Application of Tcl/Tk for a Robotic System

Antonio C. Leite and Fernando C. Lizarralde

***Abstract*— This paper addresses the application of Tcl/Tk language to develop a graphical user interface (GUI) for a robotic system, which consists of a robot manipulator, a CCD camera and a force sensor. The GUI is used to monitor the interaction forces, set the camera-robot system parameters and execute user-developed routines in order to improve the performance of the experimental tests.**

I. Introduction

In the last decades, robotic systems have been used to perform several tasks in industries, hazardous environments and research centers. Following this tendency, medical applications for multifingered robotic hands were also proposed to execute minimally invasive surgery based on concept of virtual reality and telepresence. In some applications, one can use vision sensors (e.g., cameras) to control the position of the robot end-effector relative to a target object or a set of target features. This approach is commonly referred to as visual servoing. In this framework, graphical user interfaces (GUIs) have been developed to monitor and set the system parameters in order to improve the configuration and execution of the interest task.

One of the current trends in software development is the use of portable script languages like Perl, Phyton or *Tcl/Tk* [1]. Although interpreted languages are generally slower than a compiled, Tcl/Tk in particular offers a lot of advantages for the developer as simplicity, source code freely distributed, developers community for assistance and guidance, etc. Moreover, Tcl is relatively easy to use and learn, providing most of the features one would expect from a general purpose programming language [2]. Other significant advantage of Tcl is the *extensibility*. When some application requires adding commands beyond to those that Tcl core provides, it is possible to write new Tcl commands using C language [3]. Furthermore, GUIs written using Tcl/Tk typically require significantly less code than an equivalent interface, for instance, written in C/C++ language. Graphical builders as Visual Tcl and Visual Gipsy make rapid and easy the implementation and prototyping of user interfaces [4].

This work addresses the application of Tcl/Tk language to develop a graphical user interface for a robotic system [5], which consists of a robot manipulator, a CCD camera and a force sensor. The pratical task involves the visual tracking of a reference trajectory, while the arm tip exerts a controlled contact force on a smooth constraint surface. The GUI is used to monitor the interaction forces, set the camera-robot system parameters and execute user-developed routines in order to improve the performance of the experimental tests.

The authors are with the Department of Electrical Engineering - COPPE, Federal University of Rio de Janeiro, 21945-970, Rio de Janeiro, Brazil. Email: [toni, fernando]@coep.ufrj.br

II. Motivation

In the considered robotic system, there are three main source codes written in C language used for a task execution: the first module performs the capture of the robot image (acquiring and processing) through a CCD camera and a frame-grabber. The second module reads the interaction forces measured by the force sensor and shows them in the screen as a vector. Finally, the last one performs the visual servo control of the robot manipulator. The main difficulties encountered to conduct the experimental tests are: (1) define the reference trajectory in the camera point of view; (2) set the camera-robot system parameters using a configuration file; (3) view the robot trajectory after the experiment has been performed; (4) monitor the values of interaction forces acting on the robot end-effector. Then, the key idea is to implement these source code as *new Tcl commands* written in C language and overcome the difficulties mentioned above by the development of an *ad hoc* graphical user interface.

III. Graphical User Interface

The GUI, named *VServo*, is composed by the following toplevel windows: VServo, Frame-Grabber Parameters, Control Parameters, Reference Trajectory, Showforces and Text Editor for handling files. The configuration parameters are initialized through default values inserted in the GUI code and the values entered by the user are saved in a configuration file.

A. VServo

In this window, one can execute the control algorithm for the robot arm and view the log of the system in a scroll widget text (Fig. 1). Moreover, one can open the others toplevel windows, stop the task execution or close the application.

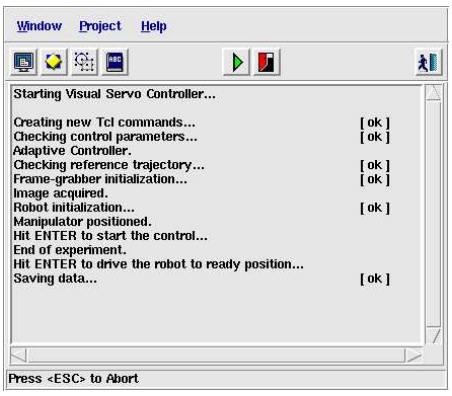

Fig. 1 - VServo window

B. Control Parameters

This window allows the user to set the following parameters: control law, width of subwindow for image processing, sample time, gain of the robot controller board, control and custom parameters (Fig. 2).

