

A Remote Access CT Colonography Training System

Vincent Luauté

A thesis submitted as a requirement for the degree
of Master of Engineering in Electronic Engineering

Supervised by Dr Robert Sadleir

School of Electronic Engineering
Dublin City University

August 2007

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Master of Engineering is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:_____

ID No:_____

Date:_____

Acknowledgements

I wish to express my sincere gratitude to my supervisor, Dr. Robert Sadleir for his guidance and support throughout my research. I wish to thank John O'Halloran and Ji Zhanlin for their contribution to this project. I also thank my colleagues of the Vision Systems Group, particularly Dr. Tarik Chowdhury and Patrica Moore for their assistance. I also 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. Helen Fenlon and Dr. Paraic MacMathuna.

Abstract

Remote Access CT Colonography Training System

Vincent Luauté

Under the supervision of Dr. Robert Sadleir at Dublin City University

A thesis submitted as a requirement for the degree of Master of Engineering in
Electronic Engineering., 2006

Computed tomography colonography (CTC) is emerging as an alternative to conventional colonoscopy (CC). However CTC is not yet in widespread use due in part to the lack of suitably trained radiologists. We have developed a novel remote access system to train radiologists for colorectal cancer screening using CTC. To ensure that radiologists can gain the relevant experience without the need for any specialist equipment or software, we opted for designing a system that is accessible via the Internet using a standard browser. The interface lets the user locate and characterise polyps with the help of appropriate tools such as windowing, polyp measurement, zooming and a 3-D view. Each user has an account in order to allow monitoring of their training. They can also run an automatic evaluation of their work based on gold standard information previously gathered from specialists. This thesis also describes an initial implementation exclusively made up of Java Servlets. The evaluation of this system has been discussed in order to determine a better approach. The final system has been developed using a combination of Java Servlets and Applets. This approach offers fast response time to the user-interface. An iteration of lumen tracking using the system takes approximately 45 seconds. This research has yielded an operational system that meets the needs of remote access users.

Table of Contents

Chapter 1. Introduction	1
1.1 The need for CTC training	1
1.2 A remote access training solution	2
Chapter 2. Background	4
2.1 Colorectal cancer	4
2.2 Computed tomographic colonography.....	4
2.3 Appropriate CTC training	6
2.4 Previous work	7
2.5 Java technologies	9
Chapter 3. Approach	10
3.1 CT dataset constitution	10
3.1.1. Dataset header	10
3.1.2. Dataset body.....	11
3.2 System requirements	14
3.3 Summary	15
Chapter 4. Implementation.....	17
4.1 Training system made up of server side programs	17
4.1.1. Introduction.....	17
4.1.2. Design of the system with server side programs	18
4.1.2.1. Operation of the system	18
4.1.3. Implementation of the system with server side programs	19
4.1.3.1. Data exchange between server and client	19
4.1.3.2. Layout using CSS	21
4.1.3.3. Home page	22
4.1.3.4. Extract and display axial images.....	23
4.1.3.5. Navigation.....	25
4.1.3.6. Flagging potential polyps.....	27
4.1.3.7. Polyp's marks.....	28
4.1.3.8. Density setting	28
4.1.3.9. Zoom setting	30
4.1.3.10. Result panel.....	31

4.1.4. Conclusion on the implementation using Java Servlets.....	32
4.2 Training system made up of Servlets and Applets.....	33
4.2.1. Introduction.....	33
4.2.2. Design of the system with Applets and server side programs	33
4.2.3. Implementation using Applets and server side programs	35
4.2.3.1. Dataset compression	35
4.2.3.2. Creation of a simulated scout x-ray	41
4.2.3.3. Digitally signing a Java Applet.....	45
4.2.3.3.1. To generate a Jar archive	45
4.2.3.3.2. The hash algorithm (Secure Hash Algorithm) SHA-1.....	46
4.2.3.3.3. To generate a RSA key pair	46
4.2.3.3.4. Digitally signing the Jar archive	47
4.2.3.3.5. Receive and verify a digitally signed Jar archive	48
4.2.3.4. Dataset selection	50
4.2.3.5. Multi-user server.....	51
4.2.3.5.1. Encapsulate a dataset into a Jar file	52
4.2.3.5.2. Server using sockets.....	52
4.2.3.5.3. Server using a Servlet	54
4.2.3.6. A separate thread to display the downloading progress.....	56
4.2.3.7. Retrieve 2-D slices on the client's side.....	58
4.2.3.8. Generate an axial image.....	60
4.2.3.9. Reformatted images	61
4.2.3.9.1. Generate a coronal image.....	61
4.2.3.9.2. Generate a sagittal image	62
4.2.3.9.3. Realistic reformatted image	64
4.2.3.9.4. Region of interest for reformatted images	65
4.2.3.10. Dataset navigation.....	67
4.2.3.11. View tabs and synchronisation	69
4.2.3.12. Measurement.....	70
4.2.3.13. Zoom Window	71
4.2.3.14. 3-D Visualisation	72
4.2.3.15. Gold standard Evaluation.....	76
4.2.3.16. Display all pictures of a specific polyp category	79
4.3 Multi-user architecture.....	82

4.3.1. Introduction.....	82
4.3.2. Login panel	82
4.3.3. User registration.....	83
4.3.4. User possibilities	84
4.3.5. Result page.....	86
4.3.6. Administration panel.....	87
4.3.7. Conclusion for the implementation using Servlets and Applets.....	89
Chapter 5. Testing & Results	90
5.1 Testing.....	90
5.1.1. Introduction.....	90
5.1.2. Test of the automatic evaluation	90
5.1.3. Response time of the interface	92
5.1.3.1. Response time of axial images.....	92
5.1.3.2. Response time of reformatted images.....	93
5.1.4. Transfer time of CT datasets between server and client	93
5.2 Discussion	94
Chapter 6. Conclusion.....	96
6.1 Initial implementation	96
6.1.1. Speed problem	96
6.1.2. Usability problem.....	96
6.2 Revised implementation.....	97
6.2.1. Better response time.....	97
6.2.2. Advanced features.....	97
6.2.3. Evaluation strategy.....	98
6.3 Future work.....	99
References	101
Conferences and Publications	107

List of figures

Figure 1-1.	CT scanner in the Mater hospital (Dublin)	2
Figure 1-2.	Client-Server architecture. The system can be accessed from every computers connected to the server.	3
Figure 2-1.	Modern analysis tools usually provide different type of 2-D image visualisation. These types include reformatted image (coronal and sagittal) in addition to axial images.	5
Figure 3-1.	Overview of the CT Dataset used on this system. The first 24 Bytes of a dataset constitute the header and contains information about the format. The rest of the dataset constitute the body and contains the CT axial slices.	10
Figure 3-2.	A 2-D axial image extracted from a CT dataset. A CT dataset can be represented as a stack of 2-D axial images.	12
Figure 3-3.	Overview of the density range of a CT Dataset. Voxels in a dataset have a value in the range of -1024 to 2400. -1024 corresponds to the lowest density e.g. air and 2400 corresponds to the highest density e.g. metal implant.	12
Figure 3-4.	Number of voxels per density value. This graphic illustrates the most recurrent density values in CT datasets.	13
Figure 3-5.	2-D axial images with different density settings.....	14
Figure 4-1.	Block diagram of the architecture of the system	17
Figure 4-2.	Operation of the system.	19
Figure 4-3.	Overview of data exchange between server and client. The server sends HTML pages to the client. The client sends “post data” information to the server via HTML forms.	20
Figure 4-4.	The home page of the system. On this HTML page, the user can select a dataset and a profile before starting the system.	22
Figure 4-5.	2-D axial images in a CT dataset. A dataset can be represented as a cube made up of a stack of axial images.....	23
Figure 4-6.	The trainee’s interface with a potential polyp (circle). On this interface, the user can navigate through the axial images using the horizontal bar, the user can also use the vertical bar on the left to select a specific density range and use the different options on the right of the interface to zoom in or zoom out.....	25
Figure 4-7.	Select and display an axial image. This flowchart describes the operation of the different image navigation features.	26

Figure 4-8. Flagging a potential polyp. If the user clicks on the axial image, a circle is drawn around the coordinates of the mouse click and a horizontal toolbar appears at the bottom of the interface. On this tool bar, the user can select the type and the size of the new polyp.	28
Figure 4-9. Axial image with different density settings. The user can select an appropriate density range using the vertical tool bar at the left of the interface.....	29
Figure 4-10. Zoom on a potential polyp. On the right of the interface, the user can select an appropriate zoom area and zoom factor.....	30
Figure 4-11. The result panel displaying a true-positive polyp. In this example, a polyp flagged by the trainee and the corresponding polyp from the gold standard. When the trainee runs the automatic evaluation, a colour code is used on the navigation bar at the top of the interface to indicate the true-positive (white), false-positive (yellow) and the false-negative (red) polyps. Also on this page, the user can use drop down menus to display the false-positive and false-negative polyps.....	31
Figure 4-12. Overview of data exchange. The server sends HTML pages to the client and the client returns post data information. When the Applet is started, the Applet communicates with the server using data streaming.....	34
Figure 4-13. Operation of the system. This flowchart highlights the operation difference between a trainee and a specialist using the system.	35
Figure 4-14. Difference between two consecutive axial images. This image highlights the fact that sending a difference image represents less data transfer than sending an entire axial image.....	36
Figure 4-15. Generate a simulated scout x-ray. The resulting image is obtained by calculating the average density on Y.	41
Figure 4-16. Administration tool to generate a simulated scout x-ray. The administrator can use 4 sliders in order to highlight the colon on the scout x-ray.	42
Figure 4-17. Focusing on the appropriate region on Y. It is possible to focus on the relevant region around the colon and prevent the remaining information to overload the scout x-ray.....	43
Figure 4-18. Generate a simulated scout x-ray. This flowchart illustrates the process used to calculate the average density of a chosen region of the CT dataset and store the result on a JPEG image.	44
Figure 4-19. Pack the Applet files into a Jar archive.....	46
Figure 4-20. Digitally signing the Jar archive. A SHA-1 algorithm hashes each file	

of the Jar and the remaining digests are stored inside the file VSG.SF after being signed using the RSA private key.....	47
Figure 4-21. Verifying a Jar archive. When the Jar is received on the client side, digests stored in the file VSG.SF after decrypted using the RSA public key. These decrypted digests are then compared one by one with the digests calculated on each file of the Jar. If each digest is identical, it means that the Jar hasn't been altered during its transfer.....	49
Figure 4-22. Security warning before starting the Applet. The user is informed that the Applet hasn't been digitally signed by a CA and asks the user if he/she wants to trust the named publisher.....	49
Figure 4-23. Dataset selection. On this page, the user can select the dataset he/she wants to train on.....	50
Figure 4-24. Dataset selection. This flowchart illustrates the process used to display the relevant scout x-ray.....	51
Figure 4-25. Sending a selected dataset from the server to the client.....	53
Figure 4-26. Handle connection of several clients. The server classes welcomes a new user and passes the corresponding.....	53
Figure 4-27. Interface of the server.....	54
Figure 4-28. Sending a selected dataset from the server to the client.....	55
Figure 4-29. Downloading progress for a specialist.....	56
Figure 4-30. Downloading progress for a trainee.....	57
Figure 4-31. Decoding process. This flowchart illustrates how each axial image is decoded bit by bit when it is received on the client side.....	59
Figure 4-32. Flagging a potential colorectal cancer polyp. When the user clicks on a 2-D slice, a dedicated tool bar appears at the bottom of the interface and the user can select his/her level of confidence, the type and the size of the new polyp.....	60
Figure 4-33. A 2-D coronal image in a CT dataset. There are as many coronal images as there are pixels along the Y axis.....	61
Figure 4-34. Coronal image.....	62
Figure 4-35. A 2-D sagittal image in a CT dataset. There are as many sagittal images as there are pixels along the X axis of an axial image.....	63
Figure 4-36. Sagittal image.....	63
Figure 4-37. Stretching a coronal image to realistic proportions.....	64
Figure 4-38. Stretching a sagittal image to realistic proportions.....	65

Figure 4-39. Axial image and region of interest of a coronal image.....	66
Figure 4-40. Simulated scout x-ray illustrating the position of one of the first, middle and last axial image in the dataset	68
Figure 4-41. Simulated scout x-ray illustrating the position of one of the first, middle and last coronal images of the dataset.	68
Figure 4-42. Simulated scout x-ray illustrating the position of one of the first, middle and last sagittal image of the dataset.	69
Figure 4-43. Image synchronisation. A polyp flagged on a supine-axial image is automatically reported on a supine-sagittal image and the system focuses on the relevant region of interest	70
Figure 4-44. Zoom window and measurement tool. The length between two mouse clicks is indicated at the bottom of the interface.....	71
Figure 4-45. Observer watching the projection of a volume on a projection plane.	72
Figure 4-46. Projection of a cube on a projection plane (viewed from above). Ray-casting is used to represent a volume on a 2-D image using distances between the observer and the volume's voxels.....	73
Figure 4-47. A 3-D rendering of the inner surface of the colon. This image has been generated with a threshold equal to -200 HU. A polyp is visible in the middle of this image.	74
Figure 4-48. 3-D rendering of a dataset subsection with a threshold superior to 350 HU. With this threshold value, the 3-D algorithm offers a view of the skeleton e.g. a section of the spine.....	75
Figure 4-49. The 3-D interface. After the user has specified a region of interest, a volume is displayed in a floating window.	75
Figure 4-50. Generate a 3-D view from a zoomed area.	76
Figure 4-51. The evaluation panel. On this panel, a colour code is used on the navigation bar, on the 2-D images and on the scout x-ray to indicate the true-positive (green), false-negative (blue) and false-positive (red) polyps.	77
Figure 4-52. Polyp Category Object.....	78
Figure 4-53. Floating window displaying polyp thumbnails of a specific category. After the running the automatic evaluation, the user can click on a tab at the top of the scout x-ray window to display thumbnails of a specific category.	79
Figure 4-54. Calculating the number of rows and columns in the scout x-ray window.	80

Figure 4-55. Find the selected thumbnail from the coordinates of the mouse click.	81
Figure 4-56. Home page of the system.	83
Figure 4-57. The registration panel. A new user can register and enter the relevant information on this page.	83
Figure 4-58. Multi-user system. This flowchart gives an overview of the operation of the system that surrounds the applet.	85
Figure 4-59. Option page	86
Figure 4-60. Result page. This page shows the number of true-positive, false-negative, false-positive and the reading time for each dataset previously studied.	87
Figure 4-61. The administration panel. After a new user register on the system, an email is sent to the administrator. The administrator can then use this page to create the new account, delete an existing account or display the results of a trainee.	88

List of tables

Table 4-1.	Overview of the time required for the different tasks.....	32
Table 4-2.	Symbols encoded using Huffman codes.....	39
Table 4-3.	Huffman codes array.....	40
Table 4-4.	Average time (in second) to generate reformatted images with “skip Bytes” and “readFully” methods.	65
Table 4-5.	Average time (in second) to generate different region of interests.....	67
Table 5-1.	Result displayed on the evaluation panel after comparing coordinates of polyps found by a trainee to the gold standard.	91
Table 5-2.	Response time (in seconds) for the different image.....	92
Table 5-3.	Transfer time of CT datasets.....	93
Table 5-4.	Transfer time of CT datasets using the a standard HTML port.	94

Chapter 1. Introduction

1.1 The need for CTC training

At present the most sensitive colorectal cancer screening technique is conventional colonoscopy (CC). This involves an endoscopic examination of the colonic mucosa using an instrument known as a colonoscope. The examination itself is extremely invasive and can lead to complications. The CC examination is embarrassing and uncomfortable for the patient and has achieved limited acceptance among those at risk of developing colorectal cancer.

