UCRL-JC-125880
Preprint

Tele-Robotic/Autonomous Control Using ControlShell

K. C. Wilhelmsen
R. L. Hurd
S. Couture

This paper was prepared for submittal to
American Nuclear Society Seventh Topical Meeting
on Robotics and Remote Systems
Augusta, Georgia
April 27-May 1, 1997

December 10, 1996

Lawrence
Livermore
National
Laboratory

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

TELE-ROBOTIC/AUTONOMOUS CONTROL USING CONTROLSHELL

Karl C. Wilhelmsen
Lawrence Livermore National Laboratory
P.O. Box 808, L-276
Livermore, CA 94550
(510)423-7919 : wilhelmsen1@llnl.gov

Randy L. Hurd
Lawrence Livermore National Laboratory
P.O. Box 808, L-276
Livermore, CA 94550
(510)422-7968 : hurd1@llnl.gov

Scott Couture
Lawrence Livermore National Laboratory
P.O. Box 808, L-276
Livermore, CA 94550
(510)423-4100 : couture1@llnl.gov

ABSTRACT

Lawrence Livermore National Laboratory has developed a tele-robotic and autonomous controller architecture for waste handling and sorting using their work in tele-robotics, autonomous grasping and image processing along with related work at ORNL. As a starting point, prior work from LLNL and ORNL was restructured and ported to a special real-time development environment. Significant improvements in the collision avoidance, force compliance and shared control aspects were then developed. Several orders of magnitude improvement were made in some areas to meet the speed and robustness requirements of the application.

The resulting controller architecture and functionality supports the following behaviors:

- master (tele-operational hand control) position to tele-robotic position control
- master position to tele-robotic velocity control
- force compliant tele-robotic control
- real-time collision avoidance
- autonomous motion control
- operator force queuing
- operator selectable tele-robotic behaviors

The control system is currently being used to integrate and control a six degree-of-freedom manipulator and a six degree-of-freedom force-reflecting master controller in a plant-prototypic waste sorting demonstration cell. The system has been in use for six months and is currently being successfully operated by personnel with little or no robotics background and essentially no training. This paper discusses, in more detail, the tele-robotic controller architecture and the system results to date.

I. OBJECTIVE

The Tele-Robotic/Autonomous Controller (TRAC) system at the Lawrence Livermore National Laboratory (LLNL) is part of the technology needed to bridge the gap between mature, bench-scale proven waste treatment technologies and full-scale treatment facilities. As an integrated facility, the technologies demonstrate the entire treatment sequence from solid waste receipt to output of the final waste form. Tele-robotics are to be used for container handling, waste characterization and preliminary sorting of the solid waste stream. In addition to meeting the specific design needs, the controller technology must be easily transferable and the development costs contained.

There have been many controllers developed which could meet some of these objectives, but their proprietary development environment impedes their transferability to other labs. The TRAC controller avoided these impediments by using an advanced, commercially supported development environment. To achieve software transferability, the TRAC controller development attempted to limit software development by using existing software to its greatest advantage. By developing a controller in this manner, the software transferability was enhanced.

Another goal of the TRAC system was to reduce the complexity of the system to the level where people with no robotics experience could operate the system effectively. Specifically, minimizing the number of interfaces to allow efficient operator training. This includes replacing the key board interface with a voice recognition interface and implementing a limited set of function buttons. Also, the tele-robotic interface was simplified to minimize the operators need to adjust his behavior for changing conditions.

II. HARDWARE

To minimize the development effort, commercial hardware was used whenever possible. The commercial hardware included the slave robot and master input systems. Two separate masters were tested: an RSI Research Ltd. six-degree-of-freedom (6-DOF) hand controller and a CyberImpact. These hand controllers were interfaced to a Titan III robot.

A Schilling[1] Titan III robot was installed in the Automation and Intelligent Systems laboratory at LLNL on October 29, 1995.[2] The Titan III is a 6-DOF hydraulic arm capable of lifting 250 lbs. In addition, the robot can move at 37 inches per second and has a maximum reach of 76 inches. The robot has a parallel jaw gripper capable of applying 540 lbs. force (see figure 1) and, as of October 1996, LLNL has upgraded their Titan to include Schilling's quick-change wrist. Also, as part of the wrist, a 6-DOF-force sensor from JR3, Inc. measures the robot end point forces in x, y, and z, and moments about x, y, and z.

