An Overview

Welcome to the National Library of Medicine Insight Segmentation and Registration Toolkit (ITK).

This document provides a brief description of the project, its history, and references to additional information describing how to use, develop, and contribute to ITK.

INTRODUCTION

What is ITK?

ITK is an open-source software toolkit for performing registration and segmentation. Segmentation is the process of identifying and classifying data found in a digitally sampled representation. Typically the sampled representation is an image acquired from such medical instrumentation as CT or MRI scanners. Registration is the task of aligning or developing correspondences between data. For example, in the medical environment, a CT scan may be aligned with a MRI scan in order to combine the information contained in both.

ITK is implemented in C++. In addition, an automated wrapping process generates interfaces between C++ and interpreted programming languages such as Tcl, Java, and Python. This enables developers to create software using a variety of programming languages. ITK's C++ implementation style is referred to as generic programming. Such C++ templating means that the code is highly efficient, and that the many software problems are discovered at compile-time, rather than at run-time during program execution.

Because ITK is an open-source project, developers from around the world can use, debug, maintain, and extend the software. ITK uses a model of software development referred to as *Extreme Programming*. Extreme Programming collapses the usual software creation methodology into a simultaneous and iterative process of design-implement-test-release. The key features of Extreme Programming are communication and testing. Communication among the members of the ITK community is what helps manage the rapid evolution of the software. Testing is what keeps the software stable. In ITK, an extensive testing process is in place that measures the quality on a daily basis.

Where are ITK's origins?

In 1999 the US National Library of Medicine of the National Institutes of Health awarded a three-year contract to develop an open-source registration and segmentation toolkit, which eventually came to be known as the Insight Toolkit (ITK). ITK's NLM Project Manager was Dr. Terry Yoo, who coordinated the six prime contractors who made up the Insight consortium. These consortium members included the three commercial partners GE Corporate R&D, Kitware, Inc., and MathSoft; and the three academic partners University of North Carolina (UNC), University of Tennessee (UT), and University of Pennsylvania (UPenn). The Principle Investigators for these partners were, respectively, Bill Lorensen at GE CRD, Will Schroeder at Kitware, Vikram Chalana at Mathsoft, Stephen Aylward with Luis Ibanez at UNC, Ross Whitaker at UT, and Dimitri Metaxas at UPenn. In addition, several subcontractors rounded out the consortium including Peter Raitu at Brigham & Women's Hospital, Pat Molholt and Celina Imielinska at Columbia University, Jim Gee at UPenn's Grasp Lab, and George Stetton at University of Pittsburgh. (Note: Ross Whitaker subsequently moved to University of Utah in 2000.)

Who are the developers?

The best way to determine the names of developers is to view the CVS source code repository logs. Most of the early developers are listed in the following, however, many developers beyond those listed here will participate in the development of Insight as the code grows and becomes publicly accessible.

Some of the early developers include:

Stephen Aylward - UNC; architecture, algorithms
Dan Blezek - GE CRD; testing infrastructure
Josh Cates - Utah; imaging classes and filters
Bill Hoffman - Kitware; CMake (build process); vxl/vnl numerics; infrastructure (smart pointers, object factories, callback mechanism)
Paul Hughett - ; quality, algorithms
Luis Ibanez - UNC; imaging classes and filters
Brad King - Kitware; Mesh class, autowrapping of C++ code
Bill Lorensen - GE CRD; testing, requirements, architecture
Ken Martin - Kitware; Build process, architecture, infrastructure
Jim Miller - GR CRD; Image classes, iterators, pipeline update mechanism
Lydia Ng - Mathsoft; Image filters, clustering/segementation algorithms
Sayan Pathak – MathSoft, Inc.; image clustering/segementation algorithms
Will Schroeder - Kitware; Mesh classes, documentation, algorithms, core classes
George Stetton - Pittsburgh; image filters, algorithms
Ross Whitaker - Utah; architecture, algorithms

How do I participate?

Because ITK is an open-source system, you can participate. If you'd like to become involved, here are the following steps that you might consider.

1. Read this document
2. Obtain access to the CVS repository (read-only, anonymous access), and obtain the distribution using the following procedure (see instructions in *For More Information* at the end of document).
3. Find the Insight/Documents/InsightStart.doc and read it thoroughly.
4. As this document suggests, join the insight-developers list.
5. Contribute code or fix bugs by mailing code to the list or contacting a developer directly.