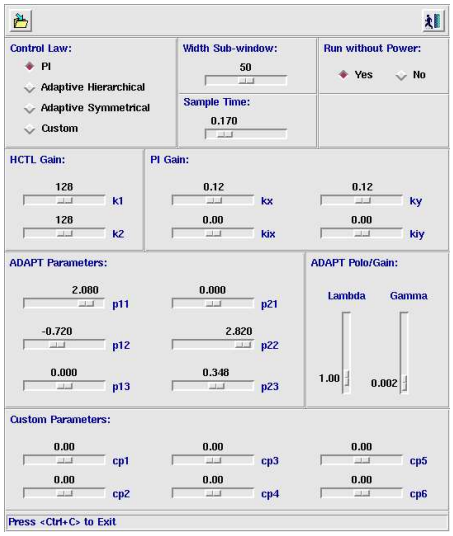

Fig. 2 - Control Parameters window

Remark 1: It is possible to modify the default parameters of the scales (e.g., -from, -resolution and -to) by editing a configuration file in order to provide more flexibility in the choice of parameters values.

C. Frame-Grabber Parameters

In this window, one can set the frame-grabber parameters as capture mode, input device, video format, image attributes (e.g., columns, rows or frames) and frame format. Furthermore, one can set the image features as bright, chrominance, contrast, hue and saturation (Fig. 3).

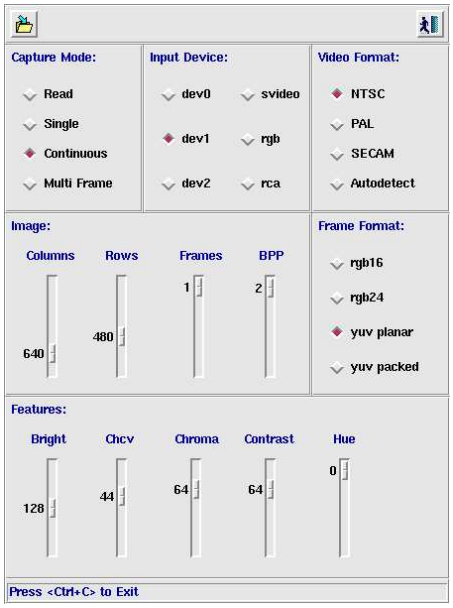

Fig. 3 - Frame-Grabber Parameters window

D. Reference Trajectory

This window allows the user to execute the following tasks: capture a image frame of the robot, draw a reference trajectory on the image (e.g., polyline, lissajous figure, rectangle or circle), put the robot arm in the ready position, drive the robot arm back to the nest, reset the robot encoders/registers or execute a custom routine. Moreover, it is possible to save an image in different formats (e.g., bmp, gif, pgm or ppm), test the image processing algorithm and view an animation of the trajectory followed by the manipulator during the task execution (Fig. 4).

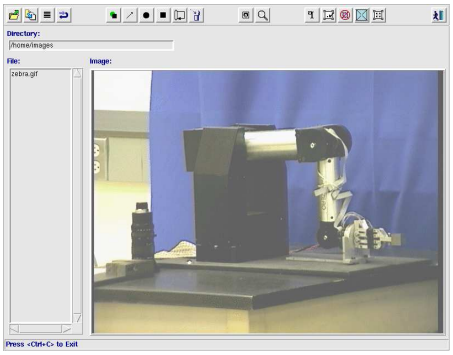

Fig. 4 - Reference Trajectory window

E. Showforces

In this window, one can verify the forces and torques acting on the robot end-effector through a bar charts (Fig. 5). The interaction forces are measured by a force sensor and the update of the each bar chart is based on the *Tcl_SetVar* function and *trace variable* command.

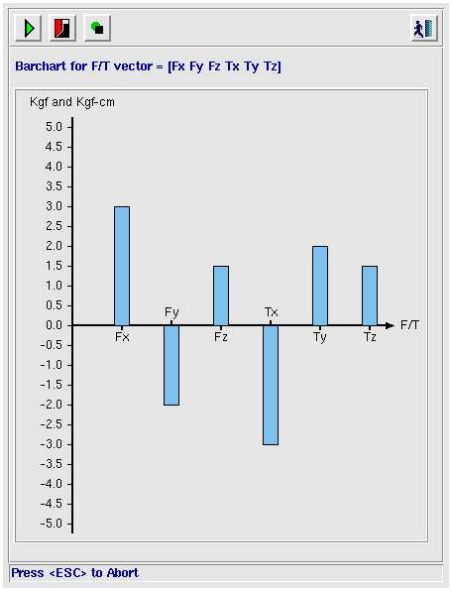

Fig. 5 - Showforces window

F. Text Editor

In this window, it is possible to handle the configuration files (e.g., open, edit, save, save as, close or refresh) and search a string by using a highlight procedure (Fig. 6).

