Integration of a Robotic Endoscope Holder with the Surgical Assistant Workstation Software Framework

Release 1.0

Jessie Young[1], Haytham Elhawary[2] and Aleksandra Popovic[2]

June 1, 2011

[1]Johns Hopkins University, Baltimore, MD
[2]Philips Research North America, Briarcliff Manor, NY

Abstract

We have integrated the Philips research robot arm AAMIT with the Johns Hopkins cisst library, an open-source platform for computer assisted surgical intervention, for assistance during minimally invasive cardiac surgery using image-based steering of an endoscope. The development of a Matlab to C++ wrapper to abstract away servo-level details facilitates the rapid development of a component-based framework with "plug and play" features. This allows the user to easily exchange the robot with an alternative manipulator while maintaining the same overall functionality.

Contents

1 Introduction

For robotic research in the medical domain, it is very common to develop several prototypes with varying levels of complexity in design and functionality. It is therefore paramount that the software framework, which controls and integrates the different data flows within the system, work consistently with each of the differing robot prototypes. At Philips Research North America, we have used several robotic prototypes for research purposes, firstly the LARS robot [1] developed at IBM and JHU. Recently we

have developed our own 8-degree of freedom (DOF) robotic arm, called Articulated Arm for Minimally Invasive Therapy or AAMIT. Integration between the Laparoscopic-Assisted Robot System (LARS) robot and our software algorithms was performed using the Computer Integrated Surgical Systems library [1] developed at Johns Hopkins University [6]. The work in this paper presents the integration of AAMIT with the Hopkins CISST libraries in a seamless manner, so that both the LARS robot and AAMIT robot can be connected to the system in an almost "plug and play" fashion and used with our developed software algorithms. This may be accomplished by integrating AAMIT with the cisstMultiTask class that is part of the cisst library, which provides the component-based framework for the cisst package [6]. The "provided," "required," "input," and "output" interfaces that each component may include are used to connect different hardware devices, each with a specific functionality, to each other. Since the AAMIT software library has a low-level joint servo controller, the integration of the robotic arm demonstrates the "plug and play" capabilities of this framework. To achieve this functionality, we had to standardize the command name strings and their data types. We accomplished this by developing a Matlab to C++ wrapper.

1.1 AAMIT Robotic System

The AAMIT robotic system is pictured in Figure 1 while being used as a robotic endoscope holder. It has been designed to emulate a human arm, and is approximately the same size of an arm. Existing control for the robot arm consists of a Linux-based real-time servo controller that can control each of the eight motors independently of each other. In order to communicate with the servos, a Windows based user interface on a separate PC workstation is provided, which allows each of the joints of the arm to be controlled using a Matlab- based interface. The interface uses the open-source Matlab Robotics Toolbox [5] to calculate kinematics, and sends the subsequent move commands over a network connection to the servo controller on the Linux machine.

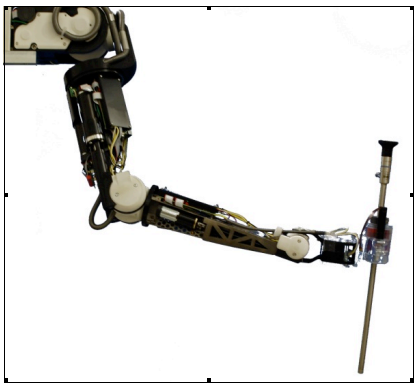

Figure 1 The AAMIT robot with a custom-made endoscope holder acting as an end effector.

1.2 Hopkins Software

The cisst package is a collection of libraries designed to ease the development of computer assisted intervention systems. One motivation is the development of a Surgical Assistant Workstation (SAW), which is a platform that combines robotics, stereo vision, and intraoperative imaging (e.g., ultrasound) to enhance a surgeon's capabilities for minimally-invasive surgery (MIS) [6]. The Surgical Assistant Workstation (SAW) is an open source, cross-platform C++ component-based software framework that makes it easy to integrate devices, such as robots, haptic interfaces, tracking systems, imaging systems, and other devices used for computer assisted intervention applications [6]. The software can be downloaded from the cisst SVN repository. [www.cisst.org/saw/Main_Page], and is available under an open source license [trac.lcsr.jhu.edu/cisst].

