristol, UK (e-mail: richard.stamp@uwe.ac.uk).

Janice Kiely is with UWE - University of the West of England, Bristol, UK (e-mail: janice.kiely@uwe.ac.uk).

978-1-4244-2632-4/08/$25.00 ©2008 IEEE

verified with a visual servo control application including collision detection and Matlab/Simulink integration.

III. Implementation of the Framework

The operation of an industrial robot is generally restricted to a small set of commands. Research was undertaken to integrate those commands that control movement with data from the path planning system.

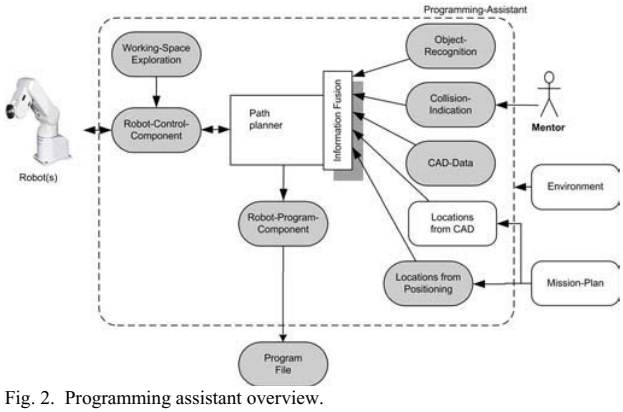


Fig. 2. Programming assistant overview.

An overview of this approach is given in Fig. 2. A programming assistant is proposed, which aims at creating a robot program for industrial robots.

Path-planning in robotics considers model-based and sensor-based information to capture the environment of the robot. Perception, initiated by sensors, provides the system with information about the environment and also interprets them. Those sensors are, among others, cameras or tactile sensors often used for robot manipulators.

The *Object Recognition* component converts the features of an image into a model of known objects. This process consists of segmentation of the scene into distinct objects, determining the orientation and pose of each object relative to the camera and determining the shape of each object. Those features are given with motion, binocular stereopsis, texture, shading and contour. Peter Corke's Machine Vision Toolbox for Matlab [10, 11] allows the user to easily use professional image processing capabilities of Matlab. In addition to the detection of the real environment, model-based data such as CAD-data is used to render the world model more accurate. CAD derived data from simulation systems for example RobCAD are exported as DXF files with all physical locations attached and stored within the world model. Attached locations of CAD-objects are employed so as to acquire information concerning the start and target locations. The path information, such as gluing, painting etc. must be given within the programming assistant. Target locations can also be defined by manual movement of the robot and visual servo control. The latter uses a pointing device to show the robot system the target location. The robot automatically moves to the shown location and stores the position.

The *Working Space Exploration* component gathers information of the environment by moving the robot by manual movement, by random movement with collision indication and by running a robot program, also with collision indication. During motion, the robot ascertains which parts of the space are free in the robot coordinate system either with visual feedback or manual collision indication. This knowledge will become more accurate during the planning process.

The kinematic of a robot is often not precisely known. A neural network based approach is employed to ascertain the dynamic model of a robot by the adoption of a learning process. Visual servo control and working space exploration is used to autonomously learn the robot kinematic.

The *Robot Control* component communicates with the robot and provides direct robot control, serial/Ethernet connection, parameter edit/read/write, program upload and download, real-time movement control, robot system backup/restore, external control over UDP[1] and equipment control.

The robot program is implicitly stored within the robot programming assistant as a trajectory. A transfer to a robot specific program is done in the *Robot Program* component in two steps: First, translation to a pseudo robot program with no other information than the provided trajectory, and, second, generation of the specific robot program with additional configuration commands and specific syntax of the robot programming language. The two step transfer can also be adopted to support other robot types and manufacturers.

The employment of industrial robots without modification of the robot and its controller is necessary for a rollout in industry. The Mitsubishi industrial scale experimental machinery robot system used is the RV-2AJ robot connected to a CR1 controller. It is well documented, industrially proven and communication with a personal computer is possible. Its commercial viability has already been demonstrated in the manufacture of car sub-assemblies, semiconductor memories and other industrial/consumer goods [12]. The main areas of application are assembly, manufacture, pick & place and handling.

The robot control framework is used to verify a path planning algorithm developed at the University of Applied Sciences Landshut in cooperation with the University of the West of England.

The Mitsubishi CR1 controller employed is equipped with a built-in RS232 communication port and an external Ethernet extension box. Both ports are used for communication in the framework.

