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

Alan L. Scheinine, Marco Donizelli, Marco Pescolscido, Piero Pili, Andrea Giachetti, Massimiliano Tuveri and Gianluigi Zanetti
{scheinin, donza, marco, piero, giach, mtuveri, zag}@crs4.it
CRS4 (Center for Advanced Studies, Research and Development in Sardinia) via Nazario Sauro 10, 09123 Cagliari (Italy)

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, using the Generalised Active Contours Model. The system has been implemented exploiting the benefits of recent software developments, such as the Java programming language and the CORBA distributed object technology, which simplify the building, the maintenance and the portability of this kind of distributed applications.

1 Introduction

Beyond the immediate diagnostic value of medical images from X-Ray Computed Tomography (CT) and Magnetic Resonance Imaging (MRI), 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 physiopathology through fluid dynamic simulations of blood flow [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 [4]. These systems go far beyond the original previsions of the instruments as stand-alone diagnostic workstations, and must have the capabilities to access images across networks, provide powerful image processing tools as well as image retrieval and storage mechanisms, manage various data from various sources, and integrate different software packages. Recent advances both in high-speed networks and in software developments, such as the Java programming language [5] and the CORBA distributed object technology [6], meet the needs underlying these distributed EMISs, improving their efficiency and 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 system is characterised by an object-oriented client-server architecture that provides a complete framework for the interactive segmentation of CT or MRI scans of carotid artery sections using the Generalised Active Contour Model (GACM) [7]. The client side of the system is represented by a graphical user interface (GUI) written in Java, while a GACM-based algorithm written in C++ represents the server side. The client and the server are connected through a CORBA communication infrastructure, i.e., through a distributed object computing middleware.

2 The system requirements

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

1. The system GUI must be complete and usable, with all the tools useful to:
   • retrieve medical images from different sources (raw data or graphic files, DICOM archives, URLs);
   • visualise the images as in conventional film-based radiology;
   • manipulate the images in order to improve their appearance, quality, or to make easier the identification of the anatomical objects under interest;
   • interact with the GACM algorithm (due to the poor definition of medical images and the complexity of anatomical structures a user interaction is essential to guide the segmentation process [8]);
• evaluate the algorithm results and postprocess them (addition/removal of contour points, smoothing of the contour curvature, extrapolation of complex contour shapes in case the algorithm fails).

2. The system GUI must be portable, in order to guarantee its functionalities on any platform.

3. The system GUI must be flexible, in order to guarantee the efficiency of the system permitting the implementation of time-consuming segmentation algorithms using the most appropriate fast computer language and, possibly, the execution on special-purpose machines.

4. The system components communication and data exchange framework must be able to work with different languages and platforms.

5. The system 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 and the CORBA distributed object technology 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, and security), three features supported by Java have been particularly relevant to its choice as the language used to implement the system GUI:

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

2. Java is a platform-independent language, so an application written in Java is completely portable and executable on any platform.

3. Java has built-in support for multi-threading programming, relevant for the efficiency of 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 (the Java Remote Method Invocation). Even if this feature guarantees the possibility of developing distributed applications using only the Java environment, a different technology has been chosen to implement the framework that manages the communication among the system components: the OMG-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. This specification defines the functionality of an Object Request Broker (ORB), a software layer which allows developers to define objects which can be accessed over a network through clearly defined, high-level interfaces, and which 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. In this way, the CORBA communication infrastructure automates common network programming tasks (such as object selection, location and activation, as well as parameter marshalling and framing), thereby enhancing application flexibility [4]. In addition, 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 design and implementation

Figure 1 shows the distributed architecture of the system. The three primary components in the architecture are the Java GUI client, the C++ GACM algorithm that acts as a server, and the CORBA ORB that manages their communication. Two secondary components are the DICOM server and the 3DGEOM server, connected to the client by a TCP/IP socket protocol. Each of these components is outlined below.

4.1 The Java GUI

On the client side of the system there is a graphical user interface written in Java. We have implemented it basing on standard Java packages, such as java.awt and java.awt.image, and developing eight new packages containing all the GUI components, both the visual ones (fig. 2 shows some of them) and the underlying modules needed to perform the tasks required to the interface.
**Figure 1.** The distributed architecture of the system.

**Figure 2.** Some visual components of the system GUI.
1. Package slice: contains the components useful to retrieve, store and manage in memory the images that form the CT or MRI scan sequence to analyse:
   - Slices Loader: which downloads a sequence of images from a DICOM archive. This component includes the visual controls and the Java methods useful to establish the connection and interact with the DICOM server of the system. In addition, being Java able to visualise only 8-bit image data, it includes the controls needed to carry out the most appropriate conversion from n-bit (n = 12, 16, 24) data to 8-bit data.
   - Slice Loader: which downloads a single image from different sources, requiring information as the location (path or URL) or the format (GIF/JPEG/TIFF/BMP graphic file, raw data file) of the source. As for the Slices Loader, this component includes controls for the 8-bit conversion of image data.
   - Slice: the component by which every loaded image with its data is stored in memory.
   - Slices Store: which manages in memory the entire sequence of images, allowing operations as the adding/removal/replacement/insertion of an image.

2. Package slicedisplayer: contains the components useful to visualise the images of the sequence:
   - Slice Displayer: which displays an image and provides tools for performing simple image processing operations (zoom, crop, rotate, flip, negative, bright, dim) and for drawing interactively overlays on the visualised image.
   - Slice Displayers Store: which manages in memory a set of Slice Displayer components (the GUI allows the contemporary presence of more than one Slice Displayer, so, for example, parallel processing sessions on the same image could be performed).

