itkFlowRun – a Visual Programming Tool for ITK Image Filters

Diego Ferreira dos Santos[(1)], Eduardo Tavares Costa[(1)], Marco Antonio Gutierrez[(2)]
1: School of Electrical and Computing Engineering (FEEC) and Center for Biomedical Engineering (CEB), State University of Campinas (UNICAMP), Campinas, Brazil
2: Informatics Division, Heart Institute (InCor),
University of São Paulo Medical School, São Paulo, Brazil
santos@ceb.unicamp.br

Abstract

itkFlowRun is an open source software that aims to be an extension of Insight Registration and Segmentation Toolkit (ITK) to rapidly and easily create image filters pipeline through a visual programming tool environment. It has been developed for Linux environment using the following libraries: Fast Light Toolkit (FLTK), Boost and ITK. This paper presents an overview of the main functionalities and describes the software architecture and plug-in mechanism, allowing third-party plug-ins to be developed. itkFlowRun is still being developed and will be released in the near future.

1. Introduction

The *Insight Registration and Segmentation Toolkit (ITK)* is an open source software to perform registration and segmentation of medical images [1,2]. It has several image filters ready to use and new tools can be incorporated. ITK is implemented in the C++ language using Generic Programming Techniques, a way to generalize software components for easily reuse in a wide variety of situations [3, 4].

The major problem in implementing and visualizing ITK image filters pipeline is that ITK is a library of methods and not really an end-user application. Moreover, it is necessary to develop code in high level language and compile it every time the user modifies the filters pipeline.

One alternative to improve the development of methods based on ITK, is to incorporate a Visual Programming Tool to rapidly build, visualize and test image filters pipeline. With this approach, filters or modules can be represented as a network of interconnected modules. Each module can have input or output ports used to receive the data flow, as well as internal parameters to control the module execution. Modules communicate with each other through connections to these ports. The implementation details of each module are transparent to the user. The user has only to connect modules in the right order and set some control parameters, when necessary for the filter execution.

Some complex frameworks exist that provide this functionality, some of them are free. As examples we can cite the MeVisLab and SCIRun [5], being both IDEs for visual data-flow programming. One of the main problems of these libraries is that they do not provide a method to use all existing ITK data types and both of them use XML file to provide ITK filters information, being necessary that the user know the right syntax to create new modules, what is not the case in our proposed software.

2. Methods

Our proposed framework, named itkFlowRun, is an open source software developed in C++ and a set of open source software toolkits: ITK (http://www.itk.org) for image processing, BOOST (http://www.boost.org) for graph algorithms, CMAKE (http://www.cmake.org) for building and FLTK (http://www.fltk.org) for graphical user interface.

In itkFlowRun, modules may have one or more input or output ports. The output port of a module is connected to the input port of another module. The algorithm chosen to parse modules connections from the diagram was the Topological Sort algorithm.

As ITK DataObject is the base class to all ITK Objects like images, cells, meshes, and others [1], all output data of ITK image filters are casting to ITK DataObject class. The data of a module input port is a pointer to the ITK DataObject of another module output port connected to it.

The main classes developed in itkFlowRun are: **userInterface** – main itkFlowRun graphical user interface; **moduleFactory** – an object factory. All modules loaded dynamically are put in a linked list; **moduleBaseGUI** – a module container. This implements a canvas to draw modules, draw module

connections and so on; **moduleGraph** – implements a graph data structure to store module connections; **moduleButtonGUI** – implements a graphical button for modules; **moduleBase** – base for all modules (plugins) developed; **modulePort –** base to all input/output ports; **moduleVars** – manage and store all module variables; **moduleEditor** – create new header and source files (to build a new plugin automatically) according to parameters set by the user;

All modules (plug-ins) were developed in C++ and the class must be inherited from **moduleBase** class. To save user time, we developed a method to create and build modules automatically. The user has only to set some control parameters, such as: class name, ITK object name, number of input or output ports and so on. The application generates the appropriate files to build the desired plug-in using all possible ITK data types and dimensions ranging from 2D to 6D.

3. Results

The proposed framework provides a Graphical Interface to rapidly and easily build ITK image filters pipeline. The visual programming can be used to connect two or more image filters through a visual data flow modeling tool. The user can choose, at run time, the data types (all supported by ITK) for input and output data for each module. At each module, there is a semaphore to show the execution status: **Green**: module was executed; **Yellow**: module is executing and **Red**: module executed with error.

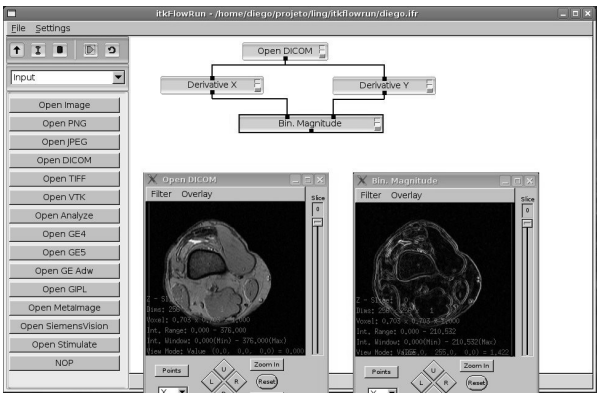

Figure 1 - An example of itkFlowRun application

In Figure 1, all active modules are in the left side of the main window. In this example, we constructed a diagram with three available modules: "Open DICOM", to read DICOM image files, "Derivative", to calculate the spatial image derivatives and "Bin Magnitude", to compute the gradient magnitude. We set control parameters to the Derivative modules. In each module, we can see the output data (image). The two images shown in Figure 1 are the output images for "Open DICOM" and "Bin Magnitude" modules, respectively.

4. Conclusion

In this paper we briefly describe itkFlowRun, an open source environment to rapidly and easily build ITK image filters pipeline.

With itkFlowRun it is less exhaustive to build and to modify image filters pipeline, when compared with the traditional method used by ITK users. The process of creating and managing the pipelines, module execution and connection becomes more intuitive, faster and easier.

Another important feature is the ability to accept third-party plug-ins. When a plug-in has been created, it becomes available for use and no more code has to be implemented because everything is done using visual programming. The efforts for coding and building every change in the pipeline disappear with the use of itkFlowRun. No more detailed knowledge of ITK syntax is needed if the user is not familiar with programming languages when developing new modules.

5. References

[1] Ibanez, L., Schroede, W., Ng, L., Cates, J., and the Insight Software Consortium., *The ITK Software Guide*, U.S.A. 2003.

[2] Johnson, C.R., and Hansen, C.D., *The Visualization Handbook*, Elsevier Academic Press, Oxford, 2005.

[3] Alexandrescu, A., *Modern C++ Design: Generic Programming and Design Patterns Applied*, Addison Wesley, 2001.

[4] Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C., *Introduction to Algorithms, Second Edition*, MIT Press, USA, 2001.

[5] Ingmar, B., Robert, V.U., Ivo, W., Luis, I., and Jan-Martin, K. "*Comparison of Four Freely Available Frameworks for Image Processing and Visualization That Use ITK",* IEEE Transactions on Visualization and Computer Graphics, Vol. 13, N. 3, pp. 483-493, 2007.