Computed tomographic colonography (CTC) is an emerging technique for colorectal cancer screening. A CTC examination consists in doing an abdominal computed tomography (CT) study using a CT scanner (figure 1-1). One of the main benefits associated with CTC is that it is potentially more patient friendly than CC due to the fact that it is minimally invasive. Although CTC has been demonstrated to have similar sensitivity to CC for the detection of significant polyps¹ (Fenlon et al. 1999, Pickhardt et al. 2003) it is not yet in widespread clinical use.

This is partly due to the fact that only a limited number of radiologists have the skills required to perform a CTC examination. The need for CTC training has been highlighted in the literature (Cotton et al. 2004) particularly for problem cases e.g. flat polyps (Fidler et al. 2004). Also, a recent study conducted by Fisichella et al. (2006), shows that the lack of CTC training is the main reason why some Swedish departments of radiology don't use CTC yet.

1) A polyp is a flat or grape-like growth tissue into organ such as the colon.



Figure 1-1. CT scanner in the Mater hospital (Dublin)

Similarly in a paper related to the evolution of CTC, Dachman et al. (2003) explains that the “steep learning curve has led researchers to advise against widespread colorectal cancer screening with virtual colonoscopy outside of academic centers with experienced readers”. This highlights the fact that accessibility to suitable training resources is an issue that needs to be addressed.

1.2 A remote access training solution

The system described in this thesis has been developed specifically to deal with the problem of accessibility to CTC training. This system allows the trainee to evaluate a CTC dataset¹ by flagging locations that they suspect to be potential polyps.

1) A CT Dataset contains computerized tomographic image data generated by a CT scanner using x-rays.

Once the trainee has completed their evaluation the results are compared against the gold standard¹. To ensure radiologists gain the relevant experience without any computer-related issues, the system should operate without the need for specific software packages. This can be achieved by using a client-server architecture over a network as illustrated in figure 1-2.

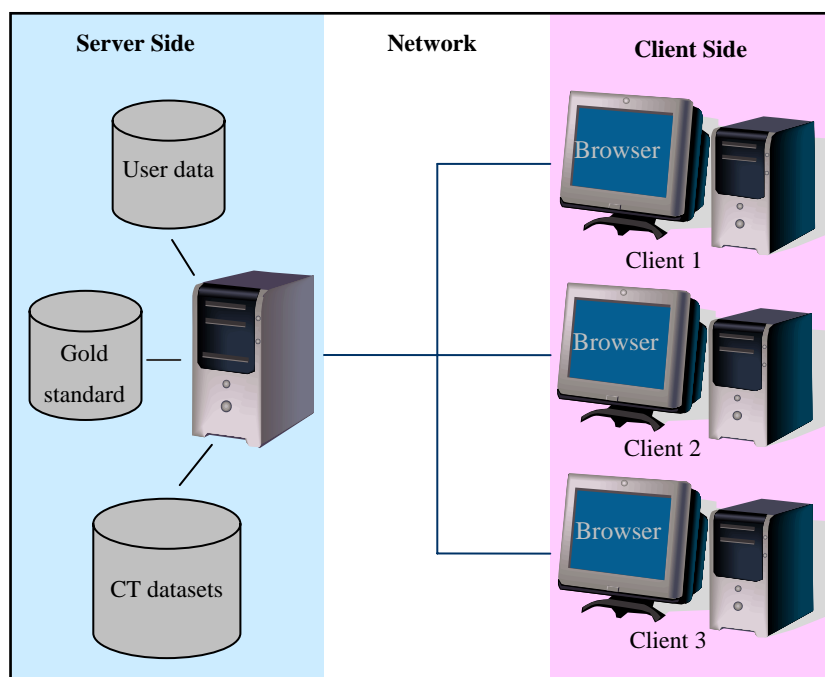


Figure 1-2. Client-Server architecture. The system can be accessed from every computers connected to the server.

This approach allows the trainee to access the system remotely from a standard web browser e.g. Mozilla Firefox or Microsoft Internet Explorer and provides the highest level of compatibility as it can be used on every operating system. In addition patient confidentiality is maintained as no direct access is provided to patient data. Instead this data is accessed via a server which strips all the sensitive patient information leaving only the image data.

1) In this case, the gold standard designates either conventional colonoscopies or CT colonoscopies that have been conducted by specialists and against which trainee's studies are compared.

Chapter 2. Background

2.1 Colorectal cancer

Colorectal cancer is a major cause of cancer related death in developed countries. Statistics published by Campo et al. (2004) indicate that colorectal cancer is the leading type of cancer in Ireland.

Colorectal cancer usually begins as adenomas¹ of the colonic mucosa (Konishi et al. 1982). Adenomas are usually classified according to their appearance as either sessile (flat) or pedunculated (having a stalk). According to a study by Villavicencio et al. (2000) men have a 1.5 relative risk of adenomas compared with women. Age, male gender, and first-degree family history of colorectal cancer are risk factors for adenomas. Polyps such as adenomas less than or equal to 5mm grow very slowly and are unlikely to be malignant at this stage. However, the probability of cancer increases with the size of the polyp and within four to eight years, colorectal cancers may arise through a multi-stage process called the adenoma-carcinoma sequence (Hill et al. 1978, Morson et al. 1983). Colorectal cancer can be prevented if precursor polyps are detected early in their course and successfully resected. This highlights the fact that regular screening is required for high risk populations in order to reduce the mortality from this disease.

2.2 Computed tomographic colonography

CTC is a relatively new technique for colorectal cancer screening that was introduced by Vining et al. (1994). Before performing a CT scan, the patient's bowels must be distended. This is achieved by insufflating room air or CO₂ via the rectum (Klein 2003). The resulting CT dataset can then be examined, either as a sequence of 2D slices or as a reconstructed 3D model of the colon, for the presence of colorectal polyps.

1) Adenoma is a benign tumour made up of glandular tissue such as colon.

Three types of 2-D slice can be generated from a CT datasets.

- Axial image is the most widespread type of 2-D image. This is explained by the fact that CT scans are usually performed in the axial plane along the spine. Therefore, the extraction of an axial image from an axial CT scan is a straight forward operation (see figure 2-1.a).
- Coronal images can be rendered by computer reconstruction. This type of image represents slices of the human body from the front to the back as illustrated in figure 2-1.b.
- Sagittal images can also be rendered by computer reconstruction and can be described as images obtained by slicing the human body from side to side as illustrated in figure 2-1.c.



a) Axial image

b) Coronal image

c) Sagittal image

Figure 2-1. Modern analysis tools usually provide different type of 2-D image visualisation. These types include reformatted image (coronal and sagittal) in addition to axial images.

Even if conventional colonoscopy is known as the most sensitive colorectal cancer screening technique, CC is not perfect and a study observed that the overall miss rate for adenomas is 24% (Rex et al. 1997). Also, results obtained by CTC are likely to improve with the natural progression of technology.

Moreover, researches are made to find out how to optimize CTC studies. For example, polyp detection using CTC can be significantly improved by using two datasets, one in supine and one in prone position (Fletcher et al. 2000).

One of the main benefits associated with CTC is that it is potentially more patient friendly than CC due to the fact that it is minimally invasive. Furthermore, there is no need for sedation and the patient can go back to work on the same day as the examination, unlike CC where the patient is required to take a full day off work. Besides, a study by Macari et al. (1999) demonstrated that CTC is particularly useful for patients who fail to undergo conventional colonoscopy. This study estimates that the percentage of incomplete conventional colonoscopies is 10%.

2.3 Appropriate CTC training

The learning curve associated with CTC has been highlighted in different studies (Gluecker et al. 2002, Soto et al. 2005). For example, Gluecker has observed that “the quality of data interpretation is directly related to the experience of the physician”. Gluecker also found that the study of 50 patients contributes to “significant improvement in specificity, an important decrease of false-positive findings and of the time for data interpretation” but “are not sufficient to achieve sensitivity in the range of 80%”. In the same way, Taylor et al. (2004) state that in order to gain proficiency in CTC it is necessary for a radiologist to be trained using a large number of datasets (50-100). These datasets should contain a mix of polyps, cancers, normal features and pseudo polyps e.g. a protruding ileocecal valve¹.

In a study intended to establish the general consensus regarding CTC training guidelines Soto et al. (2004) sent out a questionnaire to 20 international experts in the field. The outcome of their survey suggested that the most appropriate method for reader training would consist of lectures and supervised hands-on workstation training with 40 to 50 cases (20% of which should be normal). Also, Soto et al. (2005) believe that the required training should be integrated into the relevant radiology residency and fellowship programmes. However until that happens a radiologist interested in acquiring competency in CTC must undertake a specific training course or participate in dedicated fellowships.

1) An ileocecal valve prevents contents from flowing back from the colon to the small intestine.

2.4 Previous work

This research project continues on from previous work by the Vision Systems Group in the area of CTC e.g. colon centreline calculation at CTC (Sadleir et al. 2004 a) and computer aided detection (CAD) at CTC (Sadleir et al. 2002) and also work in the area of medical image interpretation (Sadleir et al. 2004 b). This previous research provides a solid background which has been used to develop this remote access system for CTC training.

Some systems already provide remote access to medical images e.g. the system described by Young et al. (2004). A system described by Bohne-Lang et al. (2005) is designed to generate and display 3-D molecule structures. This system consists of generating 3-D images on the server side and sending the result in Joint Photographic Experts Group (JPEG) format (Pennebaker et al. 1993). The 3-D rendering algorithm is only implemented on the server. The main advantage of this approach is to avoid the need for any software installation on the client side. However, each time the user interacts with the interface, a new image is generated on the server and transmitted to the client. This type of system with image “on demand” has the advantage of being easily developed and deployed. Therefore it can be a good starting point for the present work.

Masseroli et al. (2004) described a system that is able to access images in a web browser environment. This system is based on client-server architecture and is implemented using Java. On the client side, a Java applet runs in the web browser. An applet has the advantage of providing more advanced capabilities to create a user-friendly interface. For instance, the system developed by Masseroli et al. gives the end user the possibility to browse and visualize different patient and medical data. When the end-user displays an image or a patient personal data via the applet interface, a query is sent to the server side. The server is responsible for handling all queries and accessing remote databases and file systems containing the requested data. The server transfers the medical data to the client through secure connections. For the project described in this thesis, a Java-based approach seems appropriate as large amounts of data need to be exchanged between the server and the client side. This data includes medical images and polyp coordinates. In addition, a Java applet would be useful for implementing appropriate features required to facilitate the identification of potential

cancer polyps. These features include windowing, polyp measurement and zooming.

Slomka et al. (2000) also propose a Java-based remote viewing station for reading and examining nuclear medicine images using a standard Internet browser. The interface of the system is a Java applet, that provides image processing tools, such as bilinear interpolation, cine display and filtering. The author of the system explains that the performance of these tools is similar to a standard imaging workstation. Moreover, this Java-based approach doesn't require any software to reside on the client computer. The support and deployment of such a system is therefore simplified. With regard to the work described in this thesis, such an approach seems very interesting as the CTC tutor is likely to be often modified and updated. This is due to the changing needs of our medical collaborators and the constant evolution of the CTC technique. The Java platform-independence also enables the utilization of the system from all sites where an intranet or internet connection is available. The same features are accessible regardless of the kinds of platforms.

In his article, Paulson (2005) describes a new web-technology call Ajax (Asynchronous JavaScript and XML). The main advantage of this technology is the possibility to request a URL without refreshing the current page of the web browser. Therefore, a web-application using Ajax can be more interactive than a traditional web-page. The term asynchronous, means that the user doesn't have to wait for the response of an HTTP request. The requested data is instead transferred to the web-application in the background while the end-user can continue using the interface of the system. Ajax applications run on the client side. A server-side program must be developed in order to send data requested by the client.

This emerging technology seems interesting as it offers more flexibility to the interface. The interface of the system described in this thesis could be developed using Ajax. However, Ajax relies on JavaScript which is often implemented differently by the browsers and their different versions. Because of this, sites that use JavaScript may need to be tested in multiple browsers to check for compatibility issues. Also, because Ajax is an emerging technology, it might be useful to first develop a full application without Ajax for Non-Ajax users.

2.5 Java technologies

The system described in this thesis has been developed using Java 2 Standard Edition (J2SE) from Sun Microsystems. J2SE is a collection of application programming interfaces (API) largely used by programmers to develop web-based applications. According to Young (2004), “recent advances in Java applet, servlet and Java Server Page (JSP) technologies now allow very sophisticated, interactive image manipulation to be done over the web.”

Java programs are executed on a computer via a Java Virtual Machine (JVM). Sun Microsystems have developed JVMs for the different operating systems (OS) which makes Java, a platform independent programming language. There are different types of Java program. We can differentiate server side and client side programs.

- Java Servlet and Java server page (JSP) are server side programs. A server side program is an extension to a server that enhances the server's functionality. In practical terms, Servlets and JSPs are commonly used to process forms and create dynamic content.
 - A Java Servlet is written in Java but can also contain hyper text mark-up language (HTML) code in order to display some information via the user's browser.
 - A JSP is usually used with an associated Java Bean. JSPs hold the code related to the interface of the system and contains HTML and Java code. The Java code of a JSP is limited to basic operations and is mainly used to exchange data with the associated Java Bean. This Java Bean is exclusively written in Java and is able to perform advanced operations like reading and writing files on the server.
- A Java Applet is a client side program that can be run in a standard web browser e.g. Firefox or Microsoft Internet Explorer. Applets provide advanced user interfaces and are able to exchange data with server side Java programs.

This system has been developed using the NetBeans integrated development environment (IDE) with the built-in versions of Apache Tomcat.

Chapter 3. Approach

3.1 CT dataset constitution

The CT datasets used on this system are generated from the DICOM images (NEMA 2003, summarised by Mildenerger et al. 2002) obtained from the original CT scan. This is achieved using the NeatMed API (Sadleir et al. 2004 b). The resulting datasets contain a header and a body as illustrated in figure 3-1.

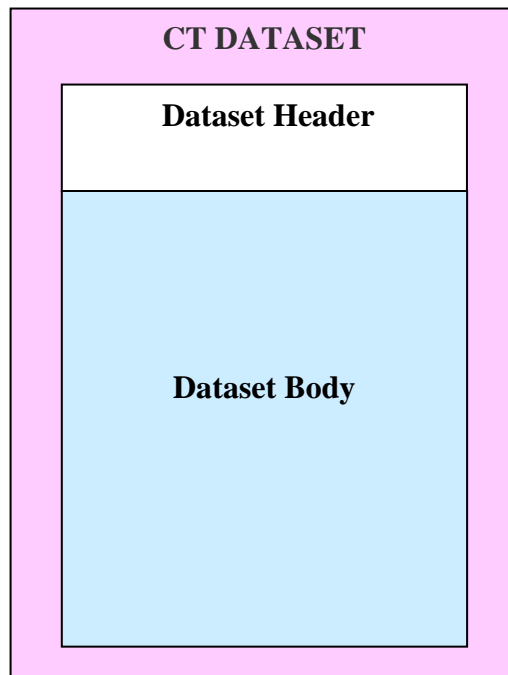


Figure 3-1. Overview of the CT Dataset used on this system. The first 24 Bytes of a dataset constitute the header and contains information about the format. The rest of the dataset constitute the body and contains the CT axial slices.

3.1.1. Dataset header

The header holds some information related to the format of the dataset and the 2-D axial images it contains. The header is made of 9 values. 6 are integers and 3 are float. Each integer is written in the header using 2 Bytes and each float is written using 4 Bytes. When reading these data with a Java program, the 2 Bytes integers are stored in “short” variables and the 4 Bytes float are stored in “float” variables.