The tool Matlab/Simulink from TheMathworks [17] is often used in the area of robot control. To use the robot control framework in Simulink it must be encapsulated in Simulink blocks. This brings the benefit to have all of the control within the model and opens the use of Matlab/Simulink for robot control applications in a model driven architecture design.

[1] User Datagram Protocol

IV. THE FRAMEWORK

The aim of the robot control framework is to simplify the usage of robot control and to cover all of the needs originated from robot control applications. It consists of the components described in the sections robot communication, data link control mode, overview of communication modes and their use, connecting the framework to Matlab/Simulink and visualization. An example is given in section F.

A. Robot communication

The available communication modes are controller link mode (CL), data link mode (DL) and real-time external control mode (RTEC).

Controller communication mode (CL)

The controller communication mode is used to set parameters, send robot control commands and read the robot status. Getting the status information during movement of the robot and controlling the robot in real time is not possible in this mode. The data sent over Ethernet is not encoded and can be read in plain text. Thus, it is possible to listen to the Ethernet communication between the controller and the personal computer. Generally, the protocol format for sending commands is the following:

[<Robot No.>];[<Slot No.>];<Command> <Argument>

Each command is followed by a message sent by the controller with status information and the result. Table I states the pattern of such returning messages, where each star stands for a digit:

TABLE I
STATUS OF SENT COMMANDS

Commands	Contents
QoK****	Normal status
Qok****	Error status
QeR****	Illegal data with error number
Qer****	Error status and illegal data with error number

Real-time external control mode (RTEC)

Real-time external control of the robot is useful for direct robot control, where the trajectory is calculated manually. The real-time external control mode is based on the UDP networking protocol. A UDP datagram is a simple and very low-level networking interface, which sends an array of bytes over the network. Even though they are not reliable their low overhead protocol makes datagram transmission very fast. Since the connection is a single point to point connection between personal computer and controller, UDP can be handled easier. Sending and receiving of packets is monitored and a timeout exception is thrown if the communication does not meet the requirement in time.

Mitsubishi provides a simple C communication program example. Running time is crucial, since every communication cycle has a time period of 7.1ms, which is dependent of the robot hardware. A plain Java port is not capable to communicate with the robot controller in time and leads to a loss of UDP packages. Movement of the robot was not continuous any more.

A dynamic link library for RTEC mode created in C is connected with JNI to Java. The library could also be used in Simulink to build a “hardware-in-the-loop” low level robot control application. This gives full control of the robot and code generation from Matlab/Simulink is possible.

Data link mode (DL)

The data link mode connects a controller with a personal computer or vice versa. Usually, it is used to send robot status information from internal robot sensors or other data to its receiver.

B. Data link control mode

The data link mode is extended by a control component, which gives the opportunity to control the robot while getting status information. The personal computer and the robot controller are arranged in a cascaded control system, where the robot controller calculates the trajectory given by the personal computer in form of movement commands. Those commands can be sent at any time over Ethernet or the serial port and the robot manipulator follows the trajectory without stopping between the buffered movement commands.

Multitasking

Multitasking is used to run the data link control programs in parallel. Multitasking is executed by placing the parallel running programs in slots. Data is passed between programs being executed in multitask operation via program external variables and user defined external variables.

Multitasking configuration

The main control program MULTITASK is executed first in slot 1. It sets the variable M_01 and M_02 to zero and starts the programs DATALINK and CONTROLLINK in slot 2 and slot 3. In line 80 and 90 the program waits for the variables M_01 and M_02 to be set from the other programs to stop execution. The main program multitask.mb4:

```
10 RELM
20 M_01=0
25 XLOAD 2,"DATALINK"
30 XRUN 2,"DATALINK"
40 WAIT M_RUN(2)=1
50 M_02=0
55 XLOAD 3,"CONTROLLINK"
60 XRUN 3,"CONTROLLINK"
70 WAIT M_RUN(3)=1
80 WAIT M_01=1
90 WAIT M_02=1
100 XSTP 2
110 WAIT M_WAI(2)=1
120 XSTP 3
130 WAIT M_WAI(3)=1
140 GETM 1
180 HLT
190 END
```

The DATALINK program in slot 3 shown below sends the timestamp, current joint position, current speed of the tool

center point and current position.

