Distributed medical imaging applications using Java technology

Qiusha Min* and Robert J.T. Sadleir†
*Central China Normal University, Wuhan, China
E-mail: qiusham@mail.ccnu.edu.cn
†Dublin City University, Dublin 9, Ireland
E-mail: robert.sadleir@dcu.ie

***Abstract*— Advances in Internet technologies have opened up new opportunities in relation to medical image interpretation. This task can be accomplished outside the hospital using distributed medical imaging applications. In this paper, a Java-based distributed medical imaging application is presented. This application is able to access to a remote medical image dataset via a network and provide the necessary interpretation functions, such as windowing and fly-through visualisations. Experimental results show that this Java-based medical imaging application has the ability to provide comprehensive functionality for radiology interpretation as well as a high level of user friendliness. Thus demonstrating Java is a suitable tool for developing distributed medical imaging applications.**

I. INTRODUCTION

In recent years, the advent of filmless radiology and increasing bandwidth have brought about many benefits in the healthcare industry. For example, via the Internet, radiologists can carry out an interpretation outside the hospital. Moreover, collaboration between radiologists at different institutions and locations becomes more convenient with minimal difficulties and cost. However, all of these advantages are dependent on the ability to efficiently use the Internet to create distributed medical imaging applications. A great deal of research has focused on the development of distributed medical imaging applications using different technologies.

In this paper, we discuss the use of Java to develop a distributed medical imaging application. The choice of Java was primarily motivated by its suitability to implement distributed applications. In addition, Java has strengths in both 2D and 3D graphics and the key feature of this technology is that is it "write once and run anywhere". Consequently, Java is suitable for delivering both medical analysis functionality as well as strong visualisation capabilities. The following sections present some details of our Java-based medical imaging application and a series of experiments undertaken in order to evaluate this application in terms of its usability, functionality and performance.

II. PREVIOUS WORK

Web technologies are widely used in the field of radiology. This is motivated by the fact that Web-based applications can provide a straightforward and cost-effective way of remotely accessing medical image data [1].

Among existing Web technologies, Java is a popular platform and many Web-based medical imaging applications are developed using this technology. An early study evaluated Java as a Web-based teaching file tool which enabled overlaying of labels and text on radiology images [2]. The overlay function could be used to display an annotation near the specific findings to facilitate student studies. In addition, 16-bit DICOM images could be windowed by Java applications without loss of any gray-scale details. One of the early Java-based teleradiology systems was developed by Lee et al. [3]. Using their system, the radiologist can make an emergency diagnosis even from a location other than a hospital. For a rural hospital without a full-time radiologist, utilisation of this system facilitated the provision of an efficient and timely diagnosis. Abrardo and Casini [4] used Java to create a more sophisticated teleradiology system. Besides basic teleradiology functions, e.g. access to a remote medical image database and 2D image display, their system could support a collaborative diagnosis by multiple users in different locations. The observation of other users' actions during the process of the collaborative diagnosis could also be achieved in real time using their system. Slomka et al. [5] and Knoll et al. [6] used Java to develop solutions for remote viewing of nuclear medicine images. Wallis [7] compared Java-based teleradiology solutions with other alternatives, e.g. screen-mirroring programs to view the workstation display at a remote site and basic HTML programs to view static images. He concluded that the Java approach to implementing remote image interpretations outperforms other alternatives, especially by offering a short response time on standard PCs.

Unfortunately, many current web-based medical imaging applications lack powerful 3D features due to the fact that 3D visualisation is an extremely computationally intensive operation. A large number of studies have confirmed the feasibility and clinical benefit of 3D functions used in radiological interpretation [8, 9]. These studies indicated that integration of 3D visualisation functions could significantly improve aspects of sensitivity and efficiency in examiner performance. Consequently, there is a clear need to develop a

Financially supported by self-determined research funds of Central China Normal University(CCNU) from the colleges' basic research and operation of MOE (No. CCNU15A05024) and Natural Science Foundation of Hubei Province in China (No. 2015CFB526).

more functional and suitable solution for remote viewing and interpretation of medical images and this paper presents an example of a Java based application that is intended to meet these requirements.

III. METHOD

A. Java

Java is a high-level computer programming and a platform independent language. It was developed by Sun Microsystems, later acquired by Oracle Corporation. This technology is designed to support distributed applications that are deployed via the Internet. Java applications are compiled once and run on all platforms that have a Java virtual machine (JVM) installed regardless of computer architecture. The most popular Web browsers and the most recent operating systems support Java-based Web applications. Secure Java based Web applications can be realized using the Java Secure Socket Extension (JSSE). In addition, Java can be used to implement sophisticated imaging capabilities in image processing and visualization.