Here is a list of Java variable containing header's data and their respective type:

- `formatID`: Identification number of the dataset format (short)
- `widthInVoxels`: Width of dataset's slices (in voxel) (short)
- `heightInVoxels`: Height of dataset's slices (in voxel) (short)
- `depthInVoxels`: Number of slices in the dataset (short)
- `minDensityValue`: Minimum density value (short)
- `maxDensityValue`: Maximum density value (short)
- `voxelWidth`: Voxel width (mm per voxel on x) (float)
- `voxelHeight`: Voxel height (mm per voxel on y) (float)
- `voxelDepth`: Voxel depth (mm per voxel on z) (float)

Dataset's header data are used in many ways on the system. For instance, the variable "depthInVoxels" gives the number of axial images in the dataset. The variables "minDensityValue" and "maxDensityValue" are used to find out the voxel¹ density range of a dataset (see equation 3.1.1).

$$\text{densityRange} = \text{maxDensityValue} - \text{minDensityValue} \quad (3.1.1)$$

"VoxelWidth" and "voxelHeight" can be used to convert the Euclidean distance measured on a 2-D slice into a length in millimetres.

3.1.2. Dataset body

The dataset's body contains the voxel values. Each voxel is written using 2 Bytes (one short). Therefore, an axial image is made of $2 \times \text{widthInVoxels} \times \text{heightInVoxels}$ Bytes. The CT dataset's body can be described as a stack of 2-D axial images as illustrated in figure 3-2.

1) Voxel is an abbreviation for "volume element" or "volume cell." It is the 3D conceptual counterpart of the 2D pixel.

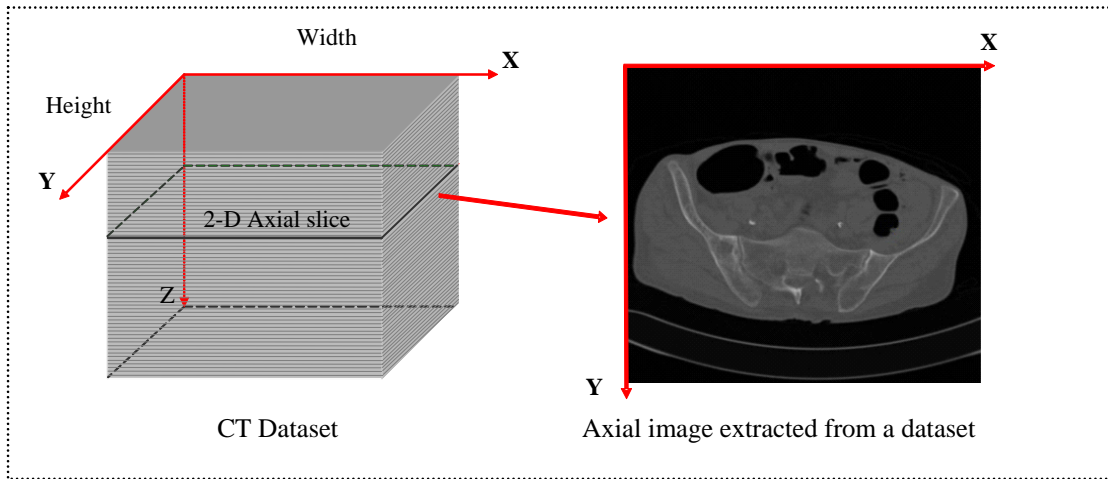


Figure 3-2. A 2-D axial image extracted from a CT dataset. A CT dataset can be represented as a stack of 2-D axial images.

Each voxel has a density value in Hounsfield units. CT images of our CT datasets have a density range of about -1024 to 2400 HU meaning that each picture can have approximately 3500 greyscales. The standard Java grey levels system divides the greyscale into 256 sections with black at 0 and white at 255. Therefore, each voxel value of a 2-D slice must be converted into a 256 greyscale value (0 to 255) in order to be displayed with Java. The equation 3.1.2 illustrates this simple conversion.

$$\text{voxel} = \frac{(\text{voxel} - \text{minDensityValue}) \times 255}{\text{maxDensityValue} - \text{minDensityValue}} \quad (3.1.2)$$

It is also possible to shrink the original voxel greyscale (setting a window width) and focus on a specific density value (window centre) to highlight information associated with the bones or lungs (see figure 3-3).

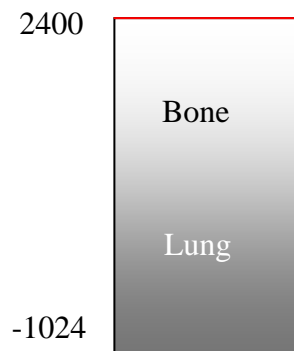


Figure 3-3. Overview of the density range of a CT Dataset. Voxels in a dataset have a value in the range of -1024 to 2400. -1024 corresponds to the lowest density e.g. air and 2400 corresponds to the highest density e.g. metal implant.

In this case the conversion of a voxel value to a 256 greyscale value is done using the window centre and the window width that have been previously defined by the user (see equation 3.1.3).

$$\text{voxel} = \frac{\left[\text{voxel} - \left(\text{window centre} - \frac{\text{window width}}{2} \right) \right] \times 255}{\text{window width}} \quad (3.1.3)$$

A small program has been developed to read the voxel values of 5 CT datasets and counts the occurrence of each density value. Microsoft Excel was used to display the results (see figure 3-4).

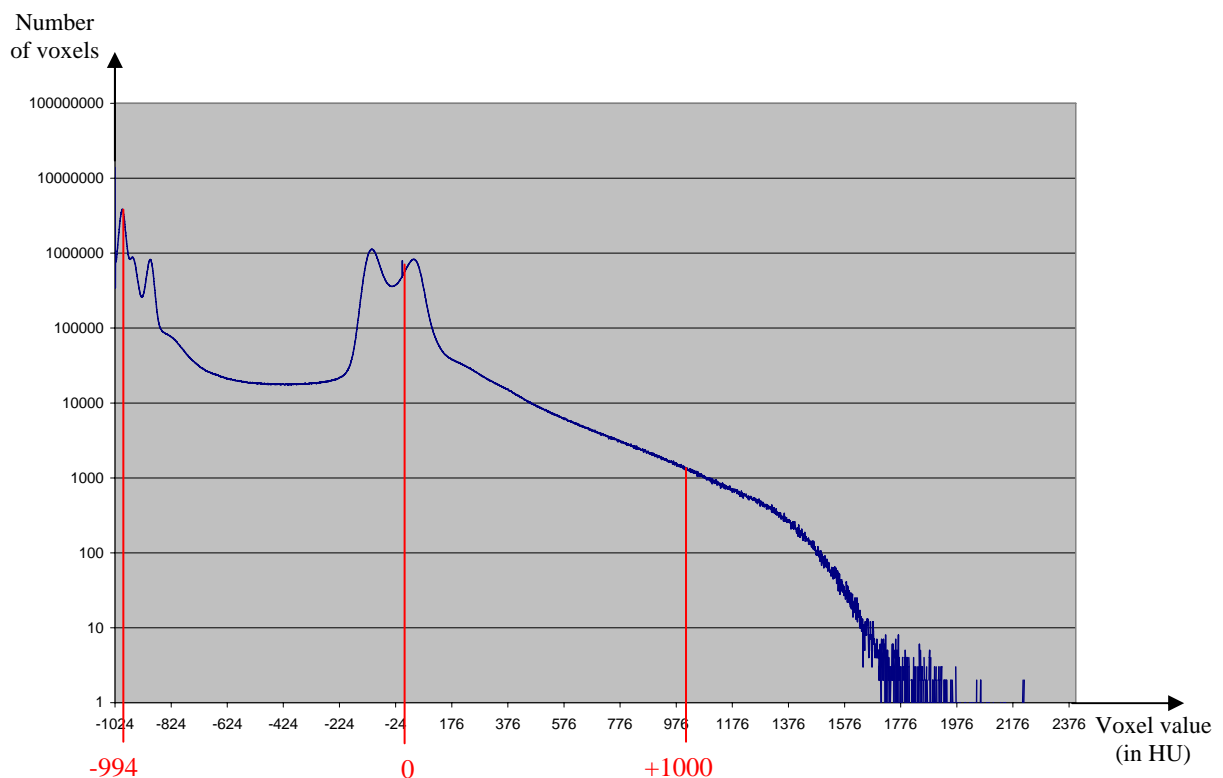


Figure 3-4. Number of voxels per density value. This graphic illustrates the most recurrent density values in CT datasets.

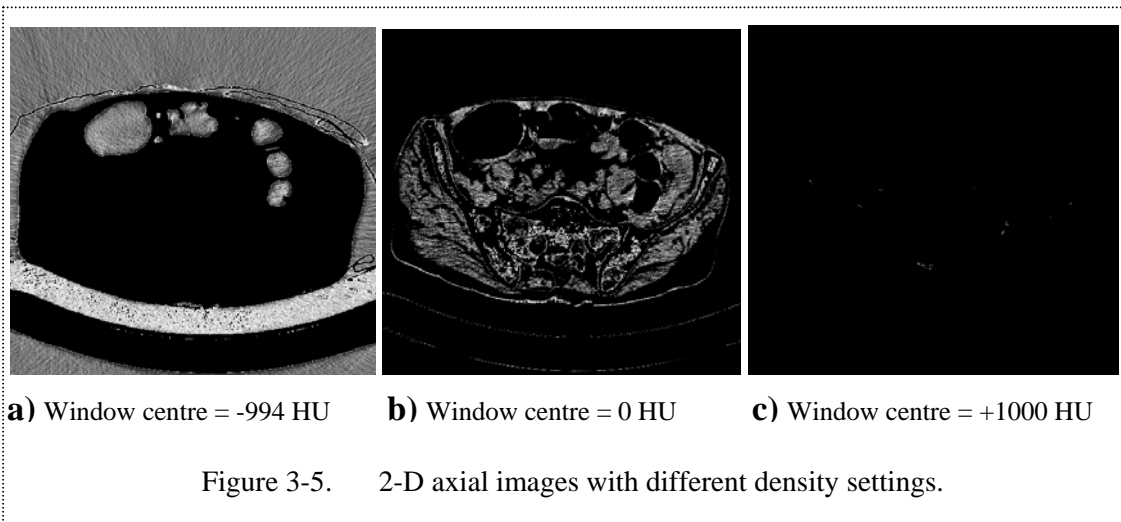
The observations of the figure 3-4 can fall into 3 parts:

- Density values lower than -824 represent air or gas. This is also illustrated in the figure 3-5.a. In this image, only ambient air outside the patient's abdomen and the gas (room air or CO₂) that has been insufflated inside the patient's colon is visible. The axial image 3-5.a has been generated using a window width equal to 100 and a window centre equal to -994. Therefore, we can calculate that the image 3-5.a is made of density values in the range of -1044 to -944 HU (see equation 3.1.4 and 3.1.5)

$$\text{min value} = \text{window centre} - \frac{\text{window width}}{2} = -1044 \quad (3.1.4)$$

$$\text{max value} = \text{window centre} + \frac{\text{window width}}{2} = -944 \quad (3.1.5)$$

- We can see in figure 3-4 that most of the significant voxels have a density value near 0. The axial image in figure 3-5.b has been generated using a window width equal to 100 and a window centre equal to 0. This image illustrates that soft tissues of a human body has a density value around 0 HU.
- Also, in figure 3-4, we can see that only few voxels have a density value superior to 1000. It is also illustrated in figure 3-5.c that has been generated using a window width equal to 100 and a window centre equal to +1000.



3.2 System requirements

Our system has been developed based on the following requirements:

1. The first target of this project is to train radiologists for colorectal cancer screening using CTC. Therefore the first step is to extract and display 2-D axial images from a CT Dataset.
2. In order to maximise the accessibility, the system must be implemented using client-server architecture. Our end-users must be able to visualise 2-D slices of CT datasets initially located on the server.
3. Once this objective has been achieved, the program must be extended in order to create a training interface. This interface must allow radiologists to locate and characterise polyps with the help of appropriate tools such as windowing, polyp

measurement and zooming.

4. When the program is able to store polyp candidates' coordinates, the next step is to develop an evaluation panel based on gold standard data previously gathered from specialists. The system should display results as true-positives, false-negatives or false-positives. In order to distinguish true-positive from false-positive polyps, the system should calculate the distance (in the 3 dimensions) between the polyps found by a trainee and the gold standard. If the distance is less than a specific margin of error threshold, this polyp is a true-positive otherwise it is a false-positive. The value of this margin of error should be equal to the length of a small polyp which is around 5mm.
5. In addition, the training system must be embedded in a multi-user architecture that provides a user identification process.

Concerning the response time of the system, our end-users should be able to browse through the different medical images in a comfortable way. Therefore the response time for displaying an image should be less than 0.5 seconds.

In order to guarantee an efficient response time, data compression can be used. However, the requirement from our medical colleagues is to only use lossless compression algorithms in order to preserve the original image quality.

3.3 Summary

The operation of our system can be divided into 3 parts.

1. First, the user enters the server URL in the address bar of his or her selected browser. The home page of the system is loaded and the user is requested to enter a login and a password.
2. Following this first step, the user can launch the training interface. In this interface the user is able to highlight polyp candidates by flagging locations via circles superimposed on 2-D axial images. Also, trainees have to define the size (in mm) and the type of the polyp e.g. "sessile" or "pedunculated".
3. Upon completion of their work, the trainee can run an automatic evaluation based on gold standard information. The evaluation panel displays the results as true-positive, false-negative and false-positive.

This approach has been achieved in two different ways. A first idea was to develop a program made of Java server-side programs (e.g. Servlets and JSPs) in order to remotely access the datasets' pictures. The observation of the results for this first version led us to develop a better implementation using a combination of Java server-side programs and Applets. Details of both implementations are described in the following chapters.