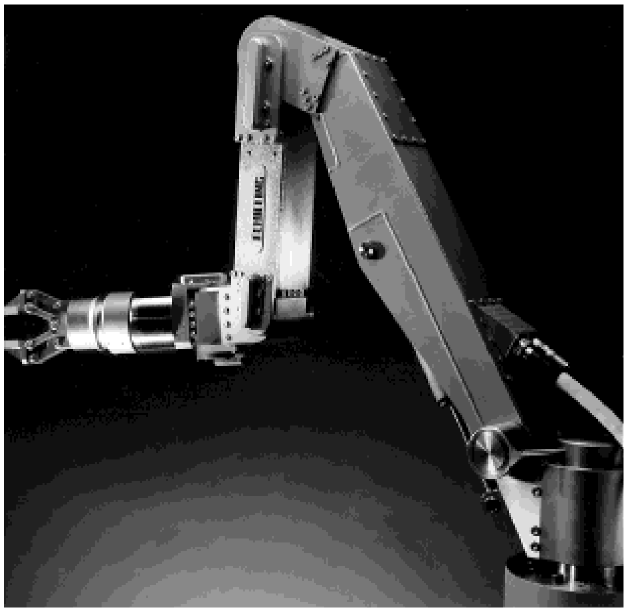

Figure 1. The Titan III robot used at LLNL.

The Titan III can be controlled with either an RSI Research Ltd. 6-DOF hand controller or a Cybernet force-reflecting 6-DOF hand controller. The RSI 6-DOF hand controller[3] is an inverted Stewart platform with two push-button switches, two trigger switches, and a deadman push-button switch (see figure 2). The Stewart platform has nine potentiometer monitoring positions which define the position and orientation of the grip handle.

The CyberImpact hand controller from Cybernet[3] is a small back-drivable robot which moves in 6 degrees-of-freedom: three linear positions (x-, y-, z-) and three attitudes (roll, pitch, yaw). The operator interface is through a grip handle with three push-button switches in addition to an analog displacement trigger switch and a deadman push button switch (see figure 3). An operator can use this motorized handle to generate position and force commands simultaneously (in x-, y-, z-) and tool angle (roll, pitch, yaw). Cybernet provides an MS-DOS C development environment for control system modification, reconfiguration, and interfacing. The standard CyberImpact hand controller supports serial communications at 19.2 KB.

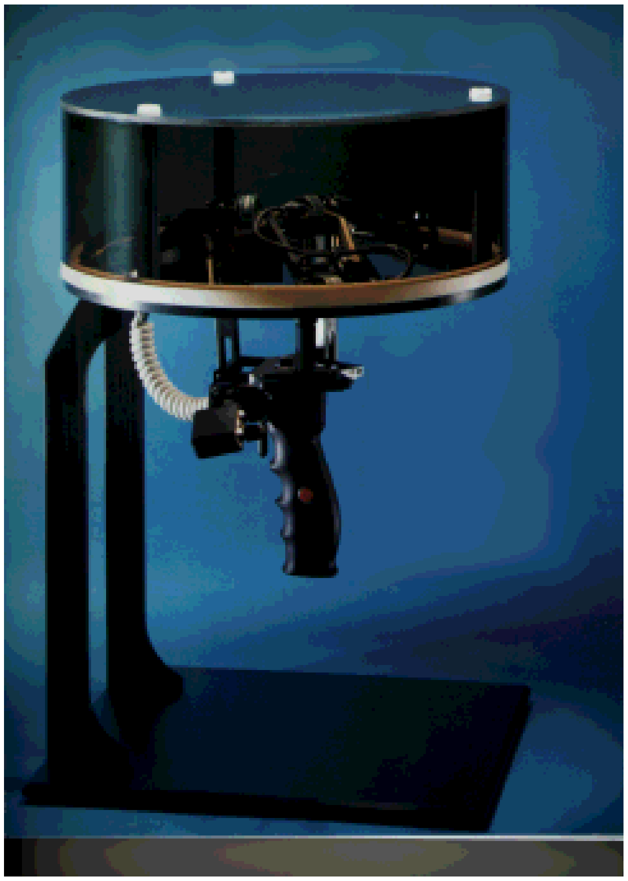

Figure 2. The RSI Research Ltd. 6-DOF hand controller used at LLNL.