B. Application Design

A comprehensive remote access medical imaging application should include the following features: (1) interaction with the local file system, (2) basic functions for 2D image processing, and (3) 3D reconstruction of regions of interest within the dataset. In this way, the application can provide remote interpretation in such a way that radiologists view images from a downloaded dataset and manipulate them using 2D or 3D functions.

In this paper, a client application for interpretation of computed tomography colonography (CTC) is designed to satisfy all these requirements. The operations of this application are presented in Fig 1. When a client requests to start the application, the executable JAR (Java Archive) file is sent to the client and the application is then launched in a browser. In order to offer a short response time, the viewing dataset is initially transmitted to the client computer and stored in the local file system. The medical images are subsequently read from the local file. Users can use 2D tools, such as windowing and zooming, to identify useful information contained in an image. In addition, 3D functions which can provide visualisations of the volumetric dataset are also implemented by this application.

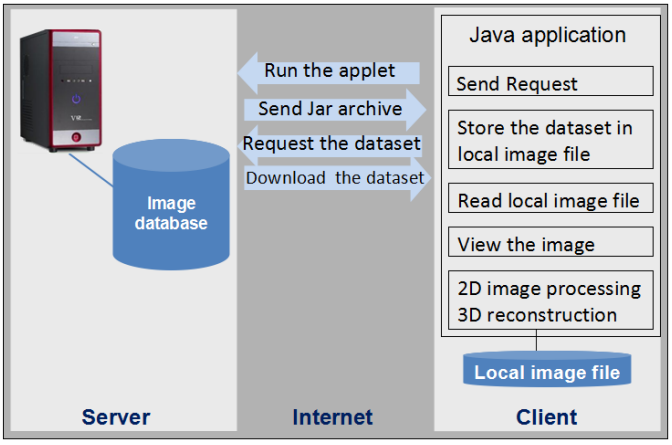


Fig. 1 The operation of the application. In this architecture, the client initially sends request to the server and the server subsequently sends back the Jar file. When the application is launched, a selected dataset is sent to the client and stored in the local file system. After that, the interpretation of the data can be performed offline.

C. Application Implementation

Access to the local file system

Java applications are usually executed in a sandbox prohibiting access to the local system. This mechanism is designed to protect private data on the computer. However, in some cases there is a need to interact with the local file system, e.g. reading and writing files on the hard disc. Thus, Java introduces the ability of signing and verifying applications to provide a safe way to directly read and write data to and from the local file system provided that the action is sanctioned by the user.

When the signed Java application is loaded, a security warning box pops up to ask the user if they agree to allow the application to access to the local system. Once the user approves, the signed Java application is able to download the file to the local computer. When the download is completed, the first slice in this dataset is automatically displayed on the screen (see Fig.2 a)). The user can then use 2D and 3D tools to assist interpretation and analysis of the downloaded dataset.

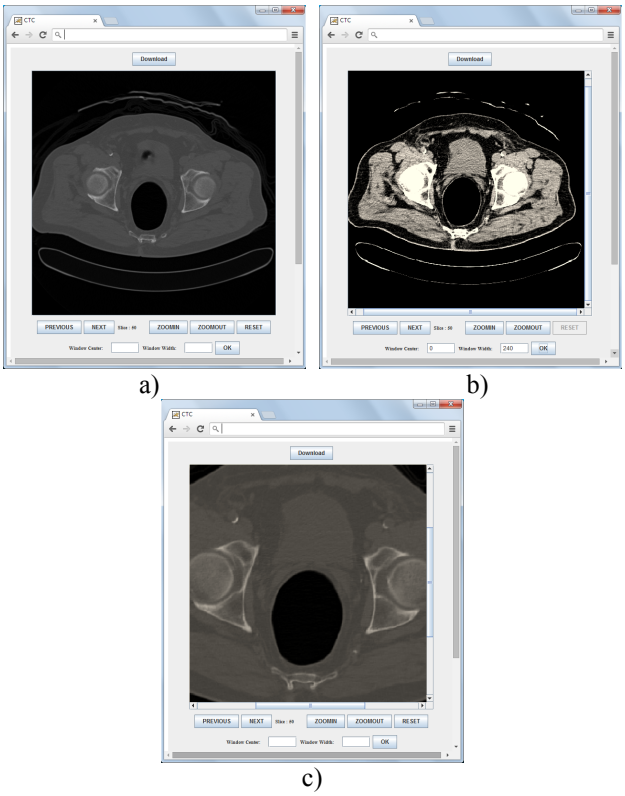


Fig. 2 The graphical user interface for 2D functions. a) displaying the first slice in the dataset. b) windowing. The image is a slice adjusted window where the following parameters have been used: centre is 0HU, width is 240HU. c) image magnification.

