COMPUTER ASSISTED INTERVENTION (CAI) SPECIFIC INTERFACES

Tinatin Mshvidobadze

Gori State University (Georgia)

Abstract

In this paper is shown development of open source software for computer assisted intervention systems. Software libraries are written in C++, but are also accessible from Python, which provides a convenient environment for rapid prototyping and interactive testing. The real-time support includes a device interface and a task library. This paper describes a set of libraries, the Cisst libraries, developed at the Johns Hopkins University to address some of the problems encountered when integrating devices for CAI.

Key words: *Cisst Vector, Real Time library, software development, LATEX documents.*

Introduction

Software development at the Engineering Research Center for Computer Integrated Surgical Systems and Technology (CISST ERC) initially focused on a library for Computer-Integrated Surgery (CIS) application development, a common interface to different tracking systems (CisTracker) and a library for Modular Robot Control (MRC). This software, now called the Cisst package, and plan to make it available under an open source license at www.cisst.org. The redesign effort has focused on portability, maintainability, real-time compatibility and establishment of a testing framework.

A prime motivation for the development of the Cisst package has been increasing need to implement novel control algorithms for new interventional systems, such as a robot for minimally-invasive throat surgery [1]. This was not feasible with the original MRC library because it relied on intelligent hardware to provide the low-level real-time control.

Cisst Vector (Foundation Library)

The CisstVector design was motivated by the desire for an efficient implementation of fixed-size vectors, matrices and transformations that is suitable for real time use. Most other vector libraries use dynamically allocated memory to store the vector elements, which is not ideal for real-time computing. An even greater number use loops as the underlying computational engine, which is not efficient for small vectors. Goal in the development of CisstVector was to achieve high computational efficiency by using stack-allocated storage, and by replacing loop mechanisms by templated engines, defined using recursive template meta programming [2]. The templated definition enables us to define vectors of different sizes and types and to apply the same operations to the vectors in a consistent form. It was identified a small number of recursive engines that would provide all the operations that we would want to perform on and between vectors.

A special feature of Cisst Vector is that it allows matrices to be stored in either row-or column-major order, to accommodate both C-style and Fortran-style two dimensional arrays. This provides convenience to C/C++ programmers, and at the same time integration with existing numerical packages based on Fortran, such as C LAPACK. In addition, CisstVector supports direct operations on subregions (slices) of vectors and matrices.

Cisst Interactive (Foundation Library)

The Cisst Interactive library provides the structure for embedding a Python-based interactive shell, the Interactive Research Environment (IRE), into our C++ programs. The IRE uses Windows for Python to provide the GUI features and relies on SWIG to automatically

wrap the C++ libraries for Python (see Fig. 1). It also provides an object registry that enables the Python and C++ software to share objects. This is especially useful when embedding the Python interpreter in an application because it allows the user to modify C++ objects from the Python interpreter.

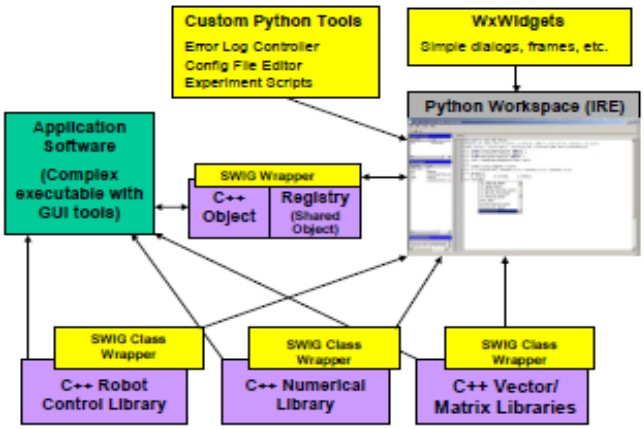

Figure 1 – Interactive Research Environment (IRE)

Cisst Real Time (Real Time Support)

The Cisst Device Interface library defines the ddiDevice Interface class, which provides the interface to the hardware.

The cisst Real Time library provides the features needed by software that must interact with the physical world in a real-time manner. It was used RTAI (Real Time Application Interface) with Linux as our real-time operating system because it is open source, appears to be well supported and provides useful features such as sharing of data between real-time and non-real-time code. It is created a Cisst OS Abstraction library so that our software can be portable to other operating systems. It was desired a software architecture that allows any control function to be transparently provided by a real-time software thread or by an external device.

The "time" indexing provides a snapshot of the history of the real-time system and can be used for data collection as well as for debugging (i.e., to provide a "flight data recorder" functionality). It also solves the mutual exclusion problem between the real-time and non-real-time parts of the system (similar to a double-buffering technique).

Development Process

This development process uses the following open source tools .

1. CVS (Concurrent Versions System) [3] for source code and document control.

2.CMake for cross-platform builds. CMake generates the appropriate compilerspecific makefiles/projects/workspaces/solutions from compiler-independent configuration files.

3. CppUnit and PyUnit to provide a unit testing framework for our C++ and Python software, respectively.

4. Dart for automated, distributed builds. We routinely use "Experimental" builds and plan to add multiple "Nightly" builds.

5. Doxygen to automatically extract (specially-formatted) documentation from the source code and create class diagrams, dependency graphs and other design documentation in HTML and LATEX formats.

6. SWIG to generate wrappers for interpreted languages such as Python.

7. CVSTrac to manage bug tracking and feature requests.

It was created of requirements documents, high-level design documents and user guides/tutorials. It was chosen to prepare these documents using LATEX[5] because it is text-based and therefore more amenable to change control.

Build process includes "compilation" of the LATEX documents into PDF and HTML formats.

Conclusions

Even though computer assisted intervention systems exist in the research and commercial communities, progress is hampered by the lack of open source software that can be certified for clinical use.

This paper presents elements of the Cisst library that facilitate the development of safe and efficient multitasking (multi-threaded) software, using concepts from component-based software engineering to achieve loose coupling between tasks. Each task contains one or more interfaces that are self-describing, in that they can dynamically provide a list of supported commands and associated parameters. This library can facilitate the development of computer assisted intervention (CAI) systems, with the ability to dynamically configure the application software to use available hardware, such as diverse robots and other devices. [4]. The Cisst software is available under an open source license. Currently, a subset of the software can be downloaded from www.cisst.org/cisst. Additional elements will become available when the implementation and documentation are sufficiently mature to support widespread dissemination

References

1. Simaan, N., Taylor, R., Flint, P.: High dexterity snake-like robotic slaves for minimally invasive telesurgery of the upper airway. In: MICCAI. (2004)

2. Veldhuizen, T.: Using C++ template metaprograms. C++ Report 7 (1995) 36–43 Reprinted in C++ Gems, ed. Stanley Lippman.

3. Fogel, K.: Open Source Development with CVS. Coriolis Open Press (1999).

4. B. Vagvolgyi, S. DiMaio, A. Deguet, P. Kazanzides, R. Kumar, C. Hasser, and R. Taylor. The Surgical Assistant Workstation: a software framework for telesurgical robotics research. In MICCAI Workshop on Systems and Arch. for Computer Assisted Interventions, Insight Journal:http://hdl.handle.net/1926/1466, Sep 2008.