Chapter 4. Implementation

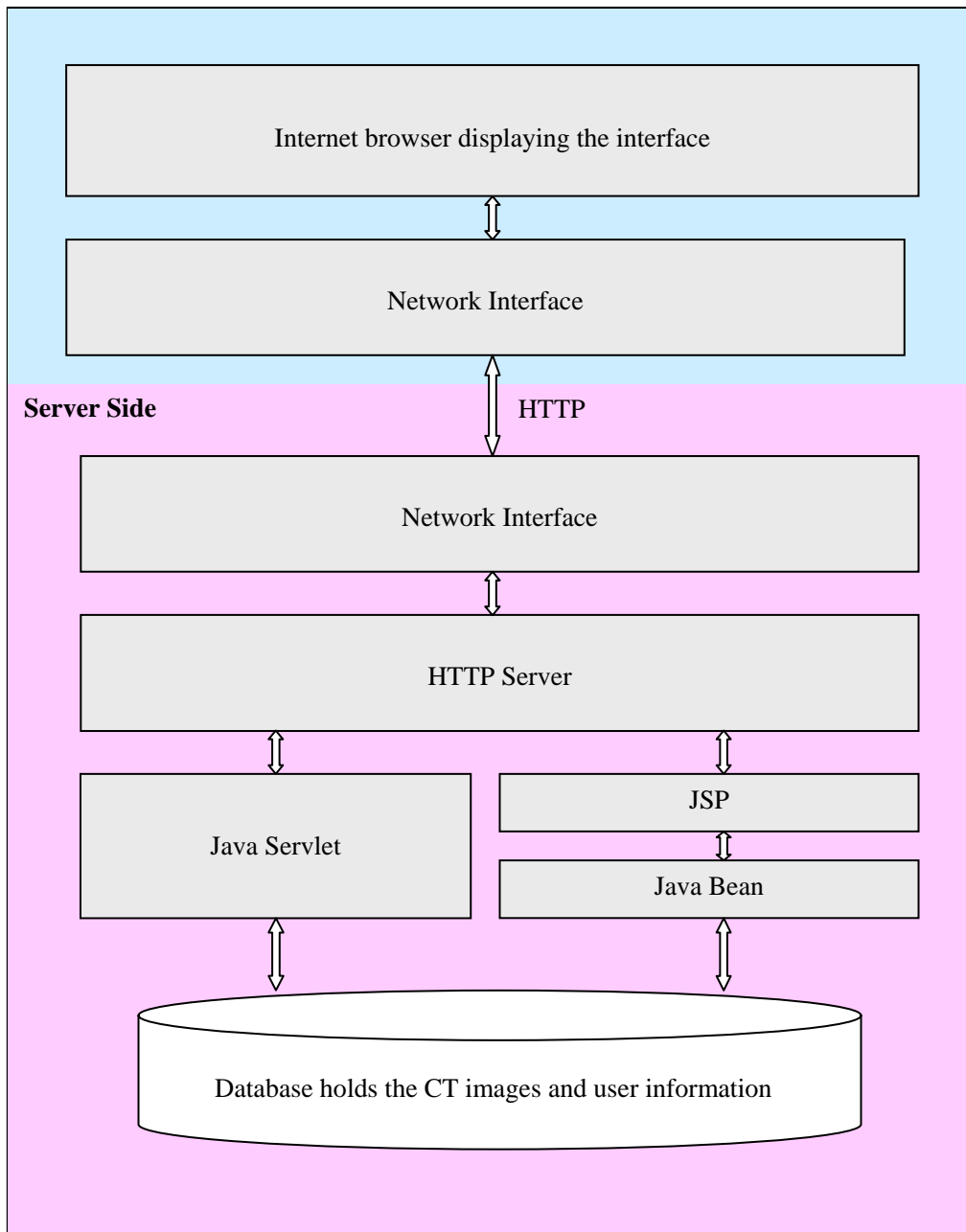


Figure 4-1. Block diagram of the architecture of the system

4.1 Training system made up of server side programs

4.1.1. Introduction

This initial implementation consists of developing a training system using Java Servlets. Technically, this means that everything is implemented on the server side of the system. On the client side, the user-interface of the system is made up of HTML

forms. Therefore, the end-user can interact with the server using standard HTML pages. In this approach, colorectal cancer screening is limited to the use of axial images.

The first part of this chapter introduces the operation of the system. Then, the next part describes the implementation of the system and focuses on data exchange between the server and the client, the design and functionalities of the user-interface. Also, this part explains how the server generates and sends axial images to the user's browser. Finally, the evaluation of the system is discussed and the observations made during this process are used to formulate further research directions and introduces the next approach of the system.

4.1.2. Design of the system with server side programs

4.1.2.1. Operation of the system

As illustrated in the flowchart figure 4-2, this system can be used by either specialist or trainee radiologists. Polyps flagged by specialists can be saved as the gold standard on the server. Following this, trainee radiologists can try flagging polyp using the same interface and then run an automatic evaluation.

The operation of this system is outlined in the following steps:

1. The first step is to enter the URL of the server in the browser's address bar.
2. The user then chooses a CT dataset and a user profile (trainee or experienced radiologist).
3. 2-D axial images of the chosen datasets are then sent to the client's computer and the user can flag potential polyps.
4. At this stage, the system provides 2 possibilities according to the user's profile:
 - a. The trainee can run an automatic evaluation of his/her study.
 - b. The experienced radiologist can save his/her work as gold standard.
5. Finally, the user exits the system or studies another dataset.

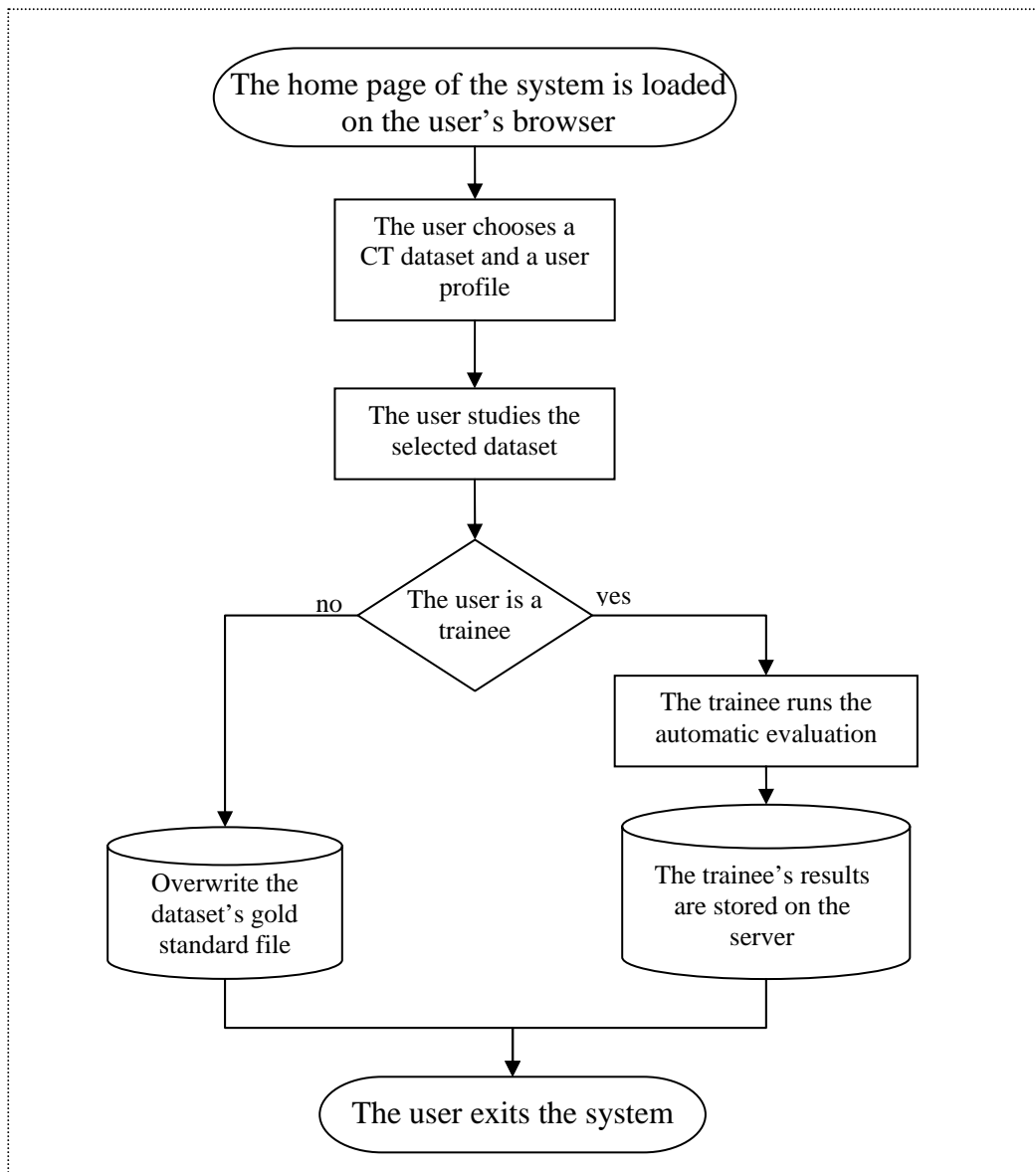


Figure 4-2. Operation of the system.

4.1.3. Implementation of the system with server side programs

4.1.3.1. Data exchange between server and client

This system is based on client-server architecture and is implemented using Java Servlets and combinations of JSPs and Java Beans. JSPs and Java Servlets are used to generate HTML forms that are sent to the user's browser via a network. These forms use various HTML objects such as combo-box, text area, radio button, submit button and allow the user to interact with the server. The forms are of type "post" so the user's browser always returns post type data to the server.

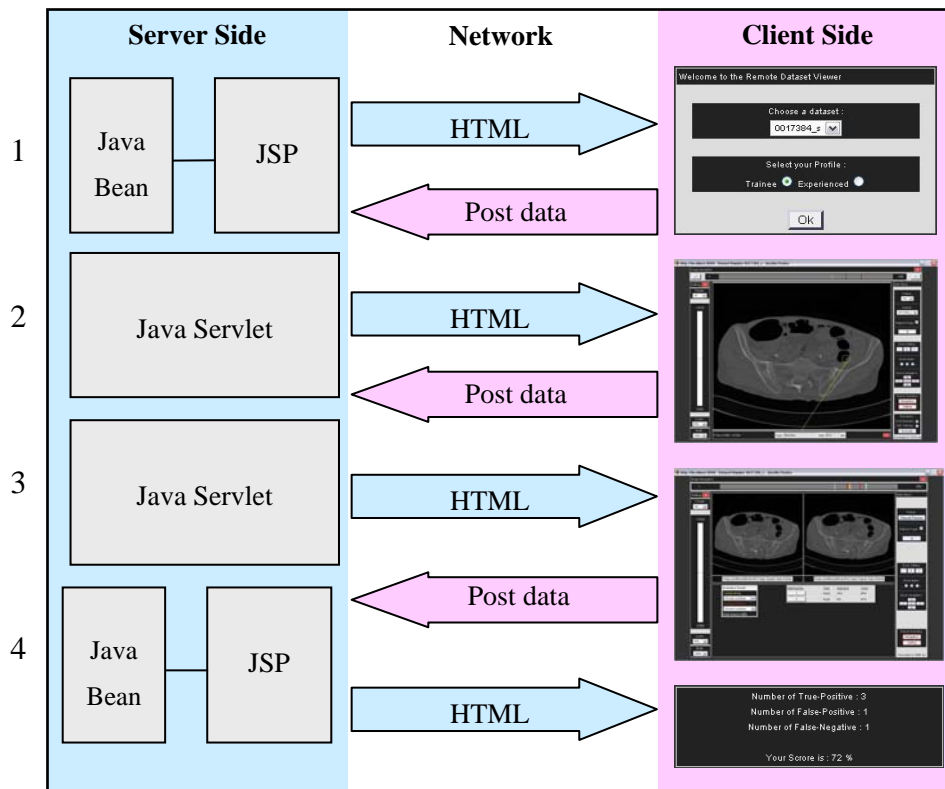


Figure 4-3. Overview of data exchange between server and client. The server sends HTML pages to the client. The client sends “post data” information to the server via HTML forms.

Figure 4-3 illustrates the data exchange between the server and the client when a trainee uses the system. For each step illustrated in this figure, an explanation is given below:

1. Home page
 - a. A JSP sends a HTML form to the user’s browser.
 - b. After choosing a dataset and a profile, the user presses a submit button on the form and his/her browser sends “post data” information back to the server.

2. Flagging potential polyps
 - a. A Java Servlet sends a HTML page displaying a JPEG image of a selected 2-D axial image.
 - b. The user flags potential polyps using mouse clicks whose coordinates are sent to the server as “post data”. Eventually, the trainee clicks on a submit button to run the automatic evaluation.

3. The evaluation panel
 - a. The Servlet sends an evaluation panel to the trainee's browser. This panel is a HTML form and displays side by side the polyps found by the trainee and the correct polyps from the gold standard.
 - b. The user interacts with the evaluation panel via the HTML form and the browser returns "post data" information.
4. Finally, the Servlet sends a HTML page displaying the results of the trainee.

4.1.3.2. Layout using CSS

As the entire graphic interface is written in HTML, all style and layout information are gathered into a Cascading Style Sheets (CSS) file. The flexibility of CSS allows the HTML interface of the system to look more consistent. As a result, each of the toolbars in the interface have the same appearance and the overall view of the system looks homogeneous. In practical terms, a css file called "globalCSS.css" has been created on the system. This file contains the layout information related to the HTML objects that are used in the interface. For instance, the css file specifies the body that will be used on every HTML page of the system:

```
BODY { font-family: Arial, Sans-serif;
        color: #111111;
        font-size: 12px;
        background-color : #333333;
        margin: 0px;
    }
```


⚠ Note that when a family name is set to "Arial", the World Wide Web Consortium (W3C) recommends adding the generic family at the end e.g. "Sans-serif".

In order to use the styles defined in the file "globalCSS.css", each HTML page must contain the code below in their header:

```
<head>
    <link rel="stylesheet" type="text/css" href="globalCSS.css"/>
</head>
```

It is possible to define a specific style in “globalCSS.css” and to use it for a selected HTML object. For instance, the style “InputButton” is defined in “globalCSS.css” as followed:

```
.InputButton {color: #EEEEEE;  
background-color: #660000;  
font-size: 10px;  
font-family: Arial, Sans-serif;  
text-align: center;  
cursor: pointer;  
cursor: hand;  
}
```

⚠ Note that there are two parameters “*cursor*” with different values. These parameters are used to turn the mouse arrow into a pointer e.g. . *Cursor: pointer* is the official way (as defined in the W3C recommendation) to change the cursor to a pointer whereas *cursor: hand* is used for the versions of Microsoft Internet Explorer prior to IE6.

The style “InputButton” can be used inside a HTML object such as a submit button by using the parameter “class” as illustrated below:

```
<input type="submit" class="InputButton" value="Enter!" />
```

4.1.3.3. Home page

The home page of this implementation (see figure 4-4) is implemented using a JSP and a Java Bean.

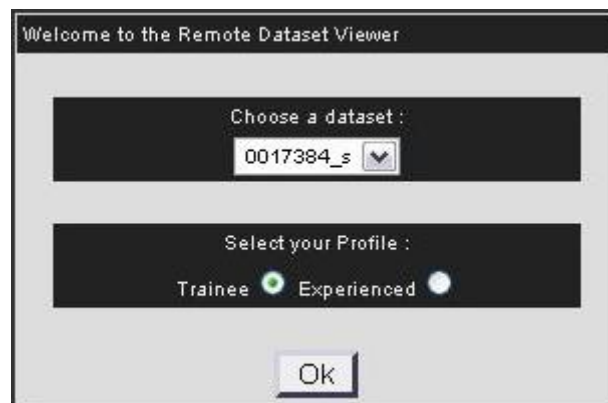


Figure 4-4. The home page of the system. On this HTML page, the user can select a dataset and a profile before starting the system.

In this interface the user can choose a CT dataset he or she wants to train with. The Java Bean gets the names of the datasets stored in the server's hard disk. These names are displayed using a HTML combo-box on the JSP. Also, two radio buttons have been implemented on the home page interface in order to allow the user choose a specific profile before running the system. It is possible to log in as a trainee or as an experienced radiologist. When the user clicks on the Ok submit button, the Servlet reads the first image of the chosen dataset and sends it to the client's browser.

4.1.3.4. Extract and display axial images

The Servlet reads the pixels of an image from a specific dataset using the class *RandomAccessFile*. In order to extract a specific axial image from the dataset, the Servlet skips the voxels of the previous image. For example, to display the axial image highlighted in figure 4-5, we first need to skip $2 \times n \times widthInVoxels \times heightInVoxels$ Bytes. After that, the axial image can be displayed by reading values of the next $2 \times widthInVoxels \times heightInVoxels$ Bytes.