Image processing

The essential image-processing functions for radiology interpretation are provided by the application:

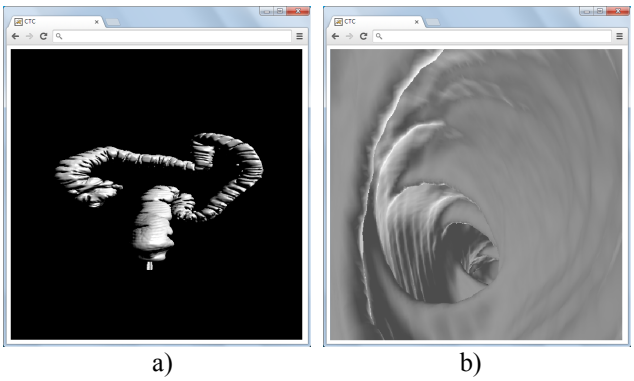
- Windowing (see Fig.2 b)) performs a linear stretching of the image histogram and is a useful tool for the enhanced visualisation of selected anatomical structures in the image. By setting parameters of window centre and window width, it is possible to view a specific range of object densities. This function is achieved by pixel-level manipulation in Java.
- A tool for zooming can be used to produce a finer detail version of a specific region in the image. A magnified image produced by the application is shown in Fig. 2 c).

3D reconstruction

3D reconstruction is an extremely computationally intensive operation. In the case of a single CTC dataset, the application needs to render approximately 800,000 polygons to generate the entire colon surface. Therefore, this operation is usually implemented on a workstation computer with hardware acceleration. Thanks to the introduction of Java3D, a low-level and GPU-accelerated API, Java-based applications are able to provide a fast 3D rendering through the Internet. Our current 3D reconstruction implementation is based on Java3D.

In this implementation, the 3D reconstruction is accomplished by surface rendering. The Marching Cubes algorithm [10] is used to extract the relevant isosurface from a volumetric dataset and this process is implemented once only. Subsequently, the information pertaining to the vertices and the normals are stored on the server. Therefore, for each client, it is only necessary to generate a 3D model according to the existing vertices and normals, instead of implementing the whole 3D reconstruction algorithm. Fig 3 a) shows a 3D colon model extracted from a CTC dataset. The user can also interact with this model and perform operations, such as rotation and translation, using the mouse.

In addition to standard 3D visualisation functions, image interpretation also benefits from virtual fly-through visualisations in situations where there are large hollow areas, e.g. the esophagus, the gastric cavity, the colon, etc. The function of virtual fly-through navigation is a time-efficient feature that allows the camera to move along a predefined path within the colon. This path is commonly referred to as the colon centreline. An example of a fast centreline calculation algorithm is given in [11]. With the sequential movement of the camera, the user can easily observe the inner wall of the colon. Fig 3 b) shows a single frame of 3D fly-through inside the colon implemented by our application.



a) b)

Fig. 3 The graphical user interface for 3D functions. a) A surface rendering of the entire colon. This model can be rotated using a mouse. b) A single frame of virtual fly through within the colon.

IV. EXPERIMENTAL RESULTS

The experiments were carried out using a ThinkPad T410i computer equipped with Intel(R) Core(TM) i3 CPU @ 2.27GHz, 2.0 GB RAM and Windows 7 32-bit operating system.

A. Data description

Three datasets were used in the experiments. These were acquired using a 16 slice Siemens CT scanner. A complete description of these datasets is provided in Table I.

B. Experimental results

An evaluation of this Java-based application is implemented using a 10Mbps local area network（LAN）. All of three datasets are used in the experiments. The performance of the application over the LAN is described in Table II. As shown in this table, the complete transmission time of a dataset over the LAN is approximately 30 seconds. After downloading the dataset from the server, the application displays a slice from the dataset and implements 2D/3D processing functions on the local computer. In terms of 2D functions, the response time for all 2D tools is <0.05s. In the case of the 3D functions, there is a minor difference in performance for each of the datasets. This is because 3D performance is related to the number of faces extracted from the surface i.e. where the number of faces is greater, the reconstructions are slower and the frame rates lower. Dataset 1 has smaller number of faces for the 3D model and consequently it has a faster reconstruction time and higher frame rate.

