S/5-37
44805

A Distributed Telerobotics Construction Set

James D. Wise
Department of Electrical and Computer Engineering
Rice University, Houston, Texas 77251

February 1, 1994

ABSTRACT

During the course of our research on distributed telerobotic systems, we have assembled a collection of generic, reusable software modules and an infrastructure for connecting them to form a variety of telerobotic configurations. This paper describes the structure of this "Telerobotics Construction Set" and lists some of the components which comprise it.

1. Introduction

The Universities Space Automation and Robotics Consortium (USARC) was formed in 1989 to promote research into robotics and telerobotics for space-based applications. It consists of five universities (Rice University, Texas A&M University, the University of Texas at Arlington, the University of Texas at Austin, and the University of Texas at El Paso) in conjunction with NASA's Johnson Space Center. One of our major areas of research has been teleoperation over long distances, typically involving long delays, low bandwidth, and possible loss of data. In the course of this research we have assembled an experimental telerobotics system which connects robots and control sites at the universities and JSC.

In order to conduct research efficiently in this distributed environment we have built our system using a construction set approach: it consists of a number of independent modules which may be connected together in any (reasonable) configuration. This independence allows each group of researchers to concentrate on their own area of expertise and have their work immediatly integratable into the overall system.

2. The Construction Set Approach

Figure 1 is a block diagram of one of our basic configurations for teleoperation: a robot worksite with two remote operator control sites. It consists of a number of clearly defined functional blocks communicating via streams of data packets using a few well defined formats.

Our experience with these diagrams has been that simply drawing the modules is enough to define the system: the interconnections largely draw themselves. Each module accepts a particular type of input data and produces another type as output. By matching the data types, we form the required connections.

This idea of letting the data define the connections, rather than imposing them externally gives rise to what we call a *data centered* approach to modularity using *undirected messages.* In such a system, consumer processes declare the types of data they are interested in and producer processes declare the type of data they distribute. Based on this data centered approach, we have developed a data exchange mechanism that we call the Telerobotics Interconnection Protocol, or TelRIP.

TelRIP is described in detail elsewhere [1]. Here ia a brief summary of the key points:

- It supports a uniform interface to a variety of underlying communications media.
- Processes communicate by exchanging Data Objects.
- Byte order and format translation are handled automatically.
- Undirected messaging provides data distribution based on properties of the Data Objects.

The last item is the key to the successful modularity of our system. Rather than data being sent to a specific