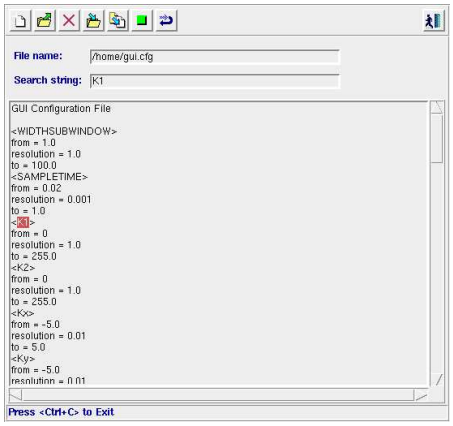

Fig. 6 - Text Editor window

IV. C Extension Commands

In this section, one includes a summary of all the new Tcl commands implemented in this work, their arguments/options and a brief description.

Commands	Arguments	Description
CtrlP	ctrlp	Set the control parameters
CustP	custp	Set the custom parameters
FGrab	fgrab	Set the frame-grabber parameters
Robot	robot	Execute the control algorithm
Start	-	Put the robot arm in the ready position
GNest	-	Drive the robot arm back to the nest
FindT	-	Execute the image processing algorithm
Reset	-	Reset the robot encoders and registers
CustR	-	Execute a custom routine
Abort	-	Abort the task execution

Tab. I - New Tcl Commands

Note that, *array* variables were used as the arguments in some new Tcl commands (see Tab. I). In addition, one describes two main C extension commands implemented through an user-developed package (Fig. 7). A package typically consists of a C function which performs all package initialization tasks and a several C functions that implement the new Tcl commands [3]. The data link between the Graphical User Interface and C routines (see Tab. II) is based on *Tcl_SetVar* and *Tcl_GetVar* functions.

Commands	Description
open_robot	Open the robot device
get_init_data	Read the robot configuration data
homerobot	Enable the motor power supply and motor servos
jmove_ready	Start the robot arm movement
jmove_is_over	Stop the robot arm movement
where	Compute the robot cartesian position
close_robot	Close the robot device
open_fgrab	Open the frame-grabber device
init_fgrab	Read the frame-grabber configuration data
mmap	Create a memory map buffer to store a image
start_capture	Start the image capture
save_frame	Save a image frame into a file
stop_capture	Stop the image capture
close_fgrab	Close the frame-grabber device

Tab. II - Description of C routines

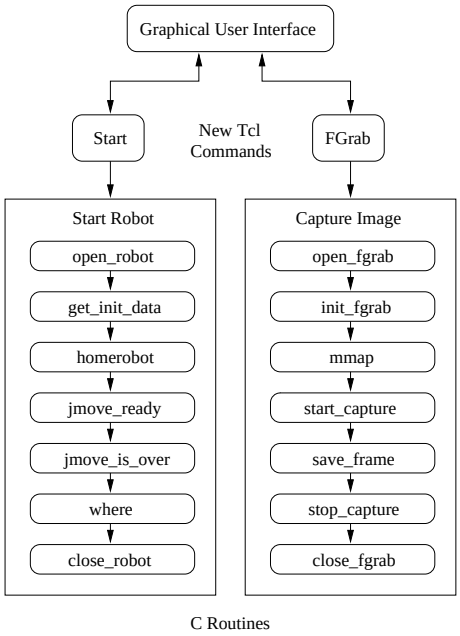

Fig. 7 - Main C extension commands and C routines.

V. Conclusion and Discussion

This work presents the application of Tcl/Tk language in the development of a graphical user interface (GUI) for a robotic system composed by a robot arm, a CCD camera and a force sensor. The GUI is used to monitor the interaction forces, set the camera-robot system parameters and execute user-developed routines for the purpose of improving the performance in the experimental tests.

Although the system output log provides a record of events or activities, it is not possible to execute a standard data input to C/C++ routines. Moreover, the robot animation is not so fast since each image frame is read from the hard-disk. In the future, one intends to enable the widget canvas to show a movie file as well as modify the event-log frame for operating as a console window in order to solve the input/output requirements.

Acknowledgment

This work was partly supported by CNPq and FAPERJ.

References

[1] J. K. Ousterhout, *Tcl and the Tk Toolkit*, 1st ed., Addison-Wesley, 1993.

[2] B. B. Welch, *Practical Programming in Tcl and Tk*, 1st ed., Prentice-Hall, 1995.

[3] S. Pather, *Creating New Tcl/Tk Commands Using C, Integrating a Tcl Interpreter into a C Application* and *Using C++ Objects in Tcl*, 1996. Available: **http://hegel.ittc.ku.edu/topics/tcltk/**

[4] R. Schwaninger, *Rapid Prototyping with Tcl/Tk*, Linux Journal, pp. 55–57 / 90–92, May, 1998.

[5] A. C. Leite and C. Lgow, *Visual Servo Control for a Robotic Manipulator*, B. A. Dissertation, Department of Electronic Engineering, Federal University of Rio de Janeiro, No 27, 2001 (in Portuguese).