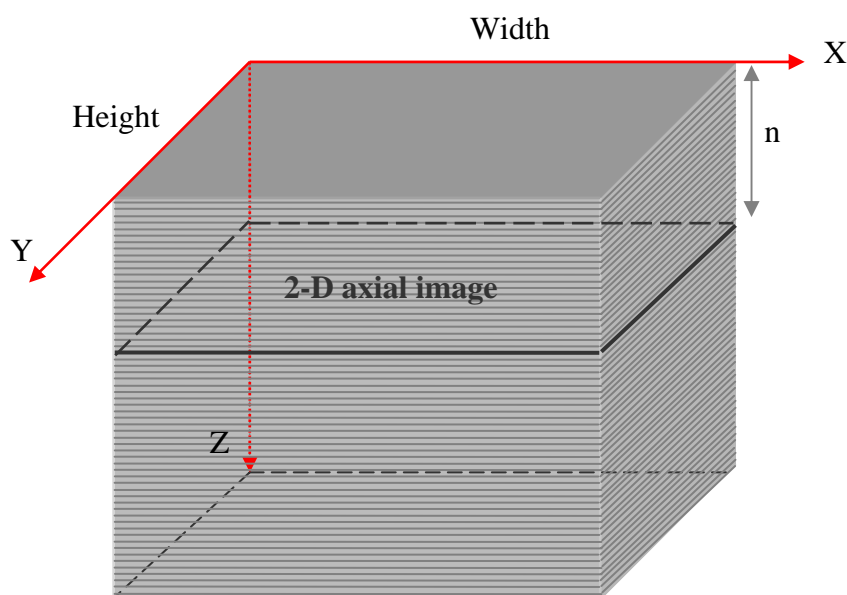
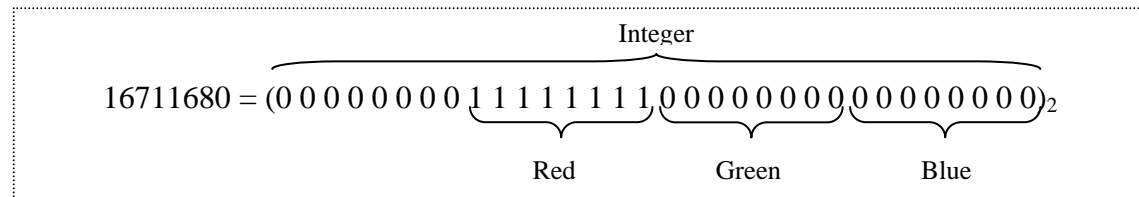


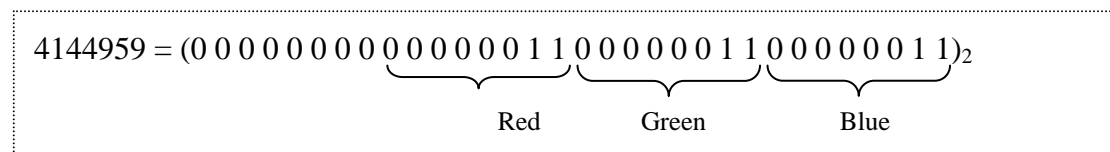
Figure 4-5. 2-D axial images in a CT dataset. A dataset can be represented as a cube made up of a stack of axial images.

To display a 2-D slice using a Java program, each pixel is drawn on a *BufferedImage*. The type of the *BufferedImage* must be a colour type such as “*TYPE_INT_RGB*” or “*TYPE_INT_ARGB*” in order to be able to add coloured polyp marks. A conversion must be performed to turn a voxel value from a 256 greyscale

into a RGB (red green blue) value. A RGB value is stored in an integer type variable. One integer = 4 Bytes = 4*8 bits. In this integer, 8 bits refer to the red component, 8 bits refer to the green component and 8bit refers to the blue component. The example below illustrates the binary representation of a red pixel value.



A grey value is obtained if the 8 bits referring to the red, green and blue component are the same. The example below illustrates a grey value.



In this example it can be seen that a grey value can be coded on 8 bits. The conversion from a grey value to an RGB value simply consists of duplicating the 8 bits of the grey value to the 3 colour components of an RGB value. The code below illustrates this conversion:

```
int int_hex = voxel << 16 | voxel << 8 | voxel;
```

A Joint Photographic Experts Group (JPEG) (Pennebaker et al. 1993) image is created from the *BufferedImage* previously generated. In order to perform this task, a method named *makeJPG()* has been created on the Servlet. To store a *BufferedImage* as a JPEG, the type of this *BufferedImage* must be *TYPE_INT_RGB* instead of *TYPE_INT_ARGB*. This can be explained by the fact that *TYPE_INT_ARGB* includes transparency whereas the JPEG format doesn't support it. Finally this JPEG image is displayed via the HTML code of the Servlet using a `` tag.

4.1.3.5. Navigation

The user can navigate through the axial images of a dataset using different features.

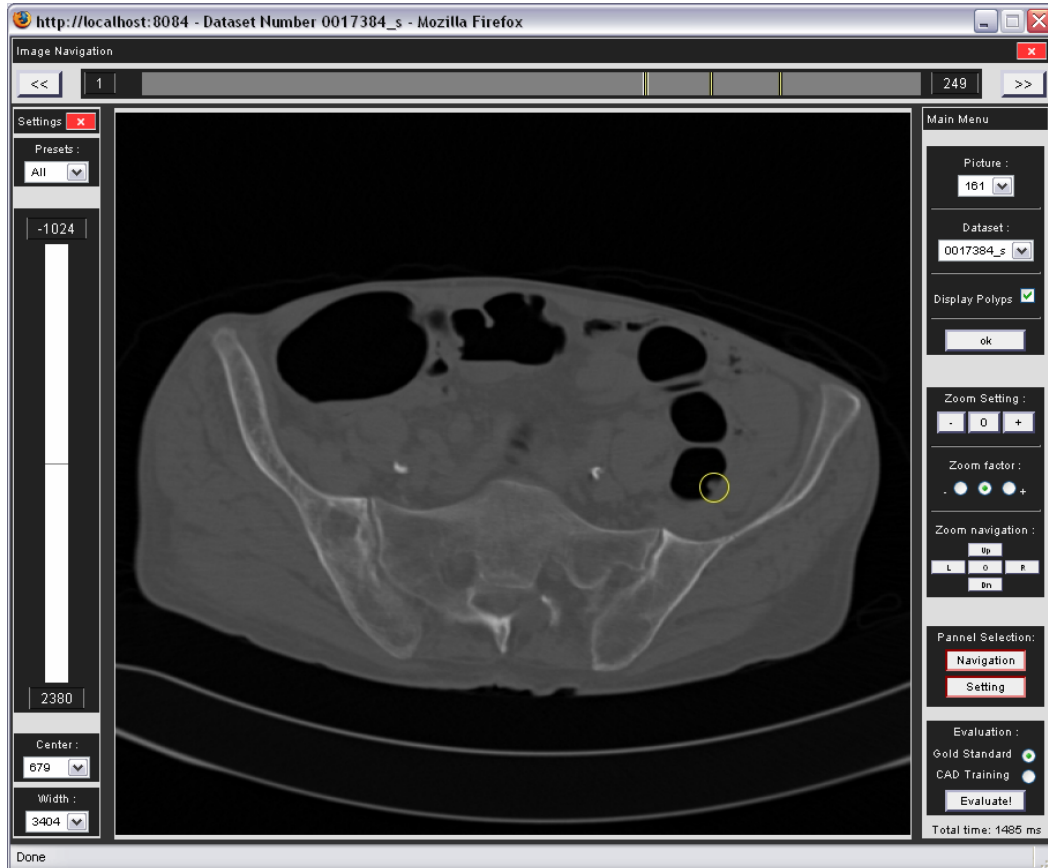


Figure 4-6. The trainee's interface with a potential polyp (circle). On this interface, the user can navigate through the axial images using the horizontal bar, the user can also use the vertical bar on the left to select a specific density range and use the different options on the right of the interface to zoom in or zoom out.

The user can use a HTML combo-box to select the axial image he/she wants to display. At the top of the Servlet (see figure 4-6), a navigation toolbar contains a forward and a backward HTML button.

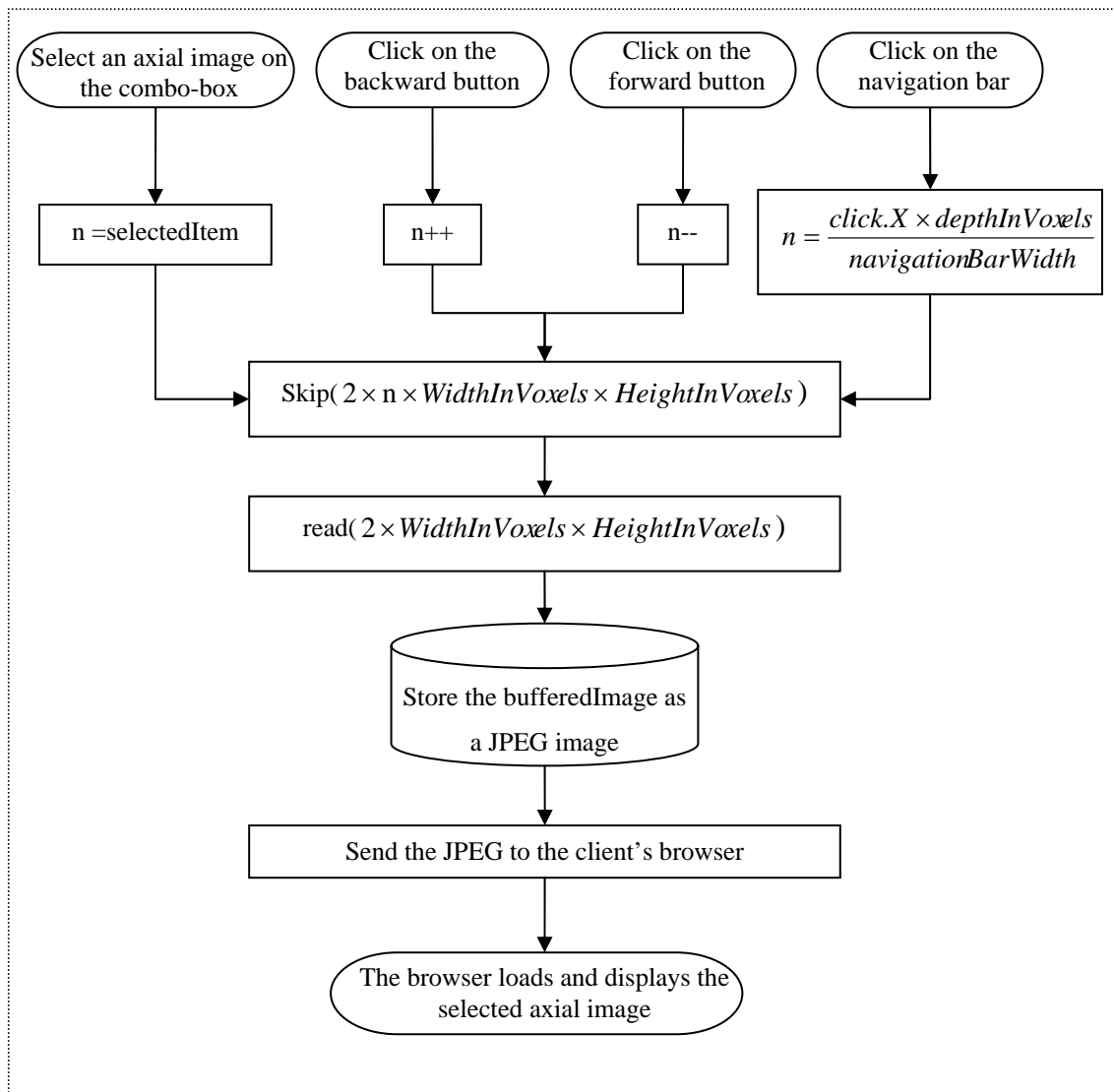


Figure 4-7. Select and display an axial image. This flowchart describes the operation of the different image navigation features.

These buttons increment or decrement the axial image number to display (see figure 4-7). Between these buttons, a “navigation bar” has been implemented. This bar is a JPEG image that can be clicked on in order to instantly access a specific image of the dataset. This is done using a HTML tag `<input type="image" name="pic">`. The x coordinate of the mouse click is retrieved using the code: `request.getParameter("pic.x")`. In the present case, we only care about the x coordinate because the navigation bar is horizontal. Finally, each time a new axial image is selected, the combo box displays the current image number.

If a trainee flags a polyp on an axial image, a little mark is drawn on the navigation bar. The user can then visualise which axial images already contain flagged polyps and can easily access it by clicking on the appropriate mark.

4.1.3.6. Flagging potential polyps

We use the HTML tag `<input type="image">` again to flag potential polyps on axial images. The trainee can simply click on the centre of a polyp. The (x,y) coordinates of the click will be retrieved and a circle will be displayed around this point. In practical terms we draw the 2-D axial image to a new **BufferedImage** and we draw a red circle around the (x,y) coordinate of the mouse click.

```
Graphics2D graph = (Graphics2D)NewBuff.getGraphics();
graph.setRenderingHint (RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON); //Set the anti-alias
```

▲ An antialias is used to avoid a stair-step look for the polyp circle.

When a trainee clicks on a potential polyp, a toolbar appears at the bottom of the page (see Figure 4-8). The trainee can use this toolbar to specify the type (sessile or pedunculated) and the size (in mm) of the highlighted polyp. Finally, the trainee can click on the button “Add polyp” to store the new polyp information. The X and Y coordinates of the mouse click, the polyp’s type and the polyp’s size are written to a text file on the server using the classes **FileOutputStream** and **OutputStreamWriter**:

```
FileOutputStream polypsFile = new
FileOutputStream(str_polypFile, true);
OutputStreamWriter wr = new OutputStreamWriter(polypsFile,
java.nio.charset.Charset.forName("UTF-8"));
wr.write(int_coordinate+"\n");
wr.close();
polypsFile.close();
```

▲ The second parameter of the constructor of the class **FileOutputStream** is set to true in order to append the new polyp information at the end of the text file if it already exists.

▲ The class **OutputStreamWriter** is used to convert the polyp information in UTF-8 format before writing it in the text file. Using this conversion, the text file can be opened on the server side with a standard text file reader. If we only used the class **FileOutputStream**, polyp information should have been converted into Bytes.

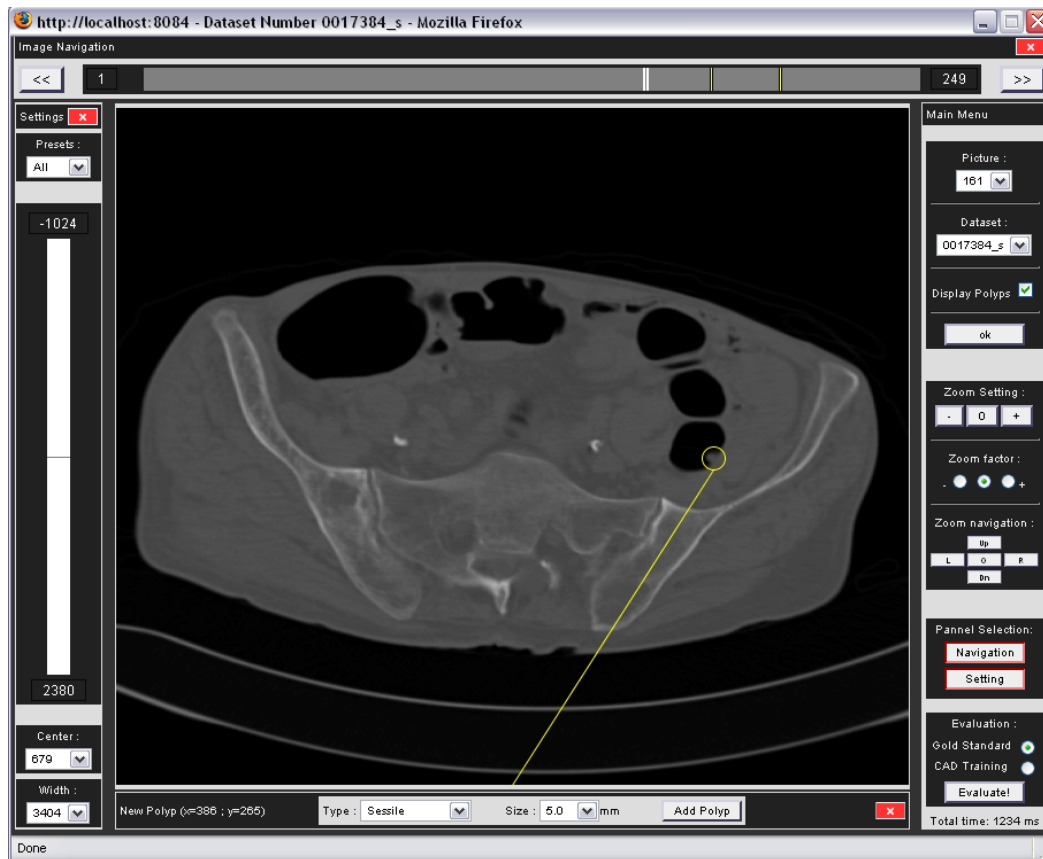


Figure 4-8. Flagging a potential polyp. If the user clicks on the axial image, a circle is drawn around the coordinates of the mouse click and a horizontal toolbar appears at the bottom of the interface. On this tool bar, the user can select the type and the size of the new polyp.