III. INTRODUCTION

For the first design, we selected a development environment which would facilitate extensive software reuse and rapid prototyping. Of the development environments examined, ControlShell from Real Time Innovations (RTI)[4] was determined to be the most appropriate. ControlShell is a component-based, object-oriented system for real-time software development. Built on top of VxWorks, ControlShell combines a modular structure, powerful graphics-based 4GL[a], and integrated data management into a unique approach to real-time software.

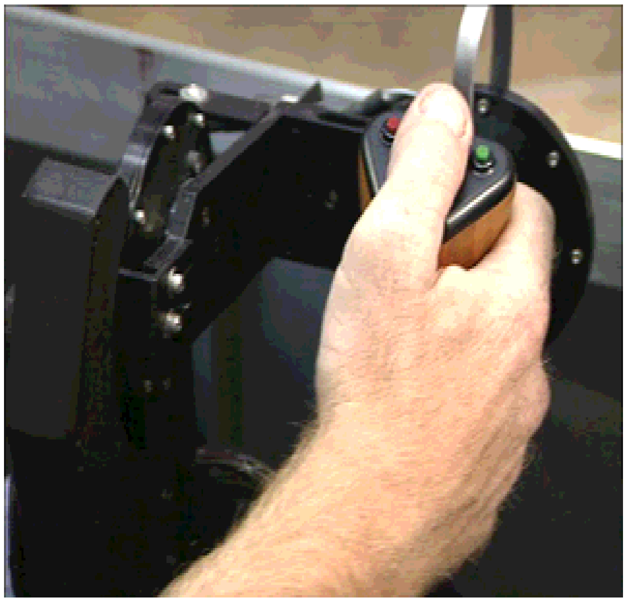

Figure 3. The CyberImpact hand controller used by LLNL.

ControlShell programs build complex systems from reusable code objects called "components". Graphical editors connect and combine these components into working real-time systems. New components are easily added with automatic C++ code generators. ControlShell takes advantage of object inheritance to allow development of complex components from simpler base classes; this makes components especially easy to reuse. ControlShell thus provides a framework that allows users, even at different sites, to share and reuse software.

ControlShell addresses both data-flow (sampled-data) systems and event-driven reactive systems. It excels when applied to complex systems combining these traits. It also features a graphical configuration manager, an object data service, a complete real-time matrix math package, an interface to RTI's data monitor (StethoScope), full integration with RTI's network data-sharing package (Network Data Delivery Service; NDDS), and an extensive library of generic and reusable components, including controllers, estimators, filters, signal generators, trajectory generators, device drivers, etc.

These features significantly reduced design time needed to develop the TRAC system. Components from different developers were loaded and used by simply transferring the source code files and installing them into the LLNL component libraries. Having a well-defined, commercially-supported graphical interface to connect components ensures easy integration of these new components. The configuration manager enabled components to be initialized and activated at will, giving the controller different behaviors on the fly. These different behaviors could be controlled through an expandable operator interface, or the controller could set the new behaviors from switch controls, or from an LLNL-developed voice command interface. The configuration manager also augments debugging. Specific problems could be tested by activating a limited subset of the system components to test for particular problems. Debugging could also be performed with StethoScope. StethoScope generates an X-window plot for any specified signal in real time. These plots were an invaluable tool for determining internal system behavior as the system operates.

The software reuse and rapid prototyping of ControlShell was demonstrated at the beginning of the TRAC controller development. After installation of the robot, rapid development of a prototype TRAC began. With only eight days devoted to controller development, a completed TRAC prototype was demonstrated which verified the software transferability and rapid prototyping capabilities of the ControlShell environment. This original controller was called the RSI TRAC.