TABLE I
DESCRIPTION OF THE DATASETS USED IN THE EXPERIMENTS

Dataset	Type	Size	3D reconstruction		
			Vertex file	**Normal file**	**The number of faces**
#1	CT	512 × 512 × 267 (136,704KB)	26,390KB	26,390KB	750,634
#2	CT	512 × 512 × 260 (133,120KB)	33,214KB	33,214KB	944,738
#3	CT	512 × 512 × 239 (122,368KB)	27,492KB	27,492KB	781,986

Table II
THE PROPOSED APPLICATION PERFORMANCE IN LAN

2D function					3D function			
Dataset	**Download a dataset**	**View one slice (Average time)**	**Windowing (centre=0, width=240)**	**Magnify (x1.2)**	**Download vertex and normal files**	**Generate 3D model**	**Total time**	**Frame rate of fly-through**
# 1	24s	0.031s	0.031s	0.016s	2.95s	0.14s	3.09s	19.24
# 2	44s	0.032s	0.047s	0.016s	4.94s	0.19s	5.13s	16.00
# 3	27s	0.031s	0.047s	0.017s	3.37s	0.17s	3.54s	19.20

V. CONCLUSIONS

The Java-based medical imaging application presented in this paper has the ability to provide functions that are necessary for remote radiological interpretations, offering excellent performance for remote-access users, especially in the case of 3D visualisation / fly-through navigation and it is clear from the results presented in this paper that Java is an excellent choice for implementing distributed medical imaging applications.

ACKNOWLEDGMENT

We wish to acknowledge contributions from our medical colleagues from the Gastrointestinal Unit and Department of Radiology at the Mater Misericordiae Hospital in Dublin, particularly Dr Padraic MacMathuna and Dr Helen Fenlon.

REFERENCES

[1] P. Wunderbaldinger, W. Schima, K. Turetschek et.al, "World Wide Web and Internet: applications for radiologists," European Radiology, vol. 9, pp. 1170-1182, February 1999.

[2] J. Eng, "Improving the Interactivity and Functionality of Web-based Radiology Teaching Files with the Java Programming Language," RadioGraphics, vol. 17, pp. 1567-1574, November 1997.

[3] S.K. Lee, C.H Peng, C.H Wen, S.K. Huang, and W.Z. Jiang, "Consulting with Radiologists outside the Hospital by Using Java," RadioGraphics, vol. 19, pp. 1069-1075, July 1999.

[4] A. Abrardo, and A.L. Casini, "Embedded Java in a web-based teleradiology system," Internet Computing, vol. 2, pp. 60 – 68, 1998.

[5] P.J. Slomka, E. Elliott, and A.A. Driedger, "Java-Based Remote Viewing and Processing of Nuclear Medicine Images: Toward "the Imaging Department without Walls"," Journal of Nuclear Medicine, vol. 41(pt 1), pp. 111-118, January 2000.

[6] P. Knoll, K. Holl, S. Mirzaei, K. Koriska, and H. Kohn, "Distributed nuclear medicine applications using World Wide Web and Java technology," European Radiology, vol. 10(pt 9), pp. 1483-1486, February 2000.

[7] J.W. Wallis, "Java and Teleradiology," Journal of Nuclear Medicine, vol. 41(pt 1), pp. 119-122, January 2000.

[8] P.J. Pickhardt, "Three-dimensional endoluminal CT colonograhy (virtual colonoscopy): comparision of three commercially available systems," AJR Am J Roentgenol, vol. 181, pp. 1599-1606, 2003.

[9] H. Greess, A. Nömayr, B. Tomandl et al., "2D and 3D visualisation of head and neck tumours from spiral-CT data," European Journal of Radiology, vol. 33, pp. 170-177, 2000.

[10] W.E. Lorensen and H.E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," ACM SIGGRAPH Computer Graphics, vol. 21, pp. 163-169, 1987.

[11] R.J.T. Sadleir and P.F. Whelan, "Fast colon centreline calculation using optimised 3D topological thinning," Computerized medical imaging and graphics, vol. 29, pp. 251-258, June 2005.