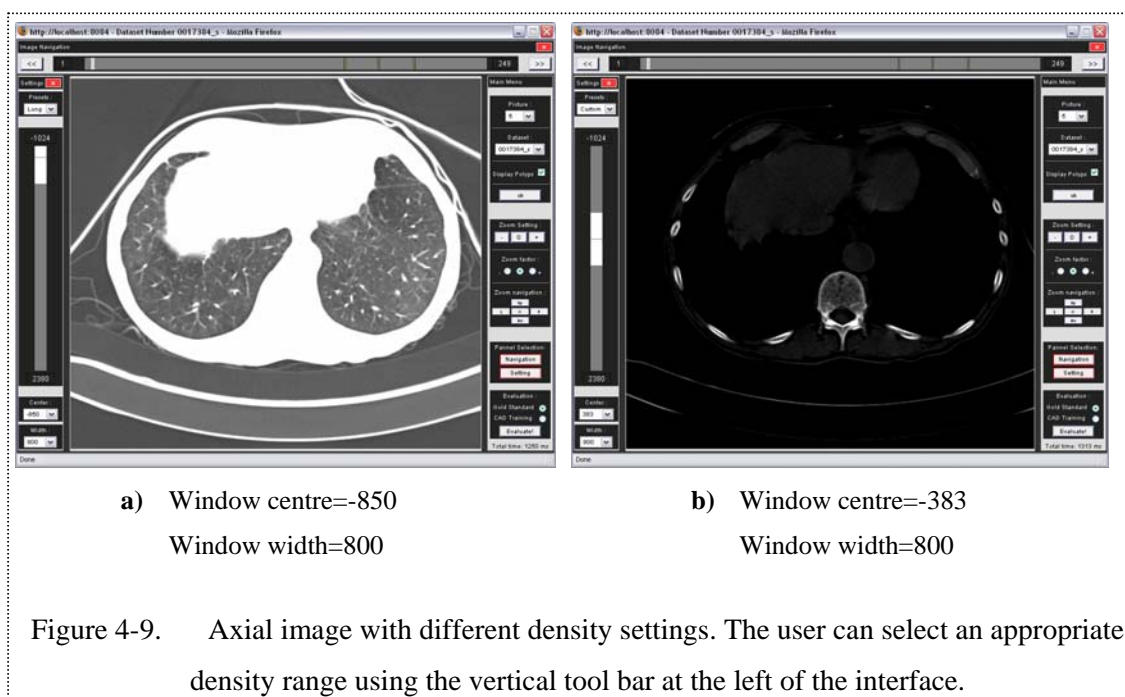
4.1.3.7. Polyp's marks

If a trainee chooses to display a 2-D axial image that contains polyp previously flagged, a circle is drawn around the polyps' centre. When the user clicks inside a polyp's oval, the mouse click's coordinates are retrieved and compared to the polyps' positions from the polyp's text file. Once the selected polyp has been found its type and size are displayed using a HTML table at the bottom of the page.

4.1.3.8. Density setting

On the left of the interface (see figure 4-9), another toolbar gives the end user the possibility to focus on a specific density. It works on two image characteristics which are commonly used by radiologists; these are the window width and the window centre. The window centre specifies the density value that we want to focus on. It is then possible to use these parameters in order to highlight information associated with

e.g. the lungs (see figure 4-9.a) or the bones (see figure 4-9.b). The window width is the amount of other density values that we want to display around our chosen window centre.



Our end-users can adjust the window width and the window centre using different features. At the bottom left of the interface, two combo boxes display the current window width and window centre. If the user chooses a different value on one of these combo boxes, the Servlet will generate the new axial image and the browser will automatically load the new image. Also, a vertical bar has been implemented on the left of the interface. The height of this bar represents the density range of the dataset voxels (about -1024 to 2400HU). Inside this grey vertical bar, a white bar indicates the current windowing setting. The height of the white bar indicates the window width of the current axial image. This feature works as a scroll bar. If the user clicks on the vertical bar, the y coordinate of the click will be retrieved and the new window centre will be calculated:

Initially, the window width is equal to the density range of the dataset. Therefore, the white bar fills the whole vertical bar and the window centre is in the middle of the bar (see figure 4-8). The new axial image will be generated and the vertical bar will be updated by placing the centre of the white bar at the y coordinate of the mouse click. The two combo boxes will display the new window width and window centre values.

4.1.3.9. Zoom setting

A zoom feature is available on the main menu of the Servlet. Three radio buttons let the user choose an appropriate zoom scale. The user can click on + or – buttons in order to zoom in or zoom out. These buttons are HTML “submit buttons” which simply increase or decrease the zoom factor in accordance to the chosen zoom scale (see figure 4-10).

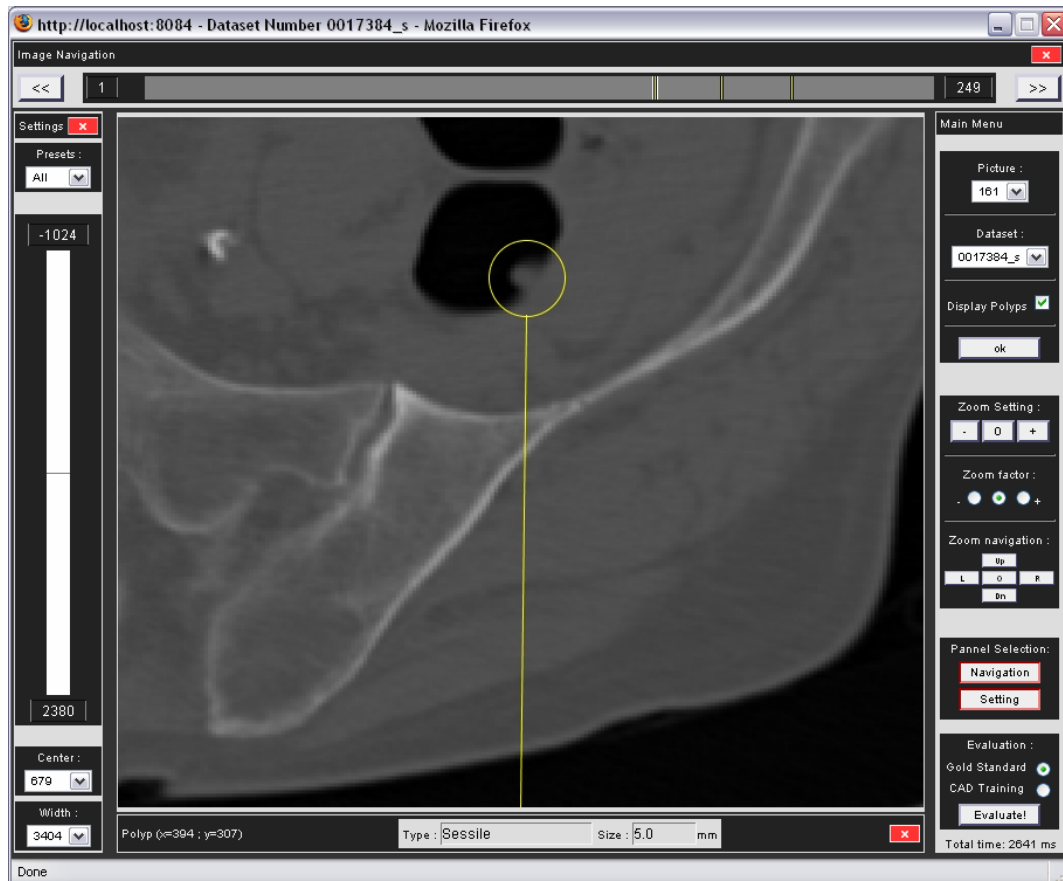


Figure 4-10. Zoom on a potential polyp. On the right of the interface, the user can select an appropriate zoom area and zoom factor.

A third button (“O” for origin) has been implemented between the zoom in and zoom out buttons. This button can be used to go back to the initial size of the axial image. The image width and height are multiplied by this zoom factor and results are passed as parameters to a *getScaledInstance()* method.

In order to navigate into the zoomed picture, there are 4 submit buttons named Left, Right, Up and Down. As we draw the zoomed Image to the *BufferedImage*, x and y values are given as parameters to the *drawImage()* method in order to apply the required translation. Between the 4 buttons (left, right, up, down) a fifth button named “O” stands here to go back to the initial position.

4.1.3.10. Result panel

The results of the evaluation are displayed via a dedicated interface (Figure 4-11).



Figure 4-11. The result panel displaying a true-positive polyp. In this example, a polyp flagged by the trainee and the corresponding polyp from the gold standard. When the trainee runs the automatic evaluation, a colour code is used on the navigation bar at the top of the interface to indicate the true-positive (white), false-positive (yellow) and the false-negative (red) polyps. Also on this page, the user can use drop down menus to display the false-positive and false-negative polyps.

The user can instantly see the results by looking at the navigation bar at the top of the screen. On this bar, white marks indicate correctly flagged polyps (true positives), yellow marks indicate incorrectly flagged polyps (false positives) and red marks indicate unflagged polyps (false negative). Images containing polyps flagged by the trainee that match with the gold standard are displayed side by side under the navigation bar. The user can then click on one of these pictures to switch it to full screen mode. Two HTML combo boxes are located under these images. On the first one, the trainee can choose to display one of the axial images containing false positive polyps. On the second list, the user can choose to display one of the axial images containing false negative polyps.

For each true positive, a score is calculated according to the type and size accuracy. A HTML button is associated with each true positive. When pressed, this button displays side by side, images matching the trainee’s selection with the gold standard. Ultimately, a final score is calculated for the type and size accuracy for each true positive polyp and the result is divided by the number of false negative polyps to yield the final result.

4.1.4. Conclusion on the implementation using Java Servlets

Upon completion of this first version of the system, a demonstration was organised in order to obtain suggestions from end users. Two radiologists evaluated the system from the Mater Misericordiae Hospital in Dublin. These specialists were satisfied with the interface and the different features of the system but they also found some issues that will have to be taken in consideration for the next version of the system.

Tasks	Time for server with CPU: P4 1.6GHz RAM: 512MB	Time for server with CPU: P4 2.8GHz RAM: 512MB
1. Creation of the BufferedImage	250ms	100ms
2. Reading polyp’s info from text file	<1ms	<1ms
3. Adding polyp’s marks and zooming	1875ms	750ms
4. Creation of the JPG picture	75ms	40ms
5. Transferring and display interface	700ms	700ms
Total time to display a new Image	2900ms	1590ms

Table 4-1. Overview of the time required for the different tasks.

1. The main issue clearly appeared to be the latency of the system. In other words, the Servlet takes too much time to perform the requested tasks and transfer pictures through the Internet. An overview of the length of time related to each step of the program is displayed in table 4-1. The times on rows 1 to 4 were locally calculated on the Servlet and the time on row 5 was obtained while requesting a new image from a remote computer.
2. It would be useful to be able to use the mouse wheel to navigate through the images of a dataset. This feature will also have to be added to the next version.
3. A few requests were made about the “evaluation” mode. First, each trainee should have the possibility to login with a specific name and password in order to train with several datasets and then send the results of their attempts to be evaluated.
4. Finally, the results should be formatted so that all the images containing false positives should be visible together on the same screen and all the images containing false negatives should be visible together on another screen.

4.2 Training system made up of Servlets and Applets

4.2.1. Introduction

Following the evaluation of the initial implementation, it appears that the best way to meet the needs of our end users is to initially transfer the CT dataset from the server to the client side. Once the dataset is stored on the client's hard disk, the interface simply needs to read the dataset and extract the relevant images. A Java Applet includes libraries and methods that can be used to extract and display pictures of a CT dataset stored on the client's hard disk. Compared to HTML pages, an Applet provides more advanced capabilities to create a user-friendly interface.

The following text gives an overview of data exchange between server and client and explains the operation of the system. Then, the next part describes the implementation focusing on the compression used on CT datasets, the Applet certification in order to interact with the client's hard disk, the servers used to exchange data with the client and the different functions provided by the user interface.

4.2.2. Design of the system with Applets and server side programs

This approach uses a client-server architecture implemented over the Internet using a combination of Java Servlets, JSPs and Java Applets. As a Java Applet is a client side Java program that can be loaded on any standard browser, this approach provides a high level of compatibility.

Communication between the server and the user's computer is crucial to the operation of the system and consists of 6 steps (see figure 4-12):

1. Each user has an account in order to allow monitoring of their training. Therefore, the first step consists of logging into the system with a password. For this purpose, when connecting to the server, the user's browser loads a HTML login form that is generated by a JSP.
2. The server then receives the login and password information and compares it to its user database. If it matches, access to the system is granted and another HTML form is sent to the user's browser. This new form gives to the user the option of choosing between starting a new training session and displaying their previous results.

3. If the user chooses to undergo training, the user's browser then loads a Java Archive (JAR) and starts a Java Applet. This JAR is sent by the server and contains the classes of the Java Applet and x-ray scout of CT datasets that are available for training.
4. The trainee can then choose a CT dataset he/she wants to train with. The name of the selected dataset is sent to the server.
5. To achieve the fastest response time to the interface, a compressed CT dataset is sent from the server and is stored on the client's hard disk (using a signed Applet). 2-D slices of the dataset are then uncompressed as soon as they arrive on the trainee's computer. It gives the user the ability to start working on the initial images while the rest of the dataset is being transferred.
6. Upon completion of their work, the trainee can run an automatic evaluation based on gold standard information previously gathered from colonoscopy records. The Applet displays the results as true-positive, false-negative and false-positive. Finally, the trainee can view a panel showing his/her new and previous results.

