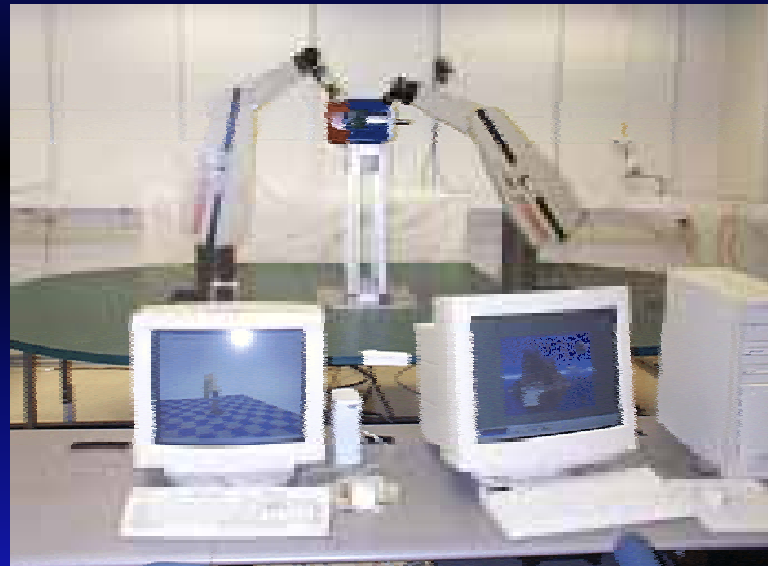# Telerobotic Decontamination and Decommissioning with QRobot, a PC-Based Robot Control System

Markus Loffler, Nick Costescu, Erkan Zergeroglu, Darren Dawson

Department of Electrical and Computer Engineering
CLEMSON UNIVERSITY

- Introduction
- Application
- Motivation for Developing a PC Based System
- Overview of the System
- Software Components
  - Client/Server Architecture
  - Joint Level Control
  - High Level Control
  - Disassembly Program
  - Operator Programs
- Experimental Results
- Conclusion
- Current and Future Research

Motivation

The U.S. Department of Energy (DOE) is facing the decontamination and decommissioning (D&D) of a high number of surplus facilities.

The DOE is looking for new and innovative technologies that allow D&D operations to be faster, safer, and more cost-effective, and do not expose workers to hazardous materials.

Semi-autonomous telerobotic systems

- D&D operations are remotely controlled
- Higher level tasks are performed autonomously, triggered by the operator
- Operator Interfaces include video and virtual reality (VR) feedback

QRobot

QRobot is a semi-autonomous robot control system for D&D operations. It is a purely PC based system that integrates the following components:

- A joint level control and a trajectory generator with a high level programming interface for Puma manipulators
- A 3D OpenGL-based hardware-accelerated robot simulator
- Video based and VR based operator interfaces
- Teleobservation programs
- Interfacing of different sensors
- Control of different robotic end-effectors

Hardware and software requirements of D&D systems are demanding:

- The joint level control task and the trajectory generator require *hard real-time performance*
- The GUI needs to integrate *VR and video techniques*. Hardware accelerated 3D rendering is required for the VR interface
- *Networking capabilities* are required in order to locate the robot control hardware remotely from the operator console. Networking is also required if a multiprocessor architecture is used to divide the work among multiple computers

Often the only solution: Integration of proprietary solutions and expensive hardware platforms.

Due to the availability of high-speed PC CPUs and real-time operating systems, the PC platform is able to fulfill all of the different requirements.