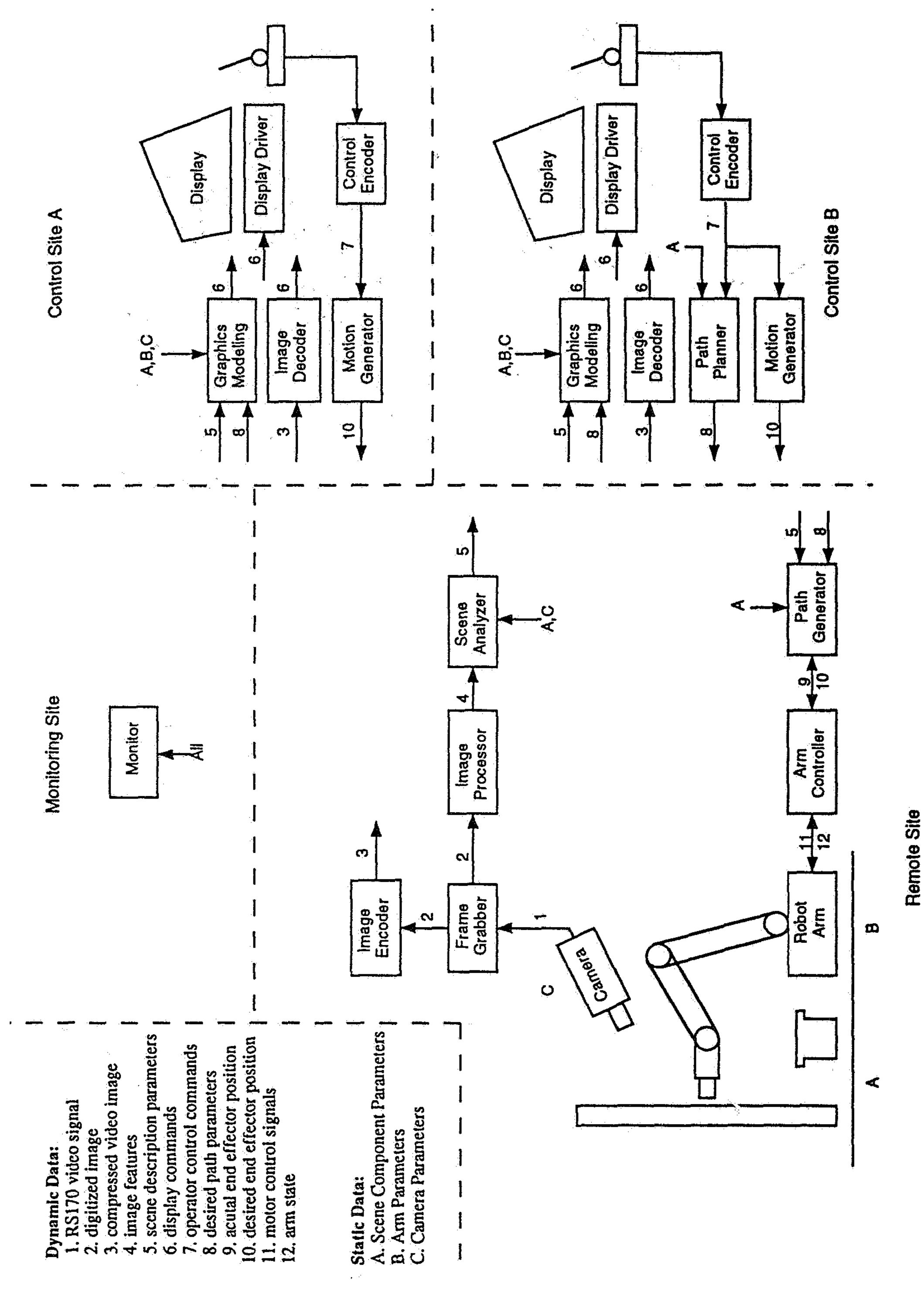

Figure 1: A distributed Telerobotics System

destination, resulting in a "hard wired" connection, modules specify the types and other characteristics of the data they wish to receive. An "automatic connection" will be made to whichever other modules provide the required data. Since this connection is dynamic, modules may be added to or removed from the configuration at any time with minimal impact on the rest of the system. This gives the system a high degree of both flexibility, and robustness.

Conceptually, our construction set consists of three components:

- A set of Functional Modules which perform fairly narrowly defined, hopefully generic, functions.
- A standard, minimal vocabulary of data objects which encapsulate the information communicated among the modules.
- A data exchange mechanism which delivers the data objects from the producers to the consumers, that is, it forms the connections between modules.

Each of these conceptual components has a physical realization:

- The functional modules are implemented as programs, or "Processing Modules"
- A common vocabulary is achieved by defining standards for data objects which are used by all programs. These standards define both the format of the data, and the semantics.
- Data exchange among modules is handled by TelRIP.

3. System Components

TelRIP provides the framework, or building board, for our construction set. The remaining components are a set of programs to implement the required functional modules, and a common data object vocabulary.

3.1. Data Object Vocabulary

The atomic unit of communication between modules is the TelRIP Data Object. A data object is similar to an object oriented programming object in that it contains a self identifying data record. TelRIP data objects lack an explicit class hierarchy and imbedded method code. However, they incorporate additional ancillary information, or *Properties* which are used in classifying and directing objects.

These properties include:

- A timestamp, which gives the time at which an object was distributed. TelRIP maintains synchronization of timestamps among systems.
- A source identifier or *address*.
- An intended destination address.
- Qualifying properties (e.g. compressed, operator generated, etc).

Although there is no explicit class hierarchy, it is convenient to group the object types into classes. Table 1 is a brief description of the major object types used in our system.

3.2. Functional Modules

Like the data object types, the functional modules may be grouped into classes:

Class	Functionality
Operator Interface	
	Hand Controller Interface
	Controller Semantics
	Simulation Display
	Video Image Display
	Control Panels
	Interoperator Communication
	Audio Response
Robot Interface	
	one for each type of robot
Motion Control	
	Motion Semantics
	Transform Generator
	Force Control
Video	
	Frame Grabber
	Image Compressor
Monitoring	
	Human Performance
	System Analysis/Debugging
Utilities	
	Video Recorder
	Audio Recorder

Table 2: Functional Modules

Class	Type	Function	Contents
sensor	IMAGE	transmit a video image	image size pixels
	FORCE_TORQUE	transmit force and torque data	f_x, f_y, f_z n_x, n_y, n_z
control	COMMAND	send a general command	command code optional arguments
	HAND_CTLR_STATE	transmit raw hand controller data	axis values button states
	ERROR	return an error condition	error code
motion	JOINT_STATE	specify joint angles	$\theta_1, ..., \theta_n$
	JOINT_TRAJECTORY	specify a path in joint space	array of joint states
	CARTESIAN_STATE	specify position and orientation of end effector	x,y,z roll,pitch,yaw
	CARTESIAN_TRAJECTORY	specify a path in cartesian space	array of cartesian states
	HOMOG_TRANSFORM	specify viewpoint to effector transformation	matrix coefficients
	MOTION_SEMANTICS_CONFIG	specify mapping from hand controller to end effector	axis modes reference frame
parameters	HAND_CTLR_DESCRIPTION	characterize a specific hand controller	gains axis assignments
	CAMERA_DESCRIPTION	describe camera view	camera location field of view
communication	TEXT_MESSAGE	printed interoperator communication	text characters
	AUDIO_RECORD	spoken interoperator communication, audio command response	sample rate audio samples