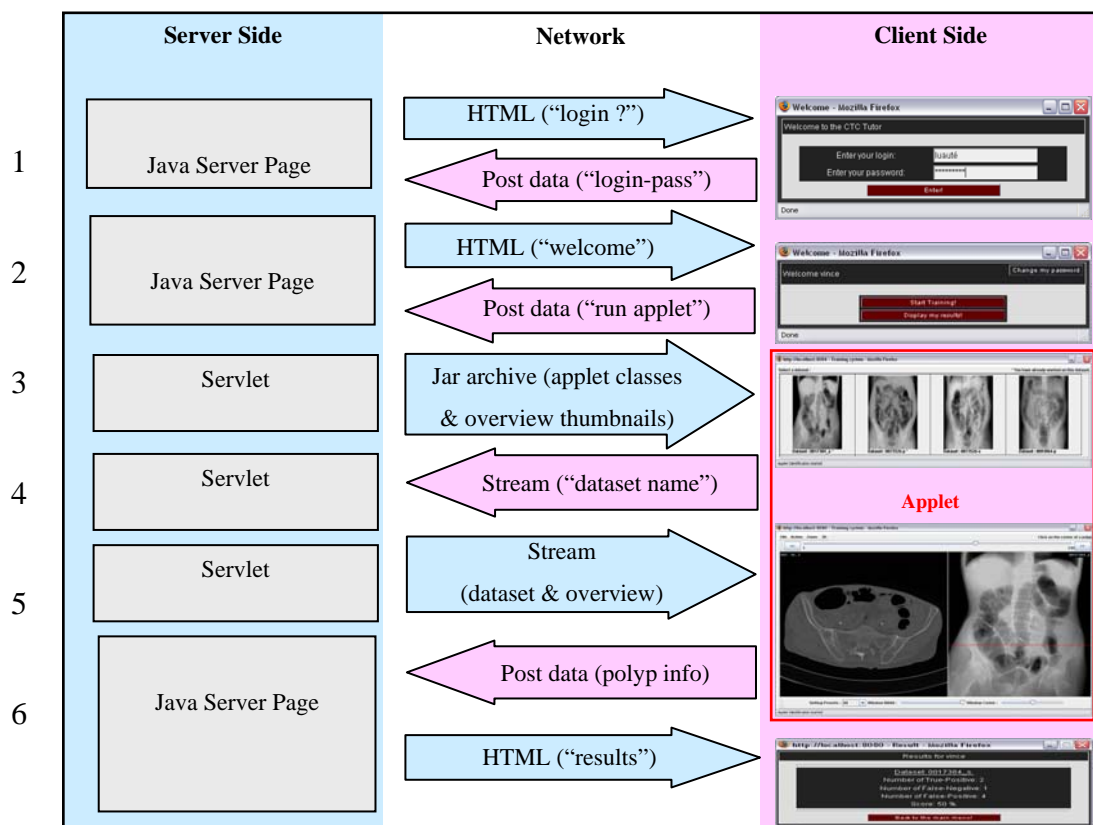


Figure 4-12. Overview of data exchange. The server sends HTML pages to the client and the client returns post data information. When the Applet is started, the Applet communicates with the server using data streaming.

The following flowchart, figure 4-13 illustrates the operation of the system for the user registered as a trainee and for the user registered as a specialist.

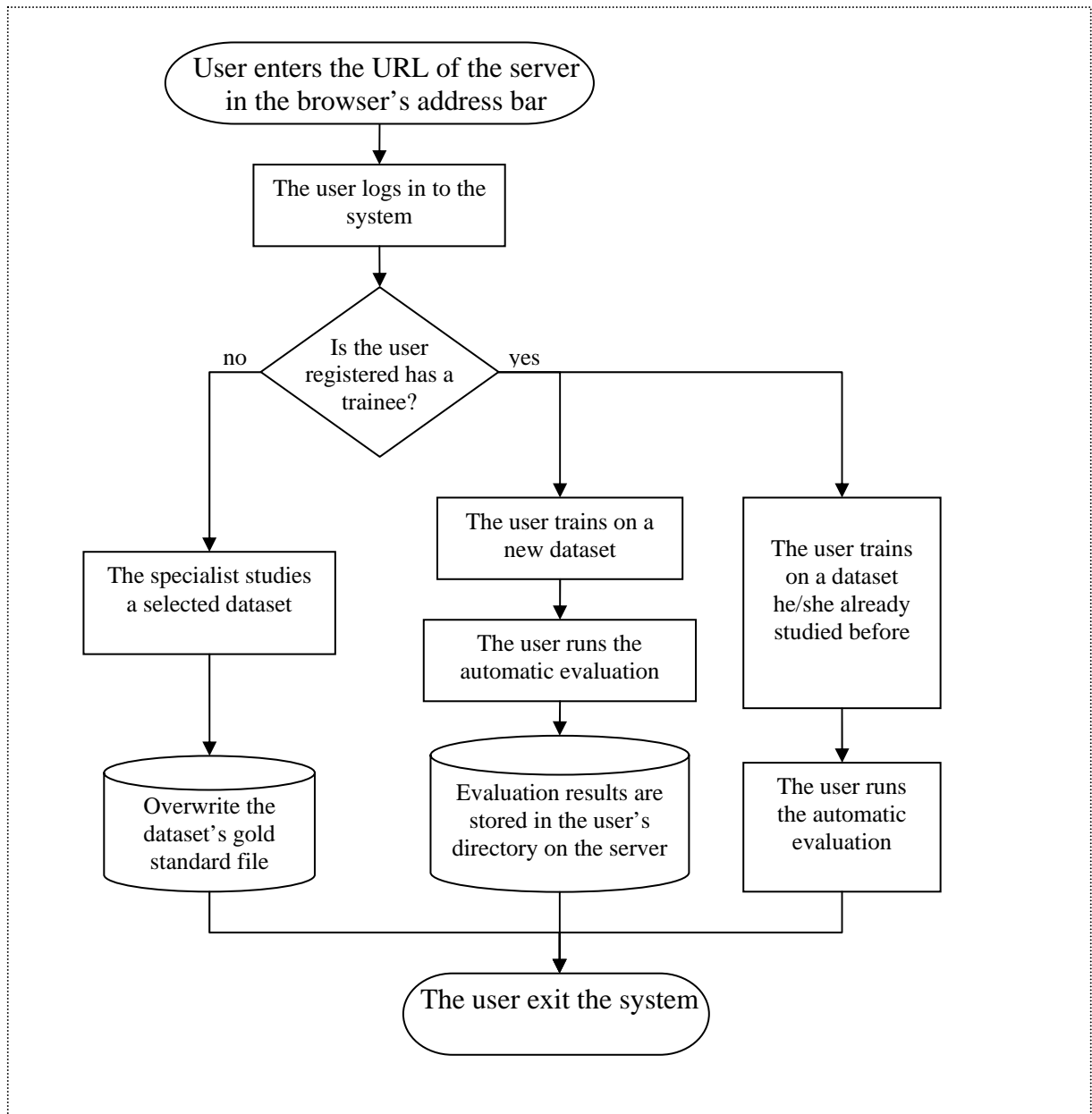


Figure 4-13. Operation of the system. This flowchart highlights the operation difference between a trainee and a specialist using the system.

4.2.3. Implementation using Applets and server side programs

4.2.3.1. Dataset compression

The CTC datasets are large (~150MB) and the transfer of this information from server to client would usually be a time consuming operation. To overcome this problem, a

loss-less compression algorithm has been developed to accelerate dataset delivery over a network. The approach of the compression that is summarised below was developed by colleague O'Halloran (2005).

This compression technique has two goals:

1. Of course the first goal is to achieve a good compression ratio.
2. The decoding process of this compression technique will be implemented on the client-side (Applet). Therefore the second goal is to reduce excessive computation and make the decoding process as fast as possible.

Compression of a CT dataset can be achieved by using a combination of different techniques. A first step is to use the motion estimation and compensation method (Zafar et al. 1991). This technique is based on the observation that there is a high correlation between consecutive slices in a CT dataset. This can be seen by viewing a difference image of two consecutive slices in figure 4-14.

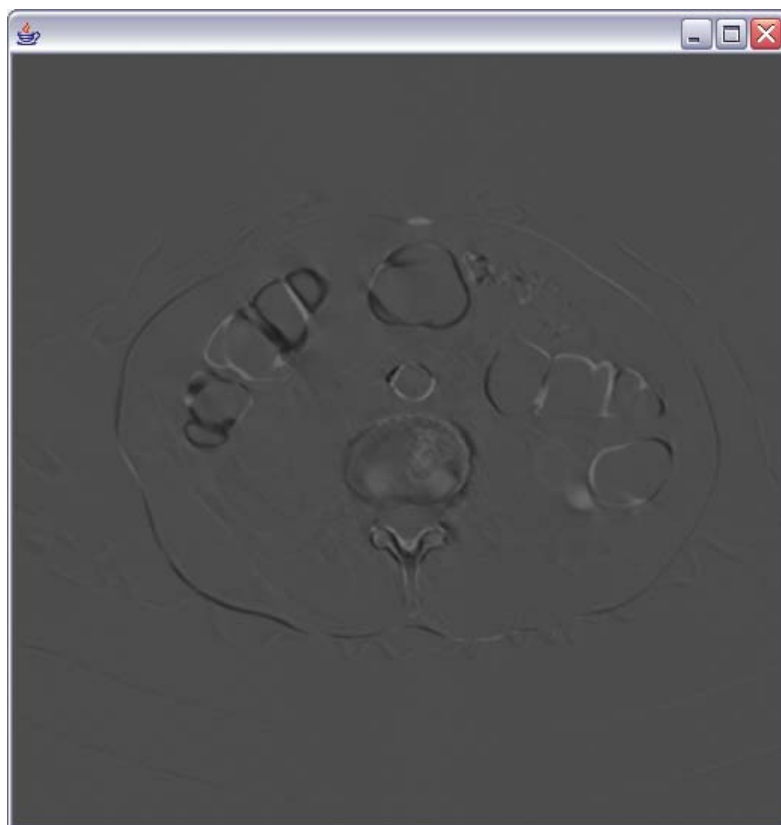


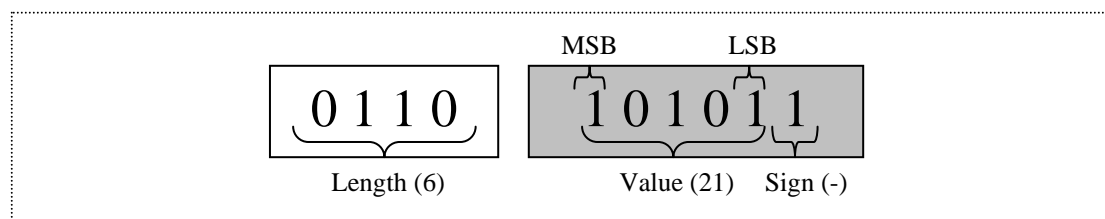
Figure 4-14. Difference between two consecutive axial images. This image highlights the fact that sending a difference image represents less data transfer than sending an entire axial image.

The motion estimation and compensation technique generates frames containing most values around 0. Therefore, the next step of our compression technique must be very efficient to encode values in this area.

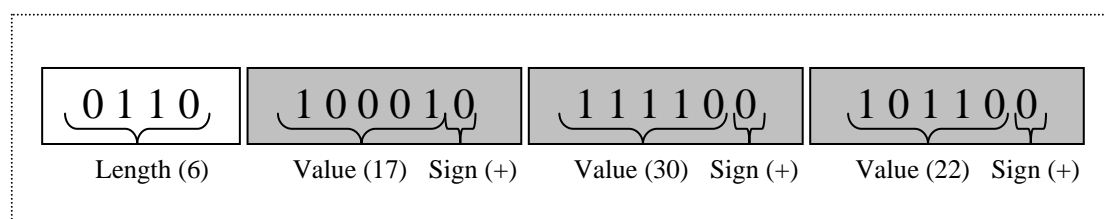
As a second step, an idea is to avoid the waste of data due to the fixed type of our dataset's voxel values. The type of a voxel value is "short" (1 short = 2 Bytes = 16 bits). A "short" can contain a value in the range of (-32768 to 32767). In our case, the density values that we want to encode can have a value in the range of (-1024 to around 2400 HU). In other words, even the minimum and the maximum voxel's values can be coded using only 12 bits + 1 sign bit instead of the 16 bits used by a "short": $-1024 = (11100000000000)_2$; $2400 = (0100101100000)_2$.

The second compression technique consists of writing the length, in bits, that the values occupy followed by the binary value. Also, the sign of the value is given by an additional bit located at the end of the binary value.

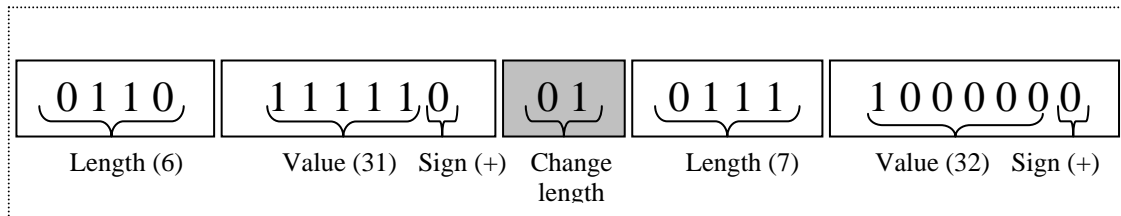
Beyond this point, this technique will be referred to as "limited length encoding". The example below illustrates this approach for the encoding of the value -21:



In order to make this approach more efficient, the 4 bits dedicated to the length are written only if the length varies from a value to the next one. For instance, if 3 values with the same length are encoded successively, the 4 bits length appears only once before the 3 values as illustrated below:

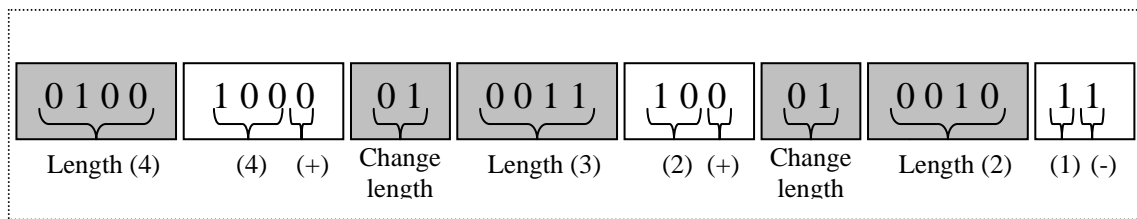


If the next value has a different bit length, we first insert a "change length" code made of 2 bits: "01" and we write the new 4 bits length. For instance, if two values 31 and 32 are sent successively, this "change length" code will be inserted as illustrated below:



All binary representations are written using the minimum bits possible. For instance, the binary representation of the decimal value 31 is $(11111)_2$ and not $(011111)_2$. Therefore, the binary representation of a voxel's value will always start with a "1". This explains how the decoder will be able to determine the difference between a "change length" (01) and a voxel's value starting with a "1". In our CT dataset's, most of the neighbouring pixels have similar values. Therefore, the 4 bits length code is not written too often.

The example below illustrates how to successively encode the values 4, 2 and -1.



As it can be seen, at values near 0 there is an excessive waste of bits due to the necessity of writing a lot of "change length" code and 4 bits length.

At this point, a solution is to use a hybrid compression scheme using the "limited length encoding" approach previously described and another technique to encode values around 0 with efficiency. The Huffman coding (Huffman 1952) is a very efficient technique for the encoding of a limited amount of different values. Using Huffman coding requires creating a list of symbols "alphabet" where each data represents a "change code" a voxel value. A vast alphabet would involve an excessive computation for the decoding process and this would be in contradiction with the second goal of our compression approach.

A brief experiment has been conducted in order to find the exact region where voxel values should be encoded using Huffman coding in order to reach the optimum efficiency. The best result was obtained when using Huffman coding to encode values between -15 and +15 HU.

To indicate that we are changing the encoding technique from Huffman coding to the initial “length limited encoding” technique or vice-versa, the bits “00” is inserted. Also, the 4 bits length code is not inserted if we are switching from Huffman encoding to “length limited encoding” and if the voxel value has a binary representation equal to 6 bits (including the sign bit). In other words, when switching back to “length limited encoding” the 4 bits length will not precede values from 16 $(10000)_2$ to 31 $(11111)_2$.