3. Package sequence: contains the components (Sequence Movie, Sequence Plate) useful to visualise the images of the sequence as in conventional film-based radiology.

4. Package image: contains the components useful to display some properties (Image Histogram, Image Cross-Section, Image Pixel Values, Image Grey Levels, Image Bit-Planes, Image Statistics) and to modify interactively the appearance (Image Contrast Stretching, Image Grey-Scale Slicing, Image Intensity) of an image displayed in a Slice Displayer component.

5. Package overlay: contains the components useful to manage the creation of various types of overlays (box, polyline, polygon) on an image displayed in a Slice Displayer component, as well as their modification and storage in memory.

6. Package segmentation: contains the component useful to employ the remote GACM algorithm through the GACM server. This component includes the visual controls and the Java methods needed to establish the connection and interact with the GACM server through the CORBA ORB.

7. Package postprocessing: contains the components useful to postprocess the contours extracted by the GACM algorithm (Contour Remove Point, Contour Smoothing, Contour Extrapolation) and the component including the visual controls and the Java methods needed to establish the connection and interact with the 3DEGEO server of the system (Contour Parameterisation).

8. Package remote: contains the component that defines and implements the methods useful to perform a client/server interaction using a TCP/IP socket protocol. These methods are used in the Slices Loader and in the Contour Parameterisation components of the interface.

4.2 The GACM server
On one server side of the system there is the GACM algorithm written in C++. We have implemented it 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. We have integrated the original code with all the C++ methods needed to interact with the Java GUI client.

4.3 The communication infrastructure between the JAVA GUI client and the GACM server
A CORBA Object Request Broker manages the communication between the JAVA GUI client and the GACM server. Being in a CORBA environment, the GACM server provides services to the Java GUI client by providing to it objects for use, and the Java GUI client exploits these services by invoking methods on the GACM server objects with standard Java method invocation syntax. The functionalities of the GACM server are specified through the functionalities of its objects, which are clearly defined through their public interface. An interface consists of operation and attribute (property) specifications, and the Interface Definition Language (IDL) is a standard language, defined by the Object Management Group, for defining such interfaces [10]. In our case, the GACM server objects interface specification has consisted in the definition of the methods and data types useful to make the GACM algorithm usable by the Java GUI client, that is, the definition of the methods needed to:
   1. post the data concerning the image to be processed from the Java GUI client to the GACM server;
2. post the data concerning the initial contour input by the user from the Java GUI client to the GACM server;
3. post the values of the parameters required by the GACM algorithm from the Java GUI client to the GACM server;
4. start the segmentation process;
5. post the data concerning the extracted contour from the GACM server to the Java GUI client, and of the data types related to:
   1. the image to be processed;
   2. the contour;
   3. the GACM algorithm parameters.

Once defined the object interfaces in IDL, we have employed the IONA OrbixWeb IDL compiler [10] and the IONA Orbix IDL compiler [11] to produce respectively the Java classes and the C++ classes corresponding to each IDL. The classes so obtained have been then implemented and integrated into the Java GUI client and the C++ GACM server, and their methods invocation represent the modality by which the Java GUI client and the GACM server interoperate through the ORB during a segmentation session.

4.4 The DICOM server and the 3DGEOM server

In addition to the GACM server, the Java GUI client interacts with two other servers using the TCP/IP socket protocol: the DICOM server and the 3DGEOM server. The former is dedicated to the retrieval of image data from a DICOM archive and to the delivery of these data to the Java GUI. We have implemented it using the C programming language and employing a public-domain C library suitable for the management of DICOM files. The latter is dedicated to the computation from the contours extracted by the GACM algorithm of the parametric curves useful for the 3D modelling of the carotid artery. We have implemented it using the C language and the Shapes geometric computing environment [12], which extends traditional programming languages such as C with capabilities for defining and processing geometric objects.

5 Results

Fig. 3 shows the contours extracted from four different CT scans of a carotid artery using the functionalities provided by the described system.

![Figure 3. The contours extracted from four different CT scans of a carotid artery using the functionalities provided by the described system.](image-url)
6 Conclusions and future work

We have described the design and implementation of an object-oriented client-server system suitable for the interactive segmentation of CT or MRI scans of carotid artery sections using the Generalised Active Contour Model. The system has been developed employing the Java programming language and the CORBA distributed object technology. The good result obtained, together with the good performances of the system and its relative simplicity of implementation, indicate that these technologies are suitable for the development of distributed medical imaging systems. The object-oriented feature of the system guarantees its simple maintenance and scalability, while the client-server architecture guarantees its efficiency, allowing the different components of the system to be implemented using the most appropriate computer language and platform.

In order to demonstrate the complete feasibility of these concepts, a wider and fully CORBA based system is to be realised. This leads to the building and insertion into the system of new image processing modules similar to the GACM component, and to the modification of pre-existing components, such as the DICOM server and the 3DGEOM server, from C-TCP/IP applications into C++-CORBA modules. Regarding the performances, the lacks of the system are focused in the Java GUI client, being Java an interpreted language written to a virtual machine. Improvements in this sense can be achieved by hand-optimised Java code, but most of all by the employment of high-speed Java “just-in-time” (JIT) compilers, which translate Java bytecode into native code for the local machine the application runs on [2].

References