An entirely PC based system has the following advantages:
- The system is *cost-effective*, because PCs and their components are less expensive than proprietary controllers or traditional Unix workstations
- The system has a *simpler architecture*, since the additional effort to integrate completely different hardware components is not required
- The system is *more flexible*. To modify or extend the system, only a change to PC source code is necessary
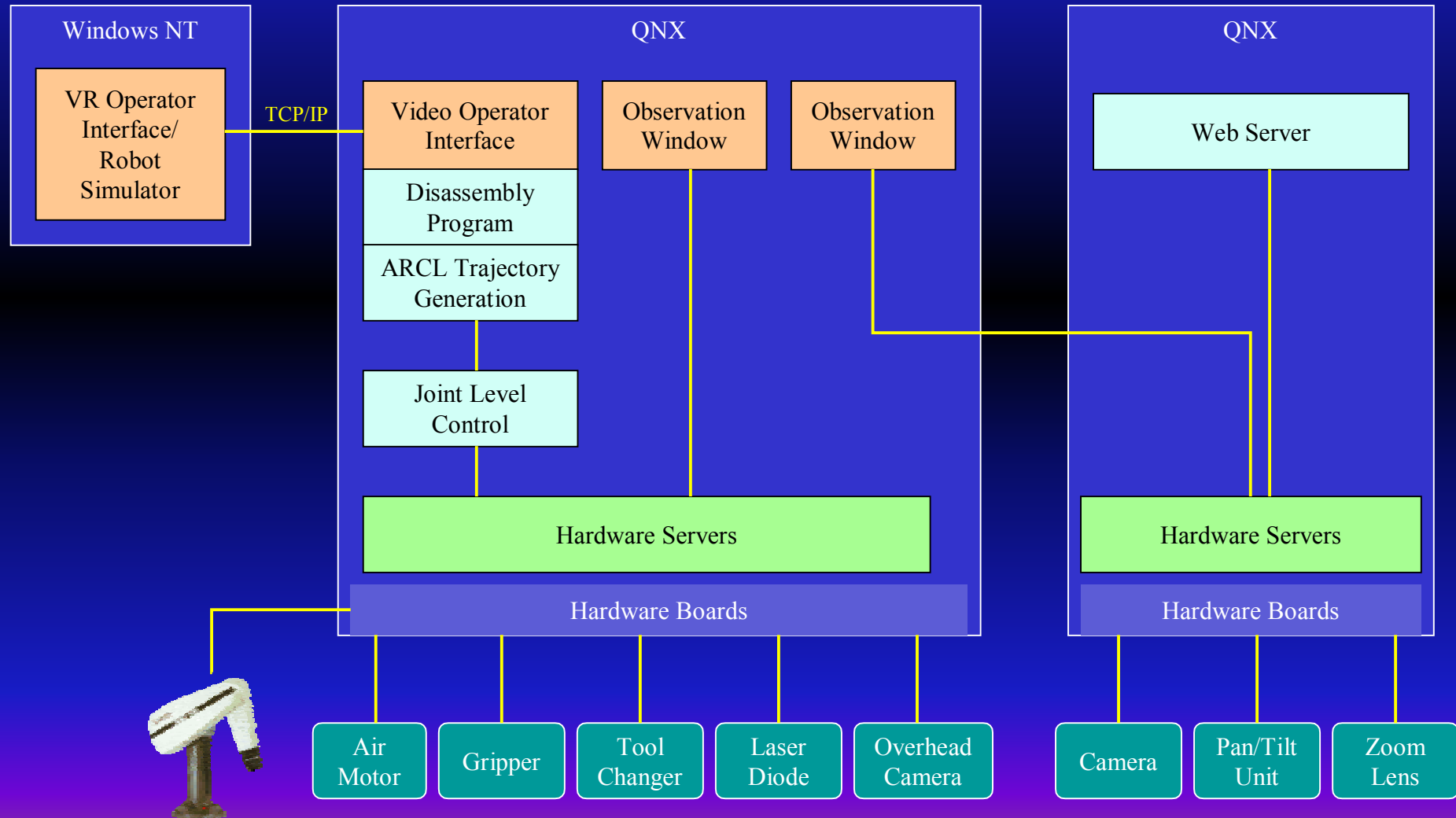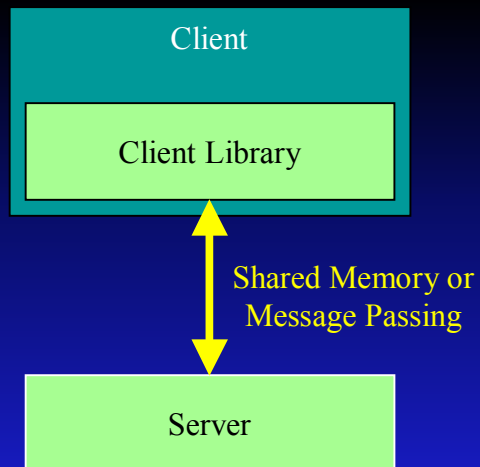
The Multitasking and Communication Architecture

