An Object-Oriented Client-Server System for Interactive Segmentation of Medical Images Using the Method of Active Contours

Alan L. Scheinine, Marco Donizelli and Marco Pescosolido

CRS4 (Center for Advanced Studies, Research and Development in Sardinia)
via Nazario Sauro 10, 09123 Cagliari (Italy)
Email: donza@crs4.it

Abstract. In this paper we describe the first prototype of a distributed medical imaging system suitable for the visualisation and processing of medical images. The prototype is an object-oriented client-server system that provides a complete framework for the interactive segmentation of blood vessel contours from X-Ray Computed Tomography (CT) or Magnetic Resonance Imaging (MRI) scans, through the method of snakes or active contours. The system has been implemented exploiting the benefits of recent software developments, such as the Java language programming and the CORBA distributed object technology, which simplify the building, the maintenance and the portability of this kind of distributed applications.

Keywords: Distributed medical imaging systems, Distributed computing, Java, CORBA, Active contours

1 Introduction

Beyond the immediate diagnostic value of medical images from X-Ray Computed Tomography (CT), Magnetic Resonance Imaging (MRI) and ultrasound scans, image data can be used for physical analysis and simulation. For example, the reconstruction as a geometric data set of a section of an artery could be used for medical procedural planning and for conducting research in vascular physiophathology through fluid dynamic simulations [1]. This key role played by medical imaging has led to an increasing demand for electronic medical imaging systems (EMISs, [2]) that allow visualisation and processing of medical images. Recently, the growing demand for EMISs has been coupled with a need to access medical images and other diagnostic information remotely across networks, and to integrate and analyse data from various sources. These requirements, together with the development of data interchange standard formats such as DICOM [3], has also led to an increasing demand for *distributed* EMISs [1]. These systems go far beyond the original previsions of the instruments as standalone diagnostic workstations, and must have the capabilities to access images across networks, provide powerful image processing tools, as well image retrieval and storage mechanisms, manage various data from various sources, and integrate different software packages. Recent software developments, such as the Java programming language [4] and the CORBA distributed object technology [5], meet the needs underlying these *distributed* EMISs, simplifying their development, maintenance and portability. This paper describes the design and implementation of the first prototype of a *distributed* EMIS suitable for the visualisation and processing of medical images from different modalities. The prototype is an *object-oriented* client-server system that provides a complete framework for the interactive segmentation of CT or MRI scans of carotid artery sections including the first bifurcation, through the method of *snakes or active contours* [6].

2 The System Requirements

In the design of the system prototype, the following requirements have been considered:

- The prototype must provide a complete, usable and efficient graphical user-interface (GUI), with all the controls needed to manage, visualise and process a sequence of images. In particular, the GUI must provide all the tools useful to interact with a snake-based edge detection algorithm and to evaluate the algorithm results (due to the poor definition of medical images and the complexity of anatomical structures a user interaction is essential to guide the segmentation process [7]).
- The prototype GUI must be portable, in order to lead the visualisation and the processing of the images on any platform.
- The prototype GUI must be flexible, in order to permit the implementation of time-consuming segmentation algorithms using the most appropriate fast computer language and, possibly, the execution on special-purpose machines.
- The framework for the communication and the data exchange among the prototype components must be able to work with different languages and platforms.
- The prototype architecture must be scalable, in order to lead the simple addition of new segmentation algorithms or other useful system components without requiring the rewriting of the system.

3 Solution: Java and Distributed Object Technology

The Java programming language [4] and the CORBA distributed object technology [5] have met the requirements underlying the design of the system listed above.

Java is an object-oriented programming language that has sparked over the past two years considerable interest among software developers. Besides the properties that it has in common with other OO languages (modularity, reusability, security), three features supported by Java has been particularly relevant to its choice as the language used to implement the prototype GUI:

- Java is a complete programming environment that provides a complete set of services and packages that support GUI development, graphics and image visualisation and treatment.

- Java is a platform-independent language, so an application written in Java is completely portable and executable on any platform.
- Java has built-in support for multi-threading programming, relevant for an application like a medical imaging system characterised by time-consuming processing algorithms.

In addition to these three properties, Java has also built-in support for TCP/IP based networking and for distributed computing middleware (Java RMI). Even if this feature guarantees the possibility of developing distributed applications within the Java environment, a different technology has been chosen to implement the framework that manages the communication among the system prototype components: the CORBA distributed object technology. CORBA, the Common Object Request Broker Architecture defined by the Object Management Group (OMG), specifies how software objects distributed over a network can work together without regard to client and server operating systems and programming languages. The kernel of the CORBA architecture is the CORBA Object Request Broker (ORB), the middleware that establishes the client-server relationships between objects. Using an ORB, a client object can invoke a method on a server object that can be on the same machine or across the network. The ORB intercepts the call and finds an object that implements the request, passes it the parameters, invokes its method, and returns the result. The client does not have to know the object's location, neither its programming language nor its operating system: the ORB takes care of all these details. An ORB can manage various servers on different hosts, and this guarantees the complete scalability of a CORBA-based distributed system. Since the Java and C++ binding of CORBA are based on the object method invocation, CORBA is completely compatible with object-oriented languages as Java or C++, leading in this way the simple development of distributed application as the proposed medical system.

4 The System Architecture

Figure 1 shows the object-oriented distributed architecture of the system prototype.