Sending is looped over lines 100 to 130 and it sends until a zero value is received. After closing the communication port, the program notifies the MULITASK program in slot 1 that the signal is turned on by means of the external variable M_02. The communication program datalink.mb4:

```
10 WAIT M_02=0
20 M_TIMER(1)=0
30 OPEN "COM2:" AS #2
35 INPUT #2,DATA
40 IF DATA = "0" THEN 160
100 PRINT#2, M_TIMER(1), "|", P_CURR, "|", J_FBC,
"|", J_CURR, "|",M_RSPD(3)
130 GOTO 100
160 M_02=1
170 WAIT M_02=0
180 END
```

The CONTROLLINK program moves the robot manipulator by receiving and executing movement commands. This program runs in a cyclic mode and no user interaction such as moving the robot with the Teachpendant or by robot commands in controller communication mode is possible. For control communication the RS232 port is used, which is a slow connection but fast enough for direct robot control commands. The data link mode is extended by a movement command and leads to the data link control mode. The movement program controllink.mb4:

```
10 WAIT M_01=0
20 OVRD 100
30 GETM 1
40 CNT 1, 300
50 SERVO ON
60 OPEN "COM1:" AS #1
70 DEF JNT JNTPOS
80 INPUT #1, JNTPOS
90 MOV JNTPOS
100 GOTO 80
```

With the CNT command, the robot continuously moves to multiple movement positions without stopping at each movement position.

C. Overview of communication modes and their use

Use cases for robot control are defined in table III. Since it is not possible to send control commands and information requests over one connection, a second connection is always needed to get actual status information during motion.

TABLE II
COMMUNICATION MODES

Mode	Phys. layer	Command type	Feed back type	UC 1	UC 2	UC 3	UC 4	UC 5	UC 6
RTEC	ETH	SDO	SDO	X	-	-	-	-	-
DL	ETH	SD	SD	-	-	-	X	X	X
DL	RS232	SD	SD	-	-	-	X	X	X
CL	ETH	Robot command		-	X	-	-	-	-
CL	RS232	Robot command		-	-	X	-	-	-
CL	ETH	Robot program		-	-	-	X	-	-
CL	RS232	Robot program		-	-	-	-	X	-

(RTEC – Real Time External Control; DL – Data Link; CL – Control Link; ETH – Ethernet; SDO – Serialized Data Object; SD – Serialized Data; UC – Use Case)

An overview of communication modes and use cases of table III is given in table II.

TABLE III
USE CASES

Use-case	Description
1	Direct robot control over Ethernet with feedback. Either the mentor or the path-planning-system can move the robot manually. No controller calculations are involved.
2	Robot operation with singular movement commands over Ethernet. The controller calculates the path. Feedback data can be retrieved by Ethernet connection after finishing movement.
3	Robot operation with singular movement commands over serial port. The controller calculates the path. Feedback data can be retrieved by serial port connection after finishing movement.
4	Robot operation with robot programs over Ethernet. The controller calculates the path. Feedback data can be retrieved either by Ethernet or by serial port connection.
5	Robot operation with robot programs over serial port. The controller calculates the path. Feedback data can be retrieved either by Ethernet or by serial port connection.
6	Robot operation by two data-link channels. One sending channel over serial port and one receiving channel over Ethernet. The robot has to be programmed so that it is possible to send movement-type and data.

D. Connecting the framework to Matlab/Simulink

This section shows the mature steps and important key issues to integrate the Java framework with a SWT[2] user interface to Matlab/Simulink.

Matlab/Simulink integration

The framework must be executed in its own thread to avoid a freeze of the Matlab thread. The implementation as a singleton of the framework GUI[3] guarantees that only one single instance of the GUI is running per Matlab instance.

Communication must be established between Matlab and Java. Calling Java classes from Matlab is supported by default. To communicate back to Matlab/Simulink, two cases of software usage are possible: As a standalone client and as a plug-in. A standalone client is running outside of Matlab/Simulink, whereas a plug-in is started within. This has a great impact on the communication of Matlab and Java. While the standalone client must have interprocess communication, a plug-in does not require this.

Fig. 3. Interprocess communication.

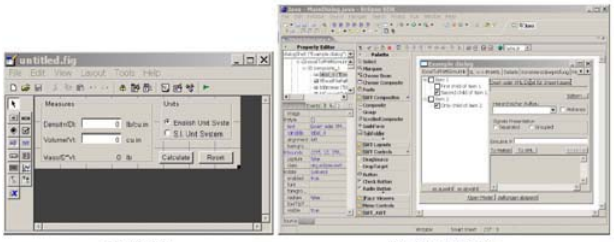
Generally, DLL[4] libraries of Matlab/Simulink can always be called by native system calls. JNI[5] is a wrapper for such

[2] Standard Widget Toolkit
[3] Graphical user interface
[4] Dynamic Link Library
[5] Java Native Interface

system calls and thus can be used. A more convenient possibility is COM[6] or DDE[7] communication. Matlab supports both, the COM and the DDE technology. COM technology is to be used, because the DDE communication server must be switched on in newer versions of Matlab (R14 onwards). In contrast, a plug-in does not need an interface for interprocess communication. In fig. 3 COM/DDE communications is illustrated for standalone clients.