- Need to run cooperating tasks of different priorities, some of them in real-time

- QNX 4: A real-time microkernel operating system

- Tasks are arranged in a client/server architecture

Client/Server Architecture

```
┌─────────────────────────┐
│        Client           │
│   ┌─────────────────┐   │
│   │ Client Library  │   │
│   └─────────────────┘   │
└─────────────────────────┘
            ↕
    Shared Memory or
    Message Passing
            ↕
┌─────────────────────────┐
│        Server           │
└─────────────────────────┘
```

Clients
- Use the client class library to communicate with the server
- Multiple clients can access the same server (hardware sharing)
- Generic clients are used to abstract from specific hardware
- Do not need to have root privilege

Servers
- Can't be crashed or delayed by clients (run at a higher priority)
- Can run on a different machine when QNX message passing is used
- Two types of servers:
    - Hardware Servers have root privilege. They access directly the hardware boards
    - Control Servers implement a control algorithm.

Joint Level Control

- Implemented as QNX program (flexibility)

- PD controller with friction and gravity compensation

- Zero gravity mode

- Sensor integration is possible (e.g. for force-based control or direct visual servoing)

High Level Control

- Port of ARCL (Advanced Robot Control Library) to C++ and QNX

- Trajectory Generation and Programming Interface

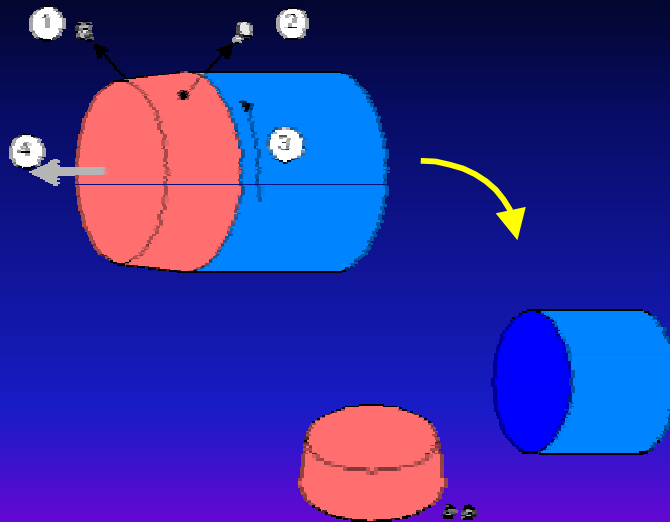- Communication to the robot simulator, using TCP/IP sockets