As part of the RSI TRAC development, components from an ORNL Titan II controller[5] and a LLNL Puma 562 controller[2] were incorporated with new software to produce an autonomous and tele-operational controller. The tele-operational controller interpreted command inputs from the hand controller as positional commands or velocity commands to direct the robot gripper position. In addition, preplanned joint trajectories were stored for autonomous operations. These capabilities were selectively activated with the hand controller switches so the configuration could be seamlessly transferred between

[a] 4GL: Fourth Generation Language code generator.

the different modes. These capabilities are described in more detail in the technical approach.

IV. TECHNICAL APPROACH

After selection of the controller development environment, the desired behavior of the controller was defined and the needed tasks identified. The basic tele-robotic task was to sort mixed waste for a waste treatment facility. The waste was to be delivered in 55-gallon barrels where the barrel lid had been removed in a previous section of the facility. The waste barrel was to be placed in a barrel dumper and the waste poured onto a sorting table. The actual waste objects were placed in the barrels under pressure so it was expected the waste would occasionally need to be pulled loose with the robotic arm. Once the waste was on the sorting table, the waste was to be sorted into different categories as specified by the treatment facility needs. For some operations, singulation[b] tasks would need to be performed through glove ports by an operator, but the majority of work was expected to be performed with the robot.

The controller behavior was designed to assist the operator in sorting the waste. The controller was set up for the operator to pickup objects tele-robotically, then log information about the object in the waste tracking record. Information about the object included a visual description of the object, entered by voice. In addition, information from any sensor data, weight, x-ray images, etc., were entered electronically. Once the object had been grasped and described, the operator directed the controller to autonomously place the object in a specific bin. From the description and sensor data, the controller determined the correct category bin to deposit the waste. These autonomous tasks were important because they provided the operator with time to examine the waste pile in preparation for the next grasping operation. The complete controller with these capabilities is the TRAC system.

The TRAC system uses two VMEbus Motorola 68060's for control and collision avoidance, respectively. In addition, the TRAC system uses a Sun workstation for GUI interface and voice recognition. The set-up and display of the robot and work cell models were done in a graphical simulation package on an SGI Indigo Extreme. The graphical simulation package used was Robline from Cimetrix.[6] The SGI was also used for experimentation of real-time path planning.

[b] That is, to segregate a pile of objects into a string of single objects according to sorting criteria.

The TRAC system is an extension of the original RSI TRAC. An overview of the TRAC system is shown in figure 4. Each indicated component of figure 4 represents several lower level components. Each active component for a given configuration is run by each controller cycle. Since the CyberImpact includes capabilities of the RSI hand controller and has additional capacity, the remainder of the TRAC system discussion will focus on the final developed controller using the CyberImpact. The TRAC control structure has two basic configurations, tele-robotic and autonomous. The autonomous configuration uses the control component to replay preplanned joint trajectories to produce desired robot motions. The tele-robotic mode interprets input from the hand controller as a desired robot gripper-position command. These gripper-position commands are modified for compliance, joint limits, and collision avoidance and then converted to robot joint commands. Each subsystem will be discussed separately.

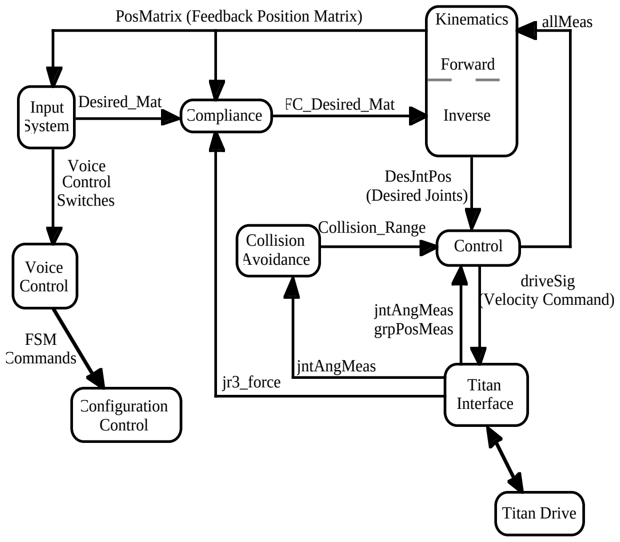

Figure 4. Overview of TRAC used to control the Titan III at LLNL.