6. Once you demonstrate your abilities, obtain read-write access to the repository. Access can be obtained by contacting Bill Hoffman at Kitware at 518-371-3971 x105 and leaving your desired account name and account password.

What are the terms of use?

Because ITK was developed through US Government funding, it does not carry a copyright as many open-source systems do. Instead, it carries a no-cost license. The licence can be found at Insight/license.txt.

The important feature of the license is that it gives permission to use Insight for any purpose, including commercial purposes, with .... There is a patented code found in the Insight/Code/Patented directory. If you use any of these code in commercial application, you must contact the patent holder to obtain permission.

TECHNICAL SUMMARY

The following sections summarize the technical features of the NLM's Insight ITK toolkit.

Design Philosophy

The following are key features of the toolkit design philosophy.

- The toolkit provides data representation and algorithms for performing segmentation and registration. The focus is on medical applications; although the toolkit is capable of processing other types.
- The toolkit provides data representations in general form for images (arbitrary dimension) and (unstructured) meshes.
- The toolkit does not address visualization or graphical user interface. These are left to other toolkits (such as VTK, VisPack, MetaImage, etc.)
- The toolkit provides minimal tools for file interface. Again, this is left to other toolkits/libraries to provide.
- Multi-threaded (shared memory) parallel processing is supported.
- The development of the toolkit is based on principles of extreme programming. That is, design, implementation, and testing is performed in a rapid, iterative process. Testing forms the core of this process. In Insight, testing is performed continuously as files are checked in, and every night across multiple platforms and compilers.

Architecture

The following are key features of the toolkit architecture.

- The toolkit is organized around a data-flow architecture. That is, data is represented using data objects which are in turn processed by process objects (filters). Data objects and process objects are connected together into pipelines. Pipelines are capable of processing the data in pieces according to a user-specified memory limit set on the pipeline.
- Object factories are used to instantiate objects. Factories allow run-time extension of the system.

Implementation Philosophy

The following are key features of the toolkit implementation philosophy.

- The toolkit is implemented using generic programming principles. Such heavily templated C++ code challenges many compilers; hence development was carried out with the latest versions of the MSVC, Sun, gcc, and SGI compilers.

- The toolkit is cross-platform (Unix and Windows).

- The toolkit supports multiple language bindings, including such languages as Tcl, Python, and Java. These bindings are generated automatically using an auto-wrap process.

- The memory model depends on "smart pointers" that maintain a reference count to objects. Smart pointers can be allocated on the stack, and when scope is exited, the smart pointers disappear and decrement their reference count to the object that they refer to.

Build Environment

Insight uses the CMake (cross-platform make) build environment. CMake uses configure and make on Unix and generates projects and workspaces in the windows environment.

FURTHER INFORMATION

The following information is available from the Insight CVS repository.

(To download Insight using CVS, use the following steps.
cvs -d :pserver:anonymous@public.kitware.com:/insight/cvsroot login
with password "insight"...then get the source/documentation as follows:
cvs -d :pserver:anonymous@public.kitware.com:/insight/cvsroot checkout Insight)

Once you have done this, look for the following documents:

- Insight/Documents/InsightStart.doc|.pdf (Word/PDF)
  The basics of working with CVS, obtaining the source code repository, and compiling Insight.
- Additional Documentation (all found in the directory Insight/Documents)
  - CMake (CMake.doc|.pdf (Word/PDF)
  - Requirements (Requirements.doc|.pdf)
  - Style Guide (a must read for coders) (Style.ppt|.pdf)
- Contacts

- Terry Yoo (NLM/NIH Insight Project Manager yoo@fluorite.nlm.nih.gov)
- Bill Lorensen (PI GE CRD lorensen@crd.ge.com)
- Will Schroeder (PI Kitware, Inc. will.schroeder@kitware.com)
- Stephen Aylward with Luis Ibanez (UNC aylward@unc.edu, ibanez@cs.unc.edu).
- Ross Whitaker (PI Utah whitaker@cs.utah.edu)
- Vikram Chalana (PI Mathsoft vikram@statsci.com)
- Dimitri Metaxas (PI UPenn dnm@central.cis.upenn.edu)