We would like to develop a component-based framework in which related functions or data are encapsulated by different modules [6]. We choose to separate the system into different components based on functionality, as this would then allow exchanging hardware devices, which provide similar functionality, in an easy fashion. Hence incorporating a research robot into the cisst framework would

allow it to perform functions such as forward and inverse kinematics calculations for path planning, trajectory interpolation, visual servoing, and control using input devices such as rate-based control using a 3D Mouse ® or a haptic device such as the PHANTOM OMNI ®.

2 Architecture

Two main methods of communication between components exist. CORBA (Common Object Request Broker Architecture) calls allow the joint controller on the PC, which calls a Matlab session to calculate kinematics, to communicate with the servo motor controller (RobotarmGUI) on the Linux machine. RobotarmGUI takes as input parameters the maximum move position (degrees), maximum velocity, and maximum acceleration. Microsoft COM (Component Object Model) calls allow Matlab to be used as an UI for controlling the servos in the arm. This is used by the executable MatlabToRobotarmWithIK, a Microsoft COM server that performs the data translation between Matlab and RobotarmGUI. It exposes a COM interface on the user side and connects to a CORBA interface on the target.

We choose to use Matlab Robotics Toolkit to calculate the forward and inverse kinematics of the robot arm, although the cisst library also has built-in kinematics capabilities. However, in order to abstract away the servo-level details and instead expose to the cisst framework only the joint-level controls, we developed a wrapper called MatlabWrap that allows the C++-based cisst libraries to communicate with the Matlab-based joint-level control. Using existing components allows for rapid prototyping of this image-guided interventional system.

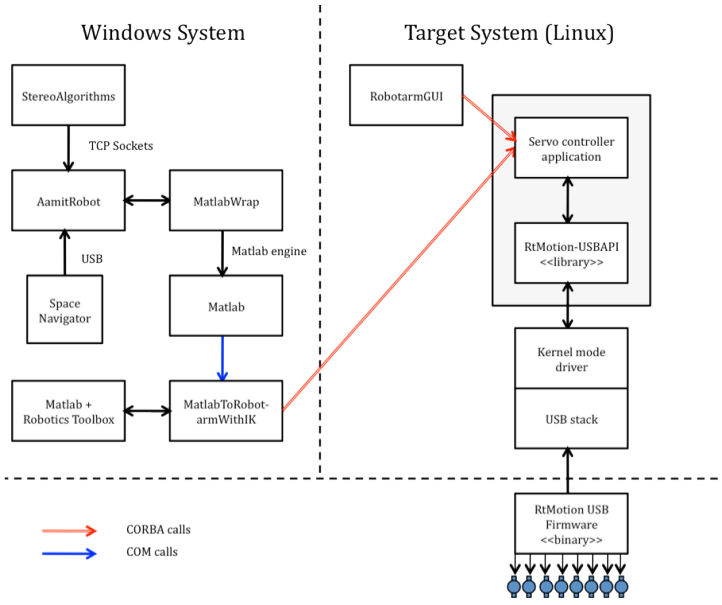

Figure 2 System architecture overview.

2.1 Matlab Wrapper

This C++ to Matlab wrapper optimizes the functionality of both languages: C++ allows us to build a more robust and complex program capable of integrating with existing control and imaging software and Matlab can perform the numerical calculations for the kinematics using existing robotics functions.