A. Input System

The input system collects all mechanical input from the operator and converts it to the appropriate command signals. There are five switches on the CyberImpact, each performing a different function. The most important switch is the deadman switch on the CyberImpact handle. While the deadman is not active, the input system outputs a constant signal representing the current robot gripper command. This constant signal represents the operator's desire not to move the robot in a tele-robotic manner. When the deadman is activated, the user can start tele-operations of the robotic arm even under conditions where the arm is performing an autonomous operation. For the input system, with the deadman active, gripper position commands are output as the signal Desired_Mat. The Desired_Mat is a 4 x 4 matrix representing position and orientation of the desired gripper robot command with respect to its base frame.[7]

All master inputs are shifts from the initial robot gripper position. The current gripper position of the arm is generated by each control loop as the signal PosMatrix. To make these shifts as intuitive for the operator as possible, the shifts in gripper position are mathematically adjusted to match the operators view. For example, for tele-robotic operations from a camera view, when an operator moves the master to the right, the robot gripper moves to the right in the operator's camera view point. Matching the master movement to the operator's view point simplifies the control of the robot by causing the behavior of the robot to be the same for any selected view.

One operator interface difficulty overcome in the TRAC system was hand controller re-indexing. Re-indexing is the process of moving over large robot distances with a positional hand control master of lower motion range. The CyberImpact range of movement is 6 inches in x, y and z compared to a maximum of 12 feet for the work space of the Titan III. Therefore, in positional operation, to make large Titan movements generally requires several full range movements with the CyberImpact. After each full range movement, the deadman switch would need to be released, and the master moved to the opposite motion side to begin the next finite move. These repetitive movements are referred to as hand controller re-indexing.

A velocity mode was added to the input system to minimize master re-indexing. To activate the velocity mode, the CyberImpact is moved near the CyberImpact mechanical range-of-movement limit. When the CyberImpact is within half an inch of its mechanical range-of-movement limit, it has past the software range-of-movement limit. When the CyberImpact has past the software range-of-movement limit, the velocity mode control is activated for that axis, and a simulated spring behavior is implemented. The closer the hand controller is to the mechanical range-of-movement limit, the faster the velocity command of the robot in the direction desired. The velocity command peaks at the maximum speed of the robot. To notify the operator of the velocity mode status, a spring-like force is applied to the operator proportional to the velocity command being sent to the robot. The higher the velocity command, the more force to be applied to the operator's hand. Once the hand controller is moved within the software range-of-movement limit, the system seamlessly returns to positional control of the robot, and the velocity mode control and spring behavior are turned off.

B. Compliance

The input system signal Desired_Mat is passed into the compliance component. The compliance component modifies the user's desired input gripper position to limit the measured forces at the robot force sensor. The measured force comes from the jr3_force sensor in the wrist, and is provided to the compliance component as a 6 point array called the jr3_force. The compliance component restricts the force the robot imparts on the work cell, protecting the robot and work cell from damage. This capability can run during autonomous and tele-robotics operations to reduce imparted force while picking up or setting down objects.

The first function of the compliance component is gravity compensation. Since the wrist weighs over 20 lbs., the orientation of the wrist has a significant predicable effect on the jr3_force sensor. The gravity compensation predicts the forces from the weight of the gripper for a calculated orientation, and subtracts it from the measured jr3_force to produce an estimate of the external force on the robot wrist.

After obtaining an estimate of the external forces acting on the wrist, the forces in the world x, y and z were obtained by adding the relative constituent forces to their respective world axes. As the wrist changes orientation, the relative constituent transform is updated to compensate for the new orientation.