The graphical user interface

Simulink is a platform for multidomain simulation and model based design for dynamic systems. It provides a customizable set of block libraries. Models are built from these blocks that can be connected to solve a given engineering challenge. Usually such systems are quite complex and users not familiar with the model will have difficulties to modify model parameters and to control the model. Therefore, a centralized user input to the model can be realized through a GUI. A GUI development environment (GUIDE) is shipped with Matlab and, thus, becomes the standard tool for GUI creation. A Java application within Matlab/Simulink has greater functionality, i.e. interconnection to a server. It also allows the use of another GUI library such as SWT or Swing for standardized development of complex GUIs.



(a) GUIDE (b) SWT Designer

Fig. 4. The GUI editors GUIDE and SWT.

SWT-based applications integrate seamlessly into the host environment. The library is an adapter to the native widgets. The design of SWT as an adapter makes a small library possible. These libraries must be available on the target computers.

The Simulink Blockset

To allow experiments in model based design methodology the Java robot control framework is integrated into a Simulink blockset. However, code generation is not possible with those blocks. For simulation, additional blocks for forward and backward calculation are needed. A Simulink block usually supports simulation and code generation. Since Java is used, this feature cannot be supported.

The blockset consists of the blocks listed in table IV. It is not possible to use more than one robot control block at the same time because every block needs its own explicit connection.

A stopped robot movement is a movement with stops between two movement commands. It is also a blocking command, which means movement finish must be awaited to send the next command. A non blocking continuous movement is a movement that can do continuous movements also between two commands and the movement command can be sent at any time. The status block always uses the data link communication mode.

TABLE IV
SIMULINK BLOCKS

Block name	Operation mode	Description
Status	data link	continuous measurement
RelJoint	data link control	continuous, non blocking
RelCart	data link control	continuous, non blocking
CircularMov	controller operation	stopped, blocking
LinMovJoint	controller operation	stopped, blocking
LinMovCart	controller operation	stopped, blocking
JointMovJoint	controller operation	stopped, blocking
JointMovCart	controller operation	stopped, blocking

E. Visualization

Visualization is done with a Java3D scenegraph. But not only viewing the robot but also collision testing should be done combined with ODEJava, a physical simulation system.

Collision detection

The built-in Java3D testing does only tests in every frame. Collisions of fast moving objects could take place between two frames that leads to an unrecognized collision. The Open Dynamics Engine (ODE) library written in C and its Java binding ODEJava is used to do collision detection. The ODEJava project allows using ODE with Java. ODE is a free, industrial quality library for simulating articulated rigid body dynamics in virtual reality environments. It has built-in collision detection. The Project also contains tools for binding ODEJava into Xith3D, jME and Openmind scenegraph projects. Since Java3D scenegraph is used, development of a graphics engine is necessary to combine ODEJava with Java3D.

The *Viewing* component also supports visual display of DXF CAD-data as well as any Java3D object. The ODEJava library is used for collision detection of basic geometric objects. DXF-data cannot be used with ODEJava, but can be viewed with Java3D. Collision detection with complex DXF data is therefore rudimentary supported. Collision points and vectors are hard to calculate from DXF-data, but can be done manually. The requirements for DXF-data collision detection are fulfilled. The basic geometric objects are fully supported and have higher requirements in terms of accuracy, because the neural net used in the *Path Planner* component is simulated by those objects [14].

F. Visual servo control

Visual servo control is used for user interaction with the robot system by a pointing device for example used in the

[6] Component Object Model
[7] Dynamic Data Exchange

Working Space Exploration or *Location Positioning* components.

The transformation of picture coordinates of the camera views to robot coordinates by a neural net is learned. The system interprets then the pointing device of the mentor and controls the robot so that it moves in the direction of that point.