Some examples of MatlabWrap functions that abstract away the servo-level details include:

> ***JointPTP***(vector<double> normalTgtPos): Accepts a 7x1 input vector, one for each arm joint, and may be used to home the arm to an acceptable position where it has maximum range of movement before executing other commands, such as CartPTPRelative.
>
> ***CartPTPRelative***<vector<double> dx): Accepts a 6x1 input velocity vector (meters), with x, y, z, and rotational x, y, and z velocities.
>
> ***Home():*** Homes the entire arm to a user-defined initial position. It also resets the incremental encoders.

2.2 Task Structure

Ehsan Basafa and Seth Billings from Johns Hopkins University developed the LarsRobot application as a cisst-integrated control software for the LARS surgical robot with teleoperative capability using a Phantom Omni and SpaceNavigator 3D Mouse" [1]. This work required developing a similarly structured, multi-threaded application that interfaced with the AAMIT robot, which we named AamitRobot.

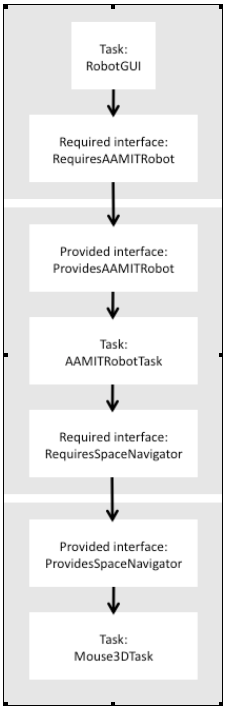

Figure 3 Task interface structure for AAMIT application

For AAMIT to share the existing LARS's RobotGUI control window, the Matlab wrapper needs to be compatible with the cisstMultiTask library used by LARS. Therefore MatlabWrap needs to inherit the base class mtsGenericObject, defined in cisstMultiTask.

cisstMultiTask allows the user to define multiple tasks, each of which corresponds to a thread with a periodic user defined function. Multiple tasks communicate using task interfaces [6]. Two tasks were created, aamitAppTask (for the UI) and aamitRobotTask (robot control) based on existing aapTask and robotTask files for the LARS.

To allow the cisst library to communicate with the Matlab-controlled COM object, a MatlabWrap object needs to be created within aamitRobotTask. The engine must be opened inside the *Run()* method of aamitRobotTask on the first run, not inside the constructor of aamitRobotTask or inside Startup() or Configure(), as this would open a new Engine session each time the *Run()* function is executed.

The Space Navigator 3D can be used to control the joint position of the end effector. The input x, y, and z velocities from the Navigator are scaled (which may be adjusted by modifying the gain matrix). The rotational x, y, and z components may be set to 0. A cutoff for the maximum amount of movement is set at 0.05 m; any input exceeding this number will be set to 0.05 m.

2.3 Sample Command

The following diagram shows the transfer of control for executing a single CartPTPRelative command:

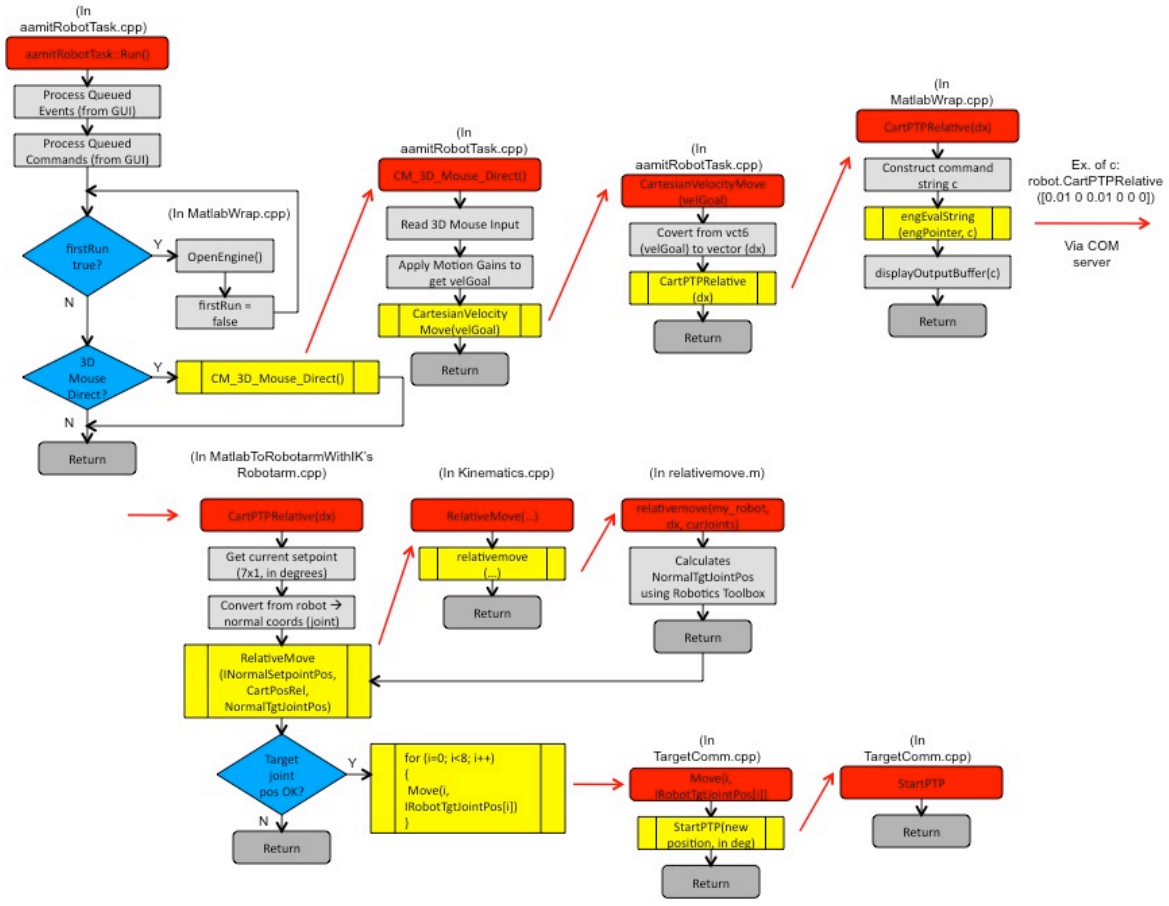

Figure 4 Example aamitRobotTask Run Method Processing

3 Discussion and Conclusion

This work has allowed a seamless integration of a robotic arm into the cisst framework. The robotic arm presents a Matlab interface with which the user can send high level commands, which are then transferred into servo commands at a Linux machine through a TCP/IP connection. By abstracting away the low level control and providing a C++ to Matlab wrapper, the robot arm has been incorporated into the framework,

allowing it to use all the other components which are available such as the use of a 3D mouse or the implementation of virtual fixtures [3].

4 Acknowledgments

The authors would like to thank Peter Kazanzides, Anton Deguet, and Seth Billings from Johns Hopkins University for their help on the cisst library and the Lars robot, and Doug Stanton for his technical help in the set-up for the AAMIT robot in the laboratory environment.

References

[1] E. Basafa and S. Billings. *Teleoperation of the LARS Robot*, Johns Hopkins University, 2009.

[2] T Xia and M Balicki. *JHU-MiMed EndoWrist Actuator*, Powerpoint Presentation, Engineering Research Center for Computer Integrated Surgical Systems and Technology, Johns Hopkins University, 2007

[3] M. Li, M. Ishii, and R.H. Taylor. *Spatial Motion Constraints Using Virtual Fixtures Generated by Anatomy*, IEEE Transactions on Robotics, 2007.

[4] R. Beekmans. *Matlab To Robotarm Interface User Manual, v 1.0*, Philips Applied Technologies, 2010

[5] P. Corke. *Documentation for Robotics Toolbox for Matlab Version 8*, 2008

[6] Johns Hopkins University LCSR*, cisst library,* trac.lcsr.jhu.edu/cisst*, 2010*