Once the world forces x, y, and z are obtained, the desired gripper position command, Desired_Mat, can be modified to limit these forces. If the world forces are below a threshold of 50 lbs., the Desired_Mat is not modified. If the forces exceed 50 lbs., the Desired_Mat is modified to relieve the force by integrating of the force error multiplied by a scalar and adding it to the Desired_Mat signal. This modified force is output as a 4 x 4 position and orientation matrix called FC_Desired_Mat. Since the measured forces can rapidly change due to the small robot movements, the speed of the robot must be slow enough for the robot to compensate for the fastest possible force transition. Limiting the speed of the robot will be discussed in the collision avoidance section.

C. Kinematics

The Kinematics component converts the input desired gripper command signal, FC_Desired_Mat, to the desired robot joint angles, DesJntPos. In addition, the Kinematics component converts the actual robot joint angles, allMeas, to a 4 x 4 feedback position and orientation matrix called PosMatrix. The generation of DesJntPos uses inverse Kinematics and the generation of FC_Desired_Mat uses forward Kinematics. These forward and inverse Kinematics are combined into the same component to ensure they remained a matched pair. If the forward and inverse Kinematics differed each time the input tele-operational system is activated, the robot would jump as it corrected for the difference. The core Kinematics and inverse Kinematics software were obtained from the ORNL Titan II development project.

Since the Titan III has finite reach, not all desired gripper positions are reachable. To handle these unreachable points, a behavior was implemented as part of the inverse Kinematics. A point may be unreachable in position, orientation, or both. The inverse Kinematics used in the TRAC reach as close to the desired position as possible and then attempt to match the orientation. In the reachable area of the robot, the position and orientation are located correctly, but as the desired gripper positions leave the reachable work space, the robot orientation command is modified to maximize the robots' reach.

D. Control

The control component takes the desired joint positions, DesJntPos, and the feedback, jntAngMeas and grpPosMeas, and calculates the appropriate velocity command, driveSig, with a control loop. The control loop uses proportional control with a limit range integrator and a simple lead lag compensator. In addition, the control component has multiple software joint limits. One software joint limit protects the mechanical stops of the robot. All the joints except the wrist roll have mechanical joint limits which may be damaged with repeated full velocity movements into the mechanical stops. The other joint limit is part of the collision avoidance system. This collision avoidance joint limit is dynamic and restricts the movement of the arm to a range that has been calculated to be safe.

E. Collision Avoidance

The collision avoidance system includes a full-arm model-based collision detection system. From the model of the arm, the collision avoidance system determines the safe operating space. The collision avoidance system has two main functions. First, the collision avoidance system must stop the arm from touching parts of the work cell it may damage during any operation. Second, the collision avoidance system must slow down the arm before the robot makes contact with the work cell in areas it may touch. The collision avoidance system operates on its own 68060 at a nominal rate of 40 Hz. At 40 Hz, the collision avoidance system is sufficient to protect the entire robot arm from collisions anywhere in the work space with minimal effects on performance.

The collision avoidance software also includes a path planner which has demonstrated the capability of calculating collision-free paths in real time. For the current Titan III work cell, the path planner calculated 90% of the anticipated paths in 0.5 seconds and the most difficult paths were determined in 7 seconds. Although the path planner was not optimized and was not designed to find every possible path, the results show useable path planning.

F. Titan Interface and Titan Driver

The Titan Interface Driver System interfaces with the Schilling VME-based Titan driver card and jr3_force sensor driver card. The drivers are interrupt-driven and control the reading of feedback position and force information from the driver cards. The drivers also communicate the desired velocity command to the Titan driver card. The resulting driver software was built from a combination of work from LLNL, Schilling and ORNL.

G. Voice and Configuration Control

The voice and configuration control system encompasses the user's ability to change the behavior of the TRAC system. Each voice command is interpreted by the voice recognition system and converted to an event. The voice recognition system is a commercially-developed speaker independent software package from Nuance Communications.[8] The voice recognition system runs on a Sun workstation and transmits data using RTI's NDDS ethernet communications system to the controller running on the VME chassis. The controller uses a state machine environment developed by RTI to control what specific components of the controller are active at a given time. The TRAC state machine is shown in figure 5.