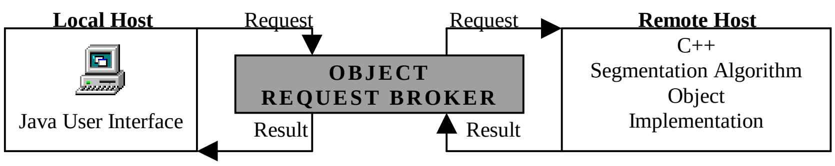

Figure 1. The OO distributed architecture of the system prototype.

The client side of the system is represented by a GUI written in Java. It has been implemented developing a set of packages and classes that represent both the visual components of the interface and the underlying modules needed to perform the retrieving, the treatment and the visualisation of a sequences of images, some simple image processing routines, and the complex remote snake-based segmentation procedures:

- *Package slice*: contains the classes useful to the creation and storage in memory of the images that form the CT or MRI scan sequence to analyse. The image data can be retrieved from graphic files of various formats (GIF, JPEG, BMP, and raw data), or from URL locations, or from a DICOM archive through a TCP/IP socket-based DICOM server.
- *Package sequence*: contains the classes that build the GUI components useful to visualise in various ways the entire sequence of images.
- *Package slicedisplayer*: contains the classes that build the GUI component useful to visualise an image of the sequence, in which it is possible to perform simple image processing operations and to draw interactively overlays on the image.
- *Package image*: contains classes that build the GUI components useful both to display some characteristics of the visualised image (histogram, cross-section, bit-planes) and to modify interactively the appearance of the image (contrast stretching, grey-scale slicing, intensity mappings).
- *Package overlay*: contains all the classes needed to draw, modify and store in memory the overlays on the visualised image.
- *Package segmentation*: contains the classes that build the GUI components by which the user can employ the remote snake-based segmentation algorithms.

A generic snake-based edge detection algorithm written in C++ represents the server side of the system. For now, in the prototype only one algorithm is available, based on the Generalized Active Contour Model theory [8]. It has been implemented using the public-domain C++ library GSNAKE API [9], jointly developed by the Information Technology Institute (ITI), Singapore, and the School of Applied Science, Nanyang Technological University (NTU), Singapore.

The client side and the server side of the system are connected through the functionalities of a CORBA Object Request Broker. The data types and methods useful to perform a complete remote segmentation procedure have been defined following the CORBA specifications and using an Interface Definition Language (IDL), which has been subsequently mapped both to the Java language and to the C++ language. The classes so obtained have been integrated into the Java GUI and the C++ algorithm, and their methods invocation represent the modality by which the client side and the server side of the system interoperate through the ORB during a segmentation session. These methods concern the connection and the disconnection between client and server, the posting of the image data and of the initial contour points from the client to the server for being processed, the posting of the extracted contour from the server to the client for being visualised and postprocessed.

5 Discussion

Fig. 2 shows the contours extracted from three different CT scans of a carotid artery using the functionalities provided by the described system. The good result obtained, together with the good performances of the system and its relative simplicity of implementation, indicate that the technologies such as Java and CORBA are suitable for the design and development of medical imaging systems. The object-oriented feature of the system guarantees its simple maintenance and scalability, while its client-server architecture allows the different components of the system to be implemented using the most appropriate computer language and platform.

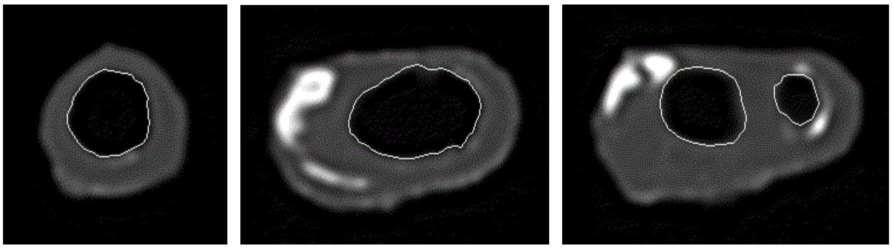

Figure 2. The contours extracted from three different CT scans of a carotid artery using the functionalities provided by the described system.

Acknowledgements

The authors gratefully acknowledge Dr. Massimiliano Tuveri for his important help in providing vascular image sequences.

References

1. D. P. Giddens, C. K. Zarins and S. Glagov: The Role of Fluid Mechanics in Localization and Detection of Atherosclerosis. Journal of Biomechanical Engineering 115:588-594, 1993.
2. P. Jain, S. Widoff, D. C. Schmidt: The Design and Performance of MedJava. USENIX Conf. On Object-Oriented Technologies and Systems, Sante Fe, New Mexico, April 1998.
3. NEMA Standard Publication PS3.X: Digital Imaging and Communications in Medicine Parts 1-10, 1994.
4. M. Campione and K. Walrath: The Java Tutorial - Object-Oriented Programming for the Internet. Addison-Wesley, 1996.
5. R. Ben-Natan: CORBA - A Guide to Common Object Request Broker Architecture. McGraw-Hill, 1995.
6. M. Kass, A. Witkin and D. Terzopoulos: Snakes: Active Contour Models. IEEE Conf. On Computer Vision and Pattern Recognition:259-268, 1987.
7. G. J. Sicewright, J. M. Knapman, W. Dickson and P. J. Elliot: Interactive Image Segmentation Applied to CT and MR Images, Proc. of CAR:328-333, 1993.
8. K. F. Lai and R. T. Chin: Deformable Contours: Modeling and Extraction. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(11):1084-1090, 1995.
9. K. F. Lai, S. Chan, C. W. Ngo, E. L. Ang and K. W. Ong: GSnake API version 1.0: reference manual. 1995.