QRobot

The Disassembly Program

- Two bolts are unscrewed and removed

- A torch cut is simulated

- The cap is removed

- Via points are determined
  with the teachpendant

Robotic Utility Programs

- Calibration programs

- Teachpendant

QRobot

Operator Programs

Observation Windows

- Visual feedback of the operation

- The user can navigate by clicking in the image and set the zoom factor

- Multiple windows can be used with different cameras
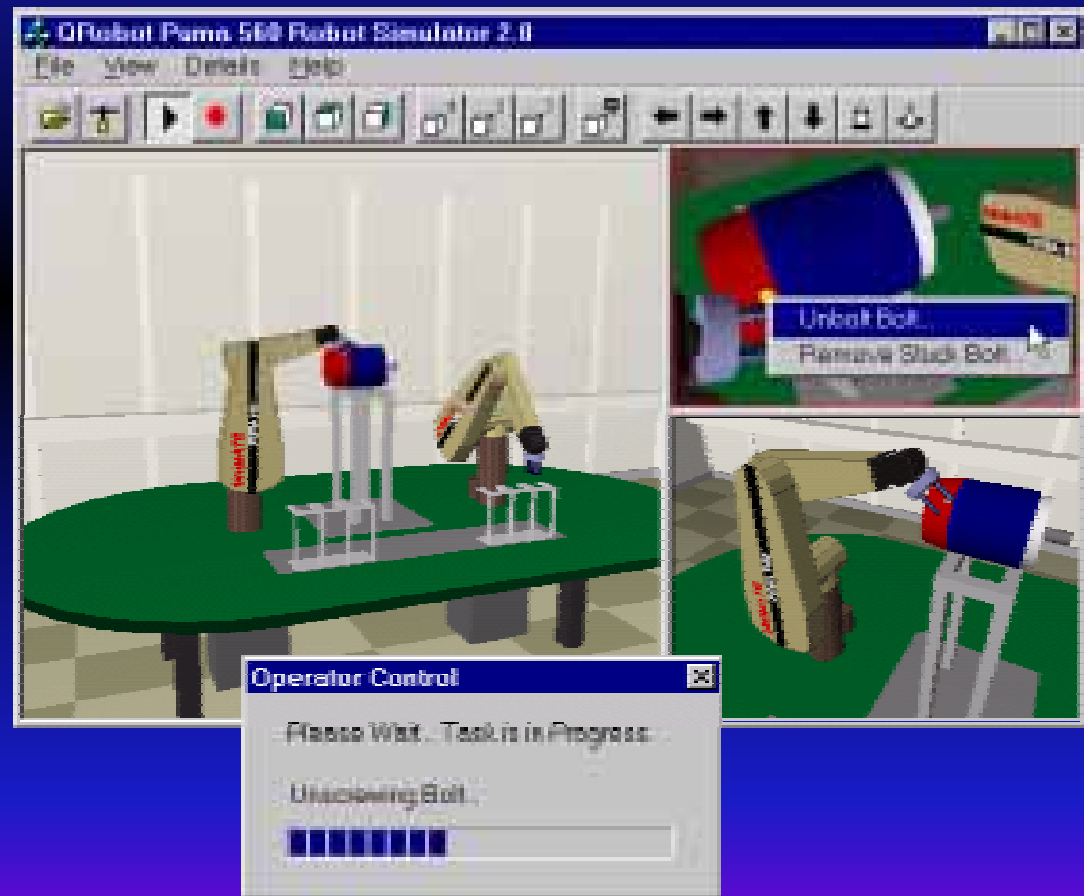
Video Operator Interface

- Even unskilled operators can control the disassembly operation

- Operations are started by moving the mouse cursor over parts in the image and selecting operations from pop-up menus

- Problems with hidden parts
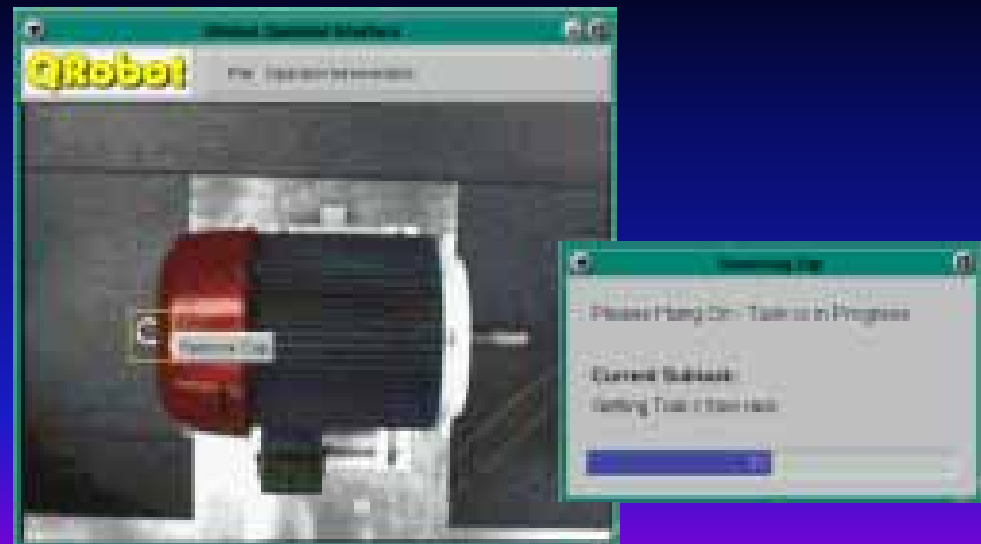
QRobot

VR Operator Interface/Robot Simulator

- Robot simulator and the VR operator interface are integrated in one Windows NT program