The state machine is centered around the TeleOperation state. The voice recognition system controls the initialization of hydraulic power during setup and leaves the controller in the TeleOperation state. There

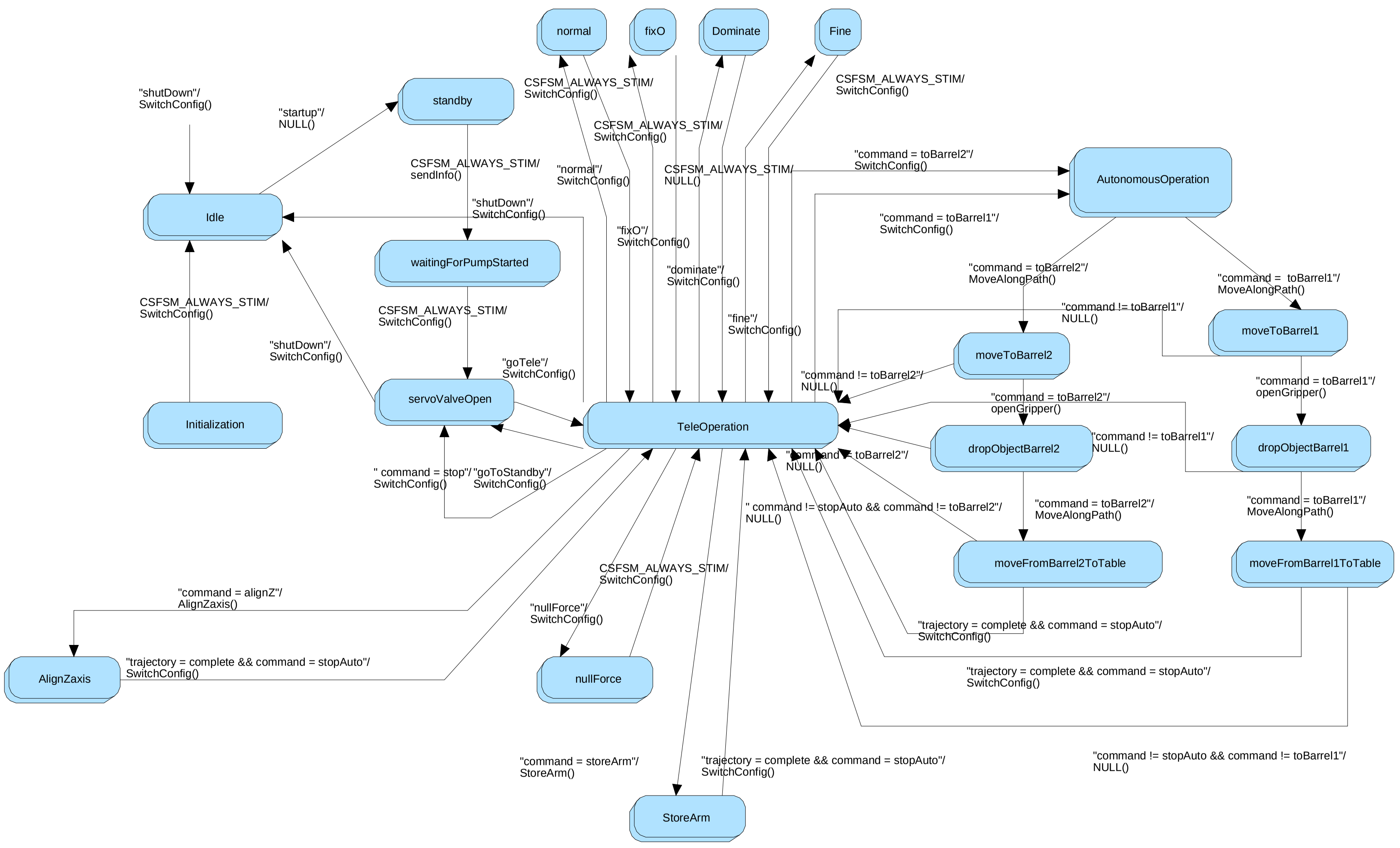
Idle
standby
waitingForPumpStarted
servoValveOpen
Initialization
TeleOperation
normal
fixO
Dominate
Fine
AutonomousOperation
moveToBarrel1
moveToBarrel2
dropObjectBarrel1
dropObjectBarrel2
moveFromBarrel1ToTable
moveFromBarrel2ToTable
AlignZaxis
nullForce
StoreArm
"shutDown"/ SwitchConfig()
"startup"/ NULL()
CSFSM_ALWAYS_STIM/ SwitchConfig()
"shutDown"/ SwitchConfig()
CSFSM_ALWAYS_STIM/ sendInfo()
"shutDown"/ SwitchConfig()
CSFSM_ALWAYS_STIM/ SwitchConfig()
"goTele"/ SwitchConfig()
" command = stop"/ SwitchConfig()
"goToStandby"/ SwitchConfig()
CSFSM_ALWAYS_STIM/ SwitchConfig()
CSFSM_ALWAYS_STIM/ SwitchConfig()
"normal"/ SwitchConfig()
CSFSM_ALWAYS_STIM/ NULL()
"fixO"/ SwitchConfig()
"dominate"/ SwitchConfig()
"fine"/ SwitchConfig()
CSFSM_ALWAYS_STIM/ SwitchConfig()
"command = toBarrel2"/ SwitchConfig()
"command = toBarrel1"/ SwitchConfig()
"command = toBarrel2"/ MoveAlongPath()
"command = toBarrel1"/ MoveAlongPath()
"command != toBarrel1"/ NULL()
"command != toBarrel2"/ NULL()
"command = toBarrel2"/ openGripper()
"command = toBarrel1"/ openGripper()
"command != toBarrel1"/ NULL()
"command != toBarrel2"/ NULL()
"command = toBarrel2"/ MoveAlongPath()
"command = toBarrel1"/ MoveAlongPath()
" command != stopAuto && command != toBarrel2"/ NULL()
"trajectory = complete && command = stopAuto"/ SwitchConfig()
"trajectory = complete && command = stopAuto"/ SwitchConfig()
"command != stopAuto && command != toBarrel1"/ NULL()
"command = alignZ"/ AlignZaxis()
"trajectory = complete && command = stopAuto"/ SwitchConfig()
CSFSM_ALWAYS_STIM/ SwitchConfig()
"nullForce"/ SwitchConfig()
"command = storeArm"/ StoreArm()
"trajectory = complete && command = stopAuto"/ SwitchConfig()

Figure 5. The finite state machine for the TRAC system.

are four modifiers of tele-operation which entail changing the scale of hand controller inputs to robotic movements. In addition, there are several autonomous functions. These autonomous functions are preplanned movements which are executed by a voice command and can be interrupted by pushing the deadman switch. The voice recognition system also logs the operator description of the object in the robot gripper and executes the classification of the object into the appropriate category. This sorting algorithm is modified for classification needs.

V. CONCLUSION

This TRAC system has been used for over six months by untrained people of various backgrounds. The collision avoidance system keeps these untrained operators from damaging the work cell while still allowing them to sort objects efficiently. A moderately trained operator can pick up a waste object, describe it with voice commands and autonomously place the object in its correct location every 17 seconds. The software structure has been successful enough that the TRAC system has been licensed to Schilling Development and parts are commercially available.

VI. ACKNOWLEDGMENTS

I would like to thank Brad Richardson and the ORNL Robotic and Process Systems Division for their support. The ControlShell components they supplied simplified the TRAC system development. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

VII. REFERENCES

[1] Schilling Development, Home page: http://www.schilling.com

[2] LLNL A&IS, Home page: http://www.llnl.gov/automation-robotics/

[3] Cybernet, Home page: http://www.cybernet.com/cybernet/index.html

[4] RTI, Home page: http://128.102.240.17/rtipage.html

[5] ORNL Robotics and Process Systems Division, Home page: http://www.ornl.gov/rpsd/rpsd.html

[6] Cimetrix, Home page: http://www.cimetrix.com

[7] John J. Craig, “Introduction to Robotics: Mechanics and Control”, second edition, Addison-Wesley publications, 1995.

[8] Nuance Communication, Home page: http://www.nuancecom.com/

Technical Information Department **• Lawrence Livermore National Laboratory**
University of California • Livermore, California 94551