Client/Server Framework for robot control in medical assistance systems

L. Richter\textsuperscript{1,2}, F. Ernst\textsuperscript{1}, V. Martens\textsuperscript{1}, L. Matthäus\textsuperscript{1}, A. Schweikard\textsuperscript{1}
\textsuperscript{1}University of Lübeck, Institute for Robotics and Cognitive Systems, Lübeck, Germany
\textsuperscript{2}University of Lübeck, Graduate School for Computing in Medicine and Life Sciences, Lübeck, Germany

Keywords: Robotics, Medical Applications, TCP/IP, Tracking, Hand-eye-calibration

Purpose

In medical applications, robots help physicians and surgeons in positioning and motion compensation issues. Existing robotized systems benefit from new developments in robotics research. Due to different attributes and features, it is nearly impossible to switch to a different robot in an existing application without fundamentally changing the application itself.

In this work, we present a client/server framework for different robot types to separate the application from direct robot communication. Since this framework is based on a TCP/IP interface, programming language, platform and computer independent development and use is possible. The C++-source-code is available on request.

Methods

On the server side, the forward and backward calculation is implemented based on the robot specific DH-parameters. Thus, a new robot can easily be included in the server without changing the client. The framework provides a unique robot home position, a unique rotating direction and a well-defined coordinate system for any robot. Hence, the server externally describes the same behavior for the robots. So far, our framework supports four different robots: Adept Viper s850 (Adept Technology, Inc., Livermore, CA, USA), Kawasaki FS03N (Kawasaki Heavy Industries, Ltd., Akashi, Japan), Kuka KR 3 and KR 16 (Kuka AG, Augsburg, Germany).

Additionally, for the Kawasaki FS03N, a special spin command allows the robot to turn the sixth joint for more than 360°. To have smoother robot movements, a real-time mode is available for the Adept Viper s850.

For communication with the server a specialized server interface on the robot is necessary. This interface runs directly on the robot and is written in the respective programming language of the robot. The program transforms the commands sent by the server to the specific robot commands and executes them. An answer from the robot is transformed back and sent to the server.

The Adept robot allows to directly attach additional hardware via serial or firewire ports. This hardware is also supported by the server. A servo-electric gripper (Schunk GmbH, Lauffen/Neckar, Germany) can be connected to the Adept Viper s850 that is controlled by the Adept robot. Furthermore, tracking systems can be connected and used directly by the robot in this way. This is very useful for a direct hand-eye-calibration with no communication latencies of the robot and the tracking system [1]. The other robot types can only use tracking systems via a special TCP/IP client/server framework for tracking systems [2].

For safety reasons, only one server can be started per robot. Otherwise, movement commands could overlap and could lead to dangerous situations. Instead, it is possible to connect more than one client, the first one in active mode, the others in a read-only mode. Thus, they cannot control the
robot but can request status information which can be used to monitor and visualize the status of the robotized systems. The communication concept is illustrated in Figure 1.

Fig. 1 Communication concept of the robot server with standardized interface for any application and specialized interfaces for the different robots.

For safety issues only one server per robot can be started. A second client thread can only connect to a server in ‘read only’ mode. Thus, no active control of the robot movement is possible for this client. Position and status values can be read by this second client. Additional hardware, like a gripper or tracking system, can be connected directly to the Adept robots via serial or firewire connections or via TCP/IP for any robot.

Results

Thanks to the open-source cross-platform build system CMake, different versions of the server for several operating systems can be generated. We have successfully tested the server for Windows XP and Vista, Linux and Mac-OS X.

To measure the latency we used two identical Adept Viper s850 robots placed next to each other. To each robot, we attached a tracking LED connected to the accuTrack system (atracsys LLC, Bottens, Switzerland). Both robots were calibrated to the tracking camera. The LEDs' positions were recorded using the tracking framework [2]. The second robot was controlled using the presented server. The first robot was then moved using its Manual Control Pendant. The second
robot was controlled by a program to follow the motion of the first robot as detected by the tracking system. Both LEDs' positions were recorded independently and the time shift between the motion trajectories of the two LEDs was evaluated as the combined latency.

![Diagram showing latency components](image)

Fig. 2 Combined latency of the complete simulation setup. The top graph shows the individual parts of the system latency of 110 ms: network communication, marker acquisition time, robot inertia and computation time. The bottom graphs explain the individual components of the network latency and the different computations involved.

The time delay between the two curves was found to be approximately 100 ms. Further experiments, using simple ping-pong commands to both the tracking and robot servers as well as to the robot from the robot server, helped in determining the latency caused by network communication (both 0.25 ms to the robot and the tracking server, 3.2 ms to the robot controller). Timing commands on a client program running on ubuntu 8.04 with a real time kernel showed the individual computation times (0.1 ms in the client program, 0.25 ms in the tracking server and 2.5 ms in the robot server). The biggest factor of the robot inertia is caused by starting the actuators and getting the joints move. An overview of all latencies determined is given in Figure 2.

**Conclusion**

As navigation plays an important role in medical assistance systems, this framework can be easily combined with another TCP/IP-based framework for different tracking systems. This combination allows the user to develop robot based systems without being fixed to a given hardware.

Latency measurements have shown that the latency is about 100 ms from sending the command to the execution of the robot. Applied implementations of the server in medical systems have proved that this latency is acceptable. Current examples of the framework in medical applications are a robotized Transcranial Magnetic Stimulation (TMS) system [3] and a robot-supported ultrasound system [4]. These applications have shown that the framework is reliable in medical surroundings and very helpful in the development and further improvement of the systems.

Using read-only clients is very useful in the development phase to identify failures in the system in an early state or in the end use for remote system observation. For teaching purposes, it helps to show how the systems work without risking safety issues with an active control. An enhancement for the management of several clients for one server would be the introduction of a mutex. The first
client could pass the mutex to the second client so that the second becomes the active client and can control the robot. The first client becomes a passive client and is in a read-only mode.

Acknowledgements

This work was partially supported by the Graduate School for Computing in Medicine and Life Sciences funded by Germany’s Excellence Initiative [DFG GSC 235/1].

References:


