A Graphical, Rule Based Robotic Interface System

James W. McKee
University of Alabama, Huntsville
Box 212, RI-A4
Huntsville, Alabama 35899

John Wolfsberger
NASA/MSFC
EB 42
Huntsville, Alabama 35812

ABSTRACT

The ability of a human to take control of a robotic system is essential in any use of robots in space in order to handle unforeseen changes in the robot's work environment or scheduled tasks. But in cases in which the work environment is known, a human controlling a robot's every move by remote control is both time consuming and frustrating to the human.

A system is needed in which the user can give the robotic system commands to perform tasks but need not tell the system how to perform the tasks. To be useful, this system should to be able to plan and perform the tasks faster than a telerobotic system. The interface between the user and the robot system must be natural and meaningful to the user.

This paper describes a high level user interface program under development at the University of Alabama, Huntsville. The authors propose in this paper a graphical interface in which the user selects objects to be manipulated by selecting representations of the objects on projections of a 3-D model of the work environment. The user may move in the work environment by changing the viewpoint of the projections.

The interface uses a rule based program to transform user selection of items on a graphics display of the robot's work environment into commands for the robot. The program first determines if the desired task is possible given the abilities of the robot and any constraints on the object. If the task is possible, the program determines what movements the robot needs to make to perform the task. The movements are transformed into commands for the robot. The information defining the robot, the work environment, and how objects may be moved is stored in a set of data bases accessible to the program and displayable to the user.

Introduction

The graphical user interface, to be described in this paper, is part of a project to develop a software system that will enable users to control robots from a task level instead of having to either teach the robot the path or write programs in the robot's language or when using simulation programs specify points in the works space and actions to be performed [3].

There are two objectives of this project which have a strong influence on the requirements of the graphical user interface. The first objective is to divide the software into functional modules such that new versions of any module may be "plugged in" and the system tested. The second objective is to be able to incorporate into the system the knowledge and expertise that a person usually needs to have to create programs for the robot.

We have divided this project into four modules: user interface, path planning, environment calibration, and robot code generation.

The user interface module will contain the graphical descriptions and all the knowledge, rules, and constraints about the robot and the work space. The function of a robot is to move objects. The user interface is the means by which the user tells the system which objects are to be moved and where. The graphical user interface being presented in this paper is only one of many possible user interface modules.

From the robot task requirements and the given geometric and dynamical constraints on the robot motion, the path planning module will define a path in the robot work space that will avoid collisions and satisfy the constraints on the joint dynamics.

The environment calibration module will allow the robot to calibrate itself to a task board or other objects in the work space. The module will also allow the system to verify that what the robot "sees" in the work space is what it should see.

The robot code generation module transforms the internal motion representation data into movement commands for a particular robot.

Graphical User Interface

The user interface module has been divided into three projects: object definition user interface (ODUI), object movement user interface (OMUI), and rule based task planner (RBTP). The ODUI program allows the user to create new objects and enter the objects into the system data base. The

OMUI allows the user to move around in the work space and select objects to be moved. Once the user selects an object and its destination, the RBTP determines if the object can be moved and if so makes a list of fixed and flexible paths.

To support the graphics requirements on this project, the user interface software is being developed on a Silicon Graphics 3020 graphics station. The Silicon Graphics computer is connected to a PUMA 562 robot by a RS232 line and running the DDCMP protocol. This will be used as the hardware configuration for system tests.

Object definition user interface

The purpose of the ODUI software is to allows a user to create objects in the system. An object is a set of data that contains the following types of information: name, geometric description, physical attributes, movement and positional constraints, and construction list.

The objective of this module is to make it easy for a user to create the graphical description of objects that will form the environment or are to be moved. The graphical information will be used by the OMUI, the path planner module and the environment calibration module. This is also how the user creates the knowledge base of the physical attributes and movement constraints of the object that the expert system will need to determine if and how objects are to be moved. The software being developed is intended to be a flexible framework by which to store the knowledge data. This software itself does not interpret or process any of the knowledge data.

The user interface consists of levels of mouse selectable pop-out menus buttons. The top level menu allows the user to examine objects, edit objects, create objects, or delete objects.

Objects are essentially data bases. To examine an object, the user views the various lists of data in the objects data base. This could be by viewing projections of the graphical representation of the object or by viewing, in text, format the contents of the other list of the object.

Each object is created with its own local coordinate system. All references to the position of the object are with respect to the origin of the object's coordinate system. The object's geometric description and positional constraints are defined in this coordinate system.

Each object has a unique name. This is the key by which the object is referenced when used to form another object or when moved. Any object can be used as a template to create

copies of itself. The user must supply a unique name for the new object when creating a new object from a template.

One or more objects may be combined with or without added graphics to create a new object. When an object is incorporated into a new object, the name of the object being incorporated and its rotation and translation are placed in the new object's construction list.

The user through a process of creating objects and combining objects builds up the robot, the environment and the robot work space. Starting from the top and working down, the robot work space contains everything and is an object which is composed of two objects: the environment and the robot. The robot is an object composed of objects that are the links of the robot and a data base of the kinematic equations for the links.

The environment is composed three types of objects: movable objects, receptacle objects, and the task board object. Movable objects are objects that have been selected by the user to be movable by the robot. Receptacle objects are objects in which or on which movable objects may be placed. Movable objects may only be moved from one receptacle object to another receptacle object.

The task board is everything else in the environment. The task board object is needed for the geometric description it generates in the path planner module and the environment calibration module and to give the user a geometric feel for where the other objects are located.

Receptacle objects contain the information about how and where movable objects may be placed. For example, a receptacle object could be a hole. In this case the hole would carry the information about the size and shape of a movable object that could be placed in the hole and the fact that the object must be inserted. Another type of receptacle object would be a pad. A pad would contain the information that a movable object could be placed on it. The pad could also contain the information about the orientation of the movable object when it placed on the pad. More than one receptacle object may be placed at the same geometric location, each designed for a particular class of movable objects.

Graphical descriptions are created by the user "drawing" simultaneously in three orthogonal projection windows. The size and position of the windows on the monitor is user controllable. Descriptions are created by combining volumes and surfaces. There is a set of primitive volumes that include cones, cylinders, and rectangles. These primitives may be stretched to whatever size is needed. Surfaces are

planar polygons. The user combines these volumes and surfaces to create graphic descriptions of the objects.

Object movement user interface

The objective of the OMUI software is to allow the user to select objects and indicate where they are to be moved in the robot's work space. The user sees on the graphics monitor a two-dimensional projection of the three dimensional work space of the robot.

The user may translate, rotate, and zoom the work space. The user may select one of these three modes of moving the work space by mouse selectable menu buttons on the side of the screen. Once an option has been selected, the user controls the direction of motion of the work space by pressing one or more of the buttons on the mouse and the rate of motion by the motion of the mouse.

The cursor mode is another mouse selectable menu button. Once the cursor mode is selected, the user may select what information about objects will be displayed as the curser moves over their projection. The user may turn on object highlighting, object name display, and/or object data display. The highlighting switch causes the movable objects and receptacle objects to be highlighted when the cursor moves onto them. The object name switch causes the name(s) of the object(s) under the cursor to be displayed. The object data display switch allows the user to examine any of the knowledge data for selected objects.

Once the work space is in the desired orientation and magnification, the user may move a curser around on the surface of the projection. After the cursor mode is selected, the cursor is moved by pressing the right button on the mouse as the mouse is moved. A movable object is grabbed by clicking the middle button of the mouse when the cursor is on the object. Only movable objects may be grabbed. The destination is selected by moving the cursor onto a receptacle object and clicking the left mouse button. The object is dropped if the left button is clicked anywhere else, in which case there is no change in the geometric configuration.

At the present time only one object movement can be selected in a session. A future enhancement will be to be able to create lists of object movements that are to be carried out in succession with the environment being updated after each object movement.

Rule based task planner

Once a movable object has been grabbed and a destination receptacle object selected, the RBTP must determine if it is possible to move the object. The RBTP generates a set of knot points for the geometric path of the movable object. A knot is a pose (x, y, z, roll, pitch, yaw) through which the movable object is to pass. Connecting the knot points are fixed and flexible paths.

A fixed path is a path on which the pose of the movable object is defined along the whole path. On a flexible path the pose of the object is defined only at the end points. The RBTP will also collect from the knowledge base of the object a set of any applicable constraints on the movement of the object. Constraints could be items such as a maximum acceleration, allowable tilt angles on the object, maximum gripping pressure, etc.

From the information in the data bases of the source and destination receptacle objects, the RBTP creates the set of knots and the fixed paths. For example, assume the source receptacle object was a pad, the movable object was a peg, and the destination receptacle object was a hole. Then the path of the object could consist of a fixed path, a flexible path and a fixed path. The first fixed path would be the path to pick the peg up off the pad. The second fixed path would be the path to insert the peg into the hole to the desired depth. And the flexible path would be between the end of the first fixed path and the start of the second fixed path.

For each flexible path, the path planning module will create a near optimum, collision-free path that does not violate any of the dynamic constraints on the joints of the robot or any of the constraints on the movement of the object.

Since we are building a software system, the expert system software must be capable of being incorporated into the overall software. Therefore, it was decided to have the expert system run on the Silicon Graphics computer. Although LISP is available on the Silicon Graphics, it was decided to use CLIPS. CLIPS is a forward chaining expert system shell written in C. The source code for CLIPS is commercially available [1]. Harrington [2] has compared CLIPS, LISP, Prolog, and OPS5 and has concluded "Because of its embedability, its expandability, and its smaller size, CLIPS would be the better selection for embedding low-level ES capability within a control system".

Conclusions

This paper has presented an overview of the software being developed at the University of Alabama, Huntsville to enable a user to control a robot from a task level. The main emphasis of the project is to develop a set of software modules that work together as a system. This presentation has concentrated on the user interface portion of the project and how the requirements of the overall system have affected the design of the user interface portion.

Acknowledgements

Research for this paper has been supported in part by a grant form the Science, Technology and Energy Division of the Alabama Department of Economic and Community Affairs. However, any opinions, findings, conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of ADECA and the State of Alabama.

References

[1] Giarratano, J. C., CLIPS User's Guide, CLIPS Reference Manual, COSMIC Program # MSC-21208, 382 E. Broad St. Athens, GA, 30602.

[2] Harrington, J. B., "CLIPS as a Knowledge Based Language," Third Conference on Artificial Intelligence for Space Applications, Huntsville, November 2-3, 1987, pp.33-40.

[3] Mckee, J. W. and Wolfsberger, J., "High Level Intelligent Control of Telerobotic Systems," Conference on Space and Military Applications of Automation and Robotics, June, 1988, Huntsville.