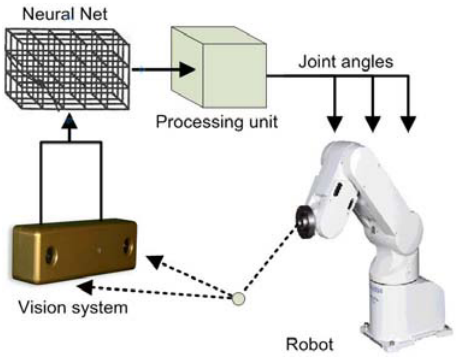


Fig. 5. Example application

An extension to Kohonen's model [9] is implemented to autonomously learn the positioning of a robot arm to a visually given point (Fig. 5). To get information of the position of the objects in space, the robot cell is equipped with two cameras, which monitor the robot cell. During training, the position of the target location within the working-space is randomly chosen. The target location is monitored from the cameras and their signals are applied to the neural net. Every neuron is responsible for a subspace of the robot cell. If a target location is chosen, this neuron becomes activated and provides control signals to the robot controller. The position of a robot arm with five joints is not only a five dimensional vector, but every camera delivers its two dimensional point of the viewing pane. The neural net has to transform that position information to control signals for the five robot joints.

More information about the robot, the cell, the cameras or its positions in space is not needed. Moreover, this must be learned by the neural net.

At the beginning, the robot will move to incorrect robot positions. The difference to the target position is used to train the net. Then the robot will be given the next target position which gives the system the opportunity to learn a second time and so on. The robot is an autonomous learnable system.

V. Conclusion

Robot control applications need a connection to the real robot system. Sending robot control commands as well as receiving information of the robot status and position is necessary especially for path-planning applications, where the focus is on algorithm development. This framework offers the possibility use a standard industry robot system. The framework extends the Mitsubishi CR1 controller family robot system to send robot commands during movement of the robot manipulator without stopping between two commands and to receive robot information during movement. All communication modes over serial port and Ethernet are discussed. Besides the use of the robot control framework as a standalone application, it can also be used with Matlab/Simulink and interconnected within Simulink models to support a wide range of robot control applications.

Acknowledgment

We thank Stefan Holzer for his support and valuable work of an earlier project at the UASL University of Applied Sciences Landshut.

References

[1] Jim Tung, "The impact of model based design on product development," in Model-Based Design Conference, 2005.
[2] *Mitsubishi-Electric Manual*, Connection with personal computer, 2005.
[3] *Mitsubishi-Electric MELFA Industrial Robots Instruction Manual (Functions and Operations) CR1/CR2/CR3/CR4/CR7/CR8 Controller*, Mitsubishi-Electric, 2003.
[4] *Mitsubishi-Electric MELFA Industrial Robots Instruction Manual CRn-500 Expansion Serial Interface*, Mitsubishi-Electric, 2003.
[5] *Mitsubishi-Electric Mitsubishi Industrial Robot CRn-500 Series Personal Computer Support Software Instruction Manual*, Mitsubishi-Electric, 2003.
[6] *Mitsubishi-Electric MELFA Industrial Robots Instruction Manual Controller CR1*, Mitsubishi-Electric, 2002.
[7] *Mitsubishi-Electric Ethernet Interface CRn-500 series Manual*, Mitsubishi-Electric, 2002.
[8] *Mitsubishi-Electric MELFA Industrial Robots Instruction Manual RV-1A/2AJ Series,* Mitsubishi-Electric, 2002.
[9] H. Ritter, T. Martinetz, K. Schulten, "Neuronale Netze," Oldenbourg, 1994.
[10] P. I. Corke, "The Machine Vision Toolbox," in IEEE Robotics and Automation Magazine, 12(4), pp 16-25, November 2005.
[11] *Programming - Matlab Image Processing Toolbox Version 2*, Mathworks, 1997.
[12] http://www.mitsubishi-automation.com, 2008.
[13] University of Karlsruhe, "EURON II Research Roadmap," www.euron.org, 2005.
[14] C. Kohrt, G. Schiedermeier, A. G. Pipe, J. Kiely, R. Stamp, "Nonholonomic Motion Planning by Means of Particles," in IEEE International Conference on Mechatronics and Automation, Luoyang, China, pp 729-732, June 2006.
[15] C. Kohrt, T. Reicher, R. Rojko, "With Model-Based Design to Productive Solutions: Professional GUIs for Simulink by Utilizing the Java SWT Library," in Design & Elektronik, Stuttgart, Germany, May 2006.
[16] P. I. Corke, "Visual Control of Robots: High-Performance Visual Servoing," New York: Wiley, 1996.
[17] http://www.mathworks.com, 2008.