- Hardware accelerated 3D

- Operator can navigate in the 3D world

- Multiple views, also end-effector view

- Disassembly operations are started by clicking on parts of the 3D scene

- The VR operator interface sends commands over the TCP/IP connection to the disassembly program
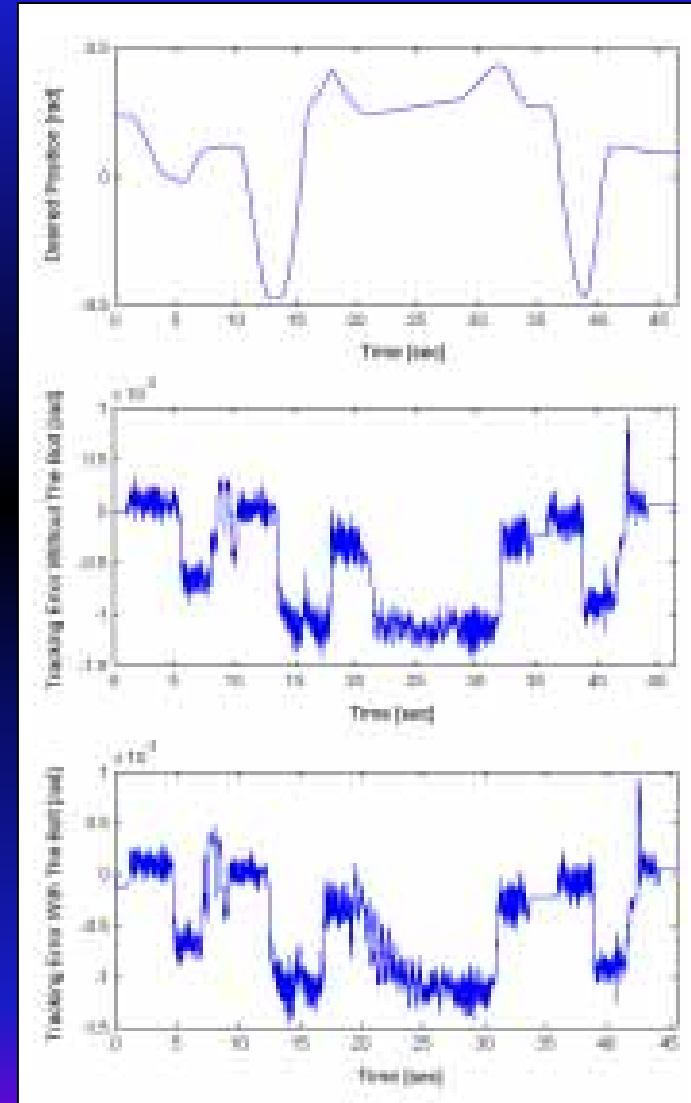
QRobot

Reliability of the Control

- Disassembly was repeated successfully many times

- Good tracking results, with and without load (lower graphs show errors for joint 3 while removing a bolt)

Usability of the System

- Easy to use operator interfaces

- Insufficient error recovery

Software Stability

- No crashes or controls falling behind, even with multiple operator interface windows open

Advantages

- The QRobot system demonstrates the feasibility of using a PC for the various tasks required in telerobotic semi-autonomous D&D operations

- Real-time control programs coexist with GUI programs on the same machine

- QRobot uses advanced video and VR techniques for easy-to-use operator interfaces

- QRobot's design allows flexibility in control algorithms and sensor integration

Disadvantages

- ARCL port is not a robust, general purpose solution

- Software architecture is not homogenous (different software components were assembled together)

- Not flexible in the operator interfaces

- Not easily expandable to different manipulators and end effectors

Current Research

QMotor Robotic Toolkit (RTK): A homogeneous, modular, and entirely object-oriented platform

- For the Puma 560 manipulator and the Barrett Whole Arm Manipulator (WAM)

- Based on the QMotor 3.0 low level control environment

- Only joint level (no kinematics, no Cartesian trajectory generation)

Future Research

Extend the QMotor RTK to a full scale robot control system

- Cartesian trajectory generator, forward and inverse kinematics

- Robot simulator

- More classes for tools, sensors and end-effectors