Table 1: Data Object Types

3.3. Processing Modules

The functional modules are implemented by a set of programs or *Processing Modules.* Table 3 lists the major programs in our construction set. The origins of some of the program names are, as they say, lost in antiquity.

A few of these will be described in more detail:

TSM Although the automatic interconnections provided by data type matching are adequate for most single robot, single controller scenarios, when multiple instances of a particular functional module are present in a configuration, it is necessary to be able to distinguish them. This is done using the address property mechanism: each instance of a duplicated module has a unique value of the address property. This property may be specified either as a physical address or a functional address.

To simplify the dynamic management of these properties in situations where control is passed from one robot or controller to another, we have developed a program called the Telerobotics Session Manager (TSM). TSM allows the connections among modules to be displayed and modified while the configuration is running.

VCP Currently the largest class of modules is what we call the Operator Interface Components. By implementing these as a set of narrowly scoped components rather than a single monolithic entity, we can construct a Virtual Workstation (VWS), where the operator interface for a variety of configurations can be assembled on a single physical workstation.

We have extended the system independence concepts of TelRIP to the construction of operator interfaces with a system called the Virtual Control Program (VCP). VCP allows a user interface to be defined in a very straightforward way, independent of the system on which it will be displayed. This allows a programmer to write a single program with a graphical user interface which will run on Open Look, Motif, Windows, or any other window system without having to know how to program any of them.

Interfaces are defined in terms of commands, parameters, adjustments, etc. and are implemented using buttons,

Program	Function
spaceball	convert device specific codes into generic hand controller objects
hc_robot	generate robot control objects from hand controller objects
TDM	graphical simulation display
tdm_sim	interface TDM to TCS
displayx	display a video image on a workstation screen
cmdks	generic command line interface
vcp	generic graphical user interface
chat	printed interoperator communication
phone	audio interoperator communication
audioresp	voice response to commands
remote_*	translate robot control objects into device specific commands
force_dsp	display forces and moments
motion	motion semantics controller
tfb	tool frame builder
pcgrab	digitize one frame of video
vcompress	compress a frame of video for long haul transmission
sggrab	grab and compress video with remote pan and zoom
tsm	Telerobotic Session Manager
rmonitor	gather human performance data
xmon	network monitor
vcr	video recorder
dat	audio recorder
siggen	audio signal generator

Table 3: Processing Modules

text items, sliders, or whatever appropriate constructs are available from the target windowing environment. A system dependent program (VCP main) runs on the target display and communicates with the client Processing Module to generate the GUI display and updates and commands between the PM and the display.

4. Demonstration Configurations

We have built or are currently building a number of demonstration configurations using these components. These include:

4.1. Manual Controller Performance

A standardized task was performed by a number of operators using several different manual controllers. The data streams were analyzed to determine the human performance parameters of each session.

4.2. Multiple Robots, Multiple Controllers

This configuration demonstrates the interoperability and dynamic reconfigurability of our system modules. Two robot worksites (A and B) and two robot control sites (Y and Z) performed the following scenarios:

- A task at site A was begun by controller Y. Midway through the task control was switched to site Z, who completed the task.
- Controller Y performed a task at site A. Using the same control components, he performed a second task at site B.
- Controller Y performed a task at site A simultaneous with controller Z performing a task at site B.

4.3. Extended Teleautonomous Control

Modules intended to increase the performance of an operator controlling a remote manipulator under adverse circumstances (delays, restricted visibility, etc) will be added to the basic configuration. These include: Time and position clutches, time brake, voice control, and force control.

4.4. Network Stress Testing

The reliability and robustness of the system are examined by increasing the stress (typically an increased data rate) on various components.

4.5. Workload Analysis in Shared Human-Autonomous Tasks

Real-time measurement of human workload and overall system performance will be made on a set of tasks having varying degrees of operator involvement and autonomous control.

5. Acknowledgments

This work has been supported by NASA under grant NAG9-461 and by the State of Texas under TATP grant 999903-267.

References

[1] J.D. Wise and L.A. Ciscon, "TeleRobotics Interconnection Protocol Operating Manual," Tech. Rep. 9103, Universities Space & Automation Research Consortium, 1991.