When encoding using Huffman coding, a “sign code” is used to make the difference between the positive and the negative voxel values. This “sign code” must be added to the Huffman alphabet. When switching from the Huffman coding to the “limited length encoding”, we need to indicate whether the next value will require 6 bits or more to be encoded. For that reason, two more symbols must be added to the Huffman alphabet.

The symbol “up 1” indicates that the next value will require 6 bits (including the sign bit) and the symbol “up many” indicates that the next value will be encoded using more than 6 bits. The table 4-2 summarises the symbols used for Huffman encoding.

Symbol	Label
Sign code	0
Up 1	1
Up many	2
(0)	3
(1)	4
(2)	5
(3)	6
(4)	7
(5)	8
(6)	9
(7)	10
(8)	11
(9)	12
(10)	13
(11)	14
(12)	15
(13)	16
(14)	17
(15)	18

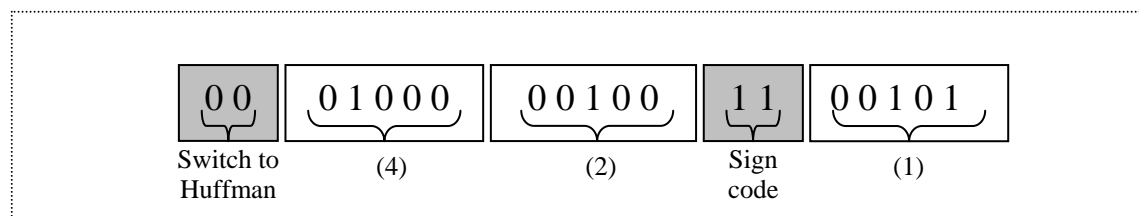
Table 4-2. Symbols encoded using Huffman codes.

These symbols are sorted according to their frequencies, (number of occurrence). The Huffman coding principle is to assign fewer bits to the values having the most occurrences. See table 4-3, an example of this process.

Sign code				1	1
Up 1			1	0	1
Up many			0	0	0
(0)	0	0	1	1	0
(1)	0	0	1	0	1
(2)	0	0	1	0	0
(3)	0	0	1	1	1
(4)	0	1	0	0	0
(5)	0	1	0	0	1
(6)	0	1	0	1	0
(7)	0	1	0	1	1
(8)	0	1	1	0	0
(9)	0	1	1	0	1
(10)	0	1	1	1	0
(11)	0	1	1	1	1
(12)	1	0	0	0	0
(13)	1	0	0	0	1
(14)	1	0	0	1	0
(15)	1	0	0	1	1

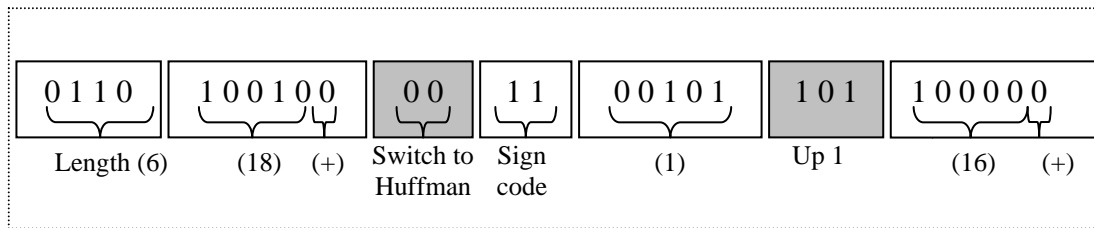
Table 4-3. Huffman codes array

In the previous example, the encoding of values 4, 2 and -1 using the “limited length encoding” took 25 bits. The new example below illustrates the benefits of using Huffman coding to encode the same values. As it can be seen in table 3, the Huffman code associated to the voxel value 4 is “01000” and the Huffman code associated to the voxel value 1 is “00101”.



Using Huffman coding, the encoding of values 4, 2 and -1 takes 19 bits. In other words, in this example we saved 6 bits.

As it can be seen in table 3, on this example, the Huffman code associated with the voxel “u1” is “101”. The example below illustrates how to successively encode the values 18, -1 and 16 using the final compression approach:



This compression technique ensures a 50% compression rate and divides transfer time by 2. Datasets are compressed on the server side prior to transmission. Also, the Applet interface is able to decompress and display 2-D images as soon as they are received on the user’s hard disk. The compression of a selected CT dataset takes approximately 6 minutes using a PC with a 2.8 GHz Intel Pentium 4 processor and 512 MB of RAM.

4.2.3.2. Creation of a simulated scout x-ray

A simulated scout x-ray of the CT dataset is displayed on a dedicated floating window in the user-interface. To obtain this image, we give to each of the (x, z) values, the average of the values present on the corresponding y axis as illustrated in figure 4-15.

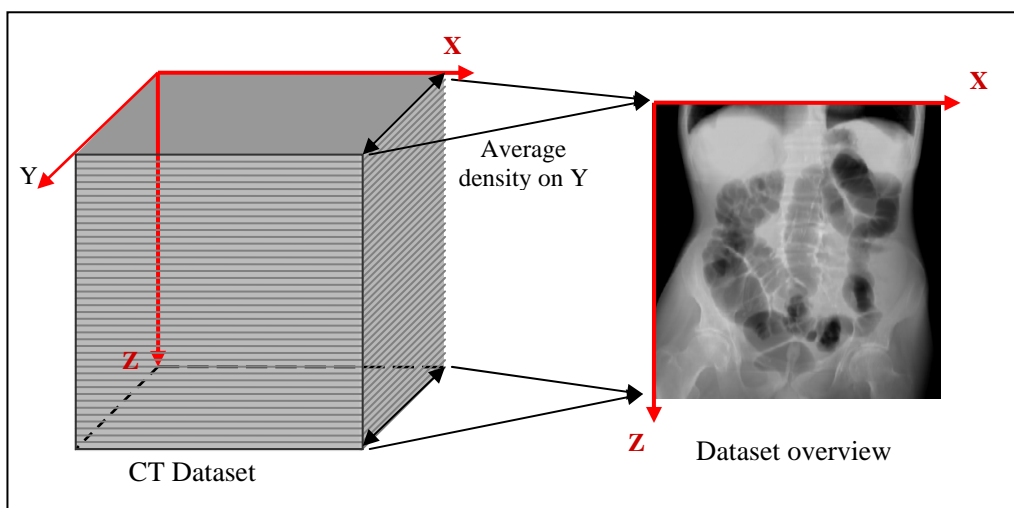


Figure 4-15. Generate a simulated scout x-ray. The resulting image is obtained by calculating the average density on Y.

After that, this picture is stored using the Joint Photographic Experts Group (JPEG) format using the method `makeJPG()`. This image is generated on the server prior to transmission and is included in the Jar archive that also contains the Applet classes.

A Java application has been developed to help the administrator of the system to generate simulated scout x-ray images. The administrator can choose to generate the x-ray image of a specific dataset or generate x-ray images of every dataset stored inside a folder. Moreover, the compression algorithm has been implemented in this application. Therefore, the administrator of the system can generate x-ray images and compress new datasets using the same application.



Figure 4-16. Administration tool to generate a simulated scout x-ray. The administrator can use 4 sliders in order to highlight the colon on the scout x-ray.

Four sliders have been implemented in the interface (see figure 4-16). The window width and the window centre are used to focus on the appropriate density values. The sliders “Y min” and “Y max” are used to focus on a specific section of the colon and to skip the rest of the dataset as illustrated in figure 4-17. The use of these four parameters can greatly improve the rendering of the colon.

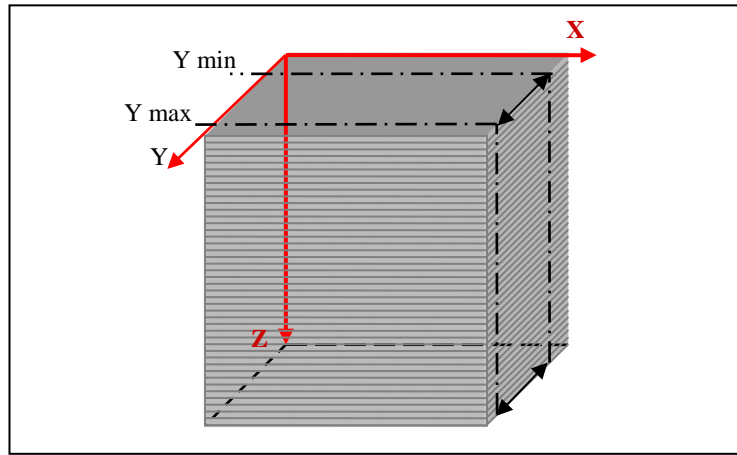


Figure 4-17. Focusing on the appropriate region on Y. It is possible to focus on the relevant region around the colon and prevent the remaining information to overload the scout x-ray.

The program used to generate the simulated scout x-ray starts by reading the header of the selected dataset in order to get the “widthInVoxels”, “heightInVoxels” and the “depthInVoxels”. After that, the whole dataset is stored into an array using the method *readFully()* of the class *DataInputStream*.

▲ Using *readFully* to store the whole dataset in random access memory (RAM) takes a few seconds at the beginning. However after this delay, moving the sliders of the interface is a real time operation.

The process used to generate a scout x-ray image is described in figure 4-18. In this flowchart, “limit down” and “limit up” are calculated from the window width and the window centre (see equation 4.2.1 and 4.2.2).

$$\text{limit up} = \text{window centre} + \frac{\text{window width}}{2} \quad (4.2.1)$$

$$\text{limit down} = \text{window centre} - \frac{\text{window width}}{2} \quad (4.2.2)$$

If the mean voxel value “v” is inferior to limit up and superior to limit down, the pixel value of the scout x-ray image is calculated using the equation 4.2.3.

$$\text{pixel} = \frac{\left[v - \left(\text{window centre} - \frac{\text{window width}}{2} \right) \right] \times 255}{\text{window width}} = \frac{(v - \text{limit down}) \times 255}{\text{window width}} \quad (4.2.3)$$

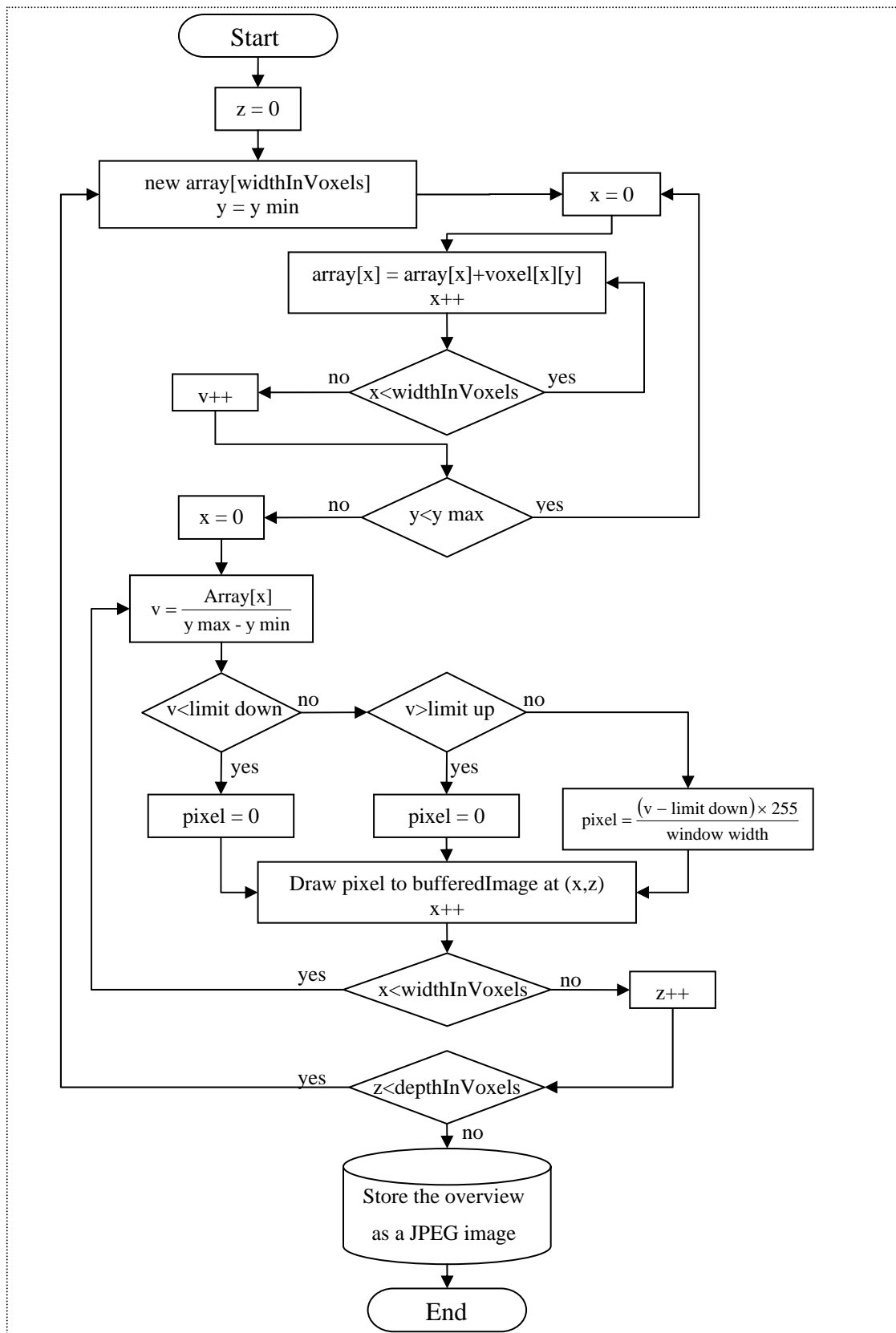


Figure 4-18. Generate a simulated scout x-ray. This flowchart illustrates the process used to calculate the average density of a chosen region of the CT dataset and store the result on a JPEG image.

When we use an array in a Java program, this array is stored in the RAM of the computer. In the present case, our whole CT dataset is stored in the RAM. It is possible to calculate that a CT dataset is made of approximately 140MB which is far superior to the 64MB default RAM size allocated to the Java virtual machine. Therefore, the application must be executed on the command prompt with the parameters “-Xmx512m” and “-Xms512m” that allows the application to use up to 512Mo of RAM. The following commands can be written inside a text file. If the extension of this text file is changed to “.bat”, this file becomes an executable file. The administrator can double-click on this file and the application will start.

```
set path=C:\Program Files\Java\jdk1.5.0_02\bin  
java -Xmx512m -Xms512m -classpath  
C:/Project/CTCTutor/build/web/WEB-INF/classes  
overview.OverviewMaker
```

4.2.3.3. Digitally signing a Java Applet

To be able to write the CT dataset on the client’s hard disk, the Applet classes must be packed into a Jar archive and this Jar must be digitally signed.

4.2.3.3.1. To generate a Jar archive

The Applet classes and the relevant files that are used by the Applet must be packed into a Jar archive. A command line is used to create the Jar archive “CTC_Tutor.jar” using the executable program “jar” that is available in the Java Development Kits (JDK). This command must be executed on the command prompt or in a “.bat” file.

```
jar cf C:/CTCTutor/web/CTC_Tutor.jar -C  
C:/CTCTutor/build/web/WEB-INF/classes .
```

⚠ The last dot is used to specify that all the files present into the directory “classes” should be packed into the Jar archive.

As illustrated in figure 4-19, a Jar archive contains a directory called “META-INF”. META-INF contains the manifest file “MANIFEST.MF” which holds information about the Jar such as the name and version of the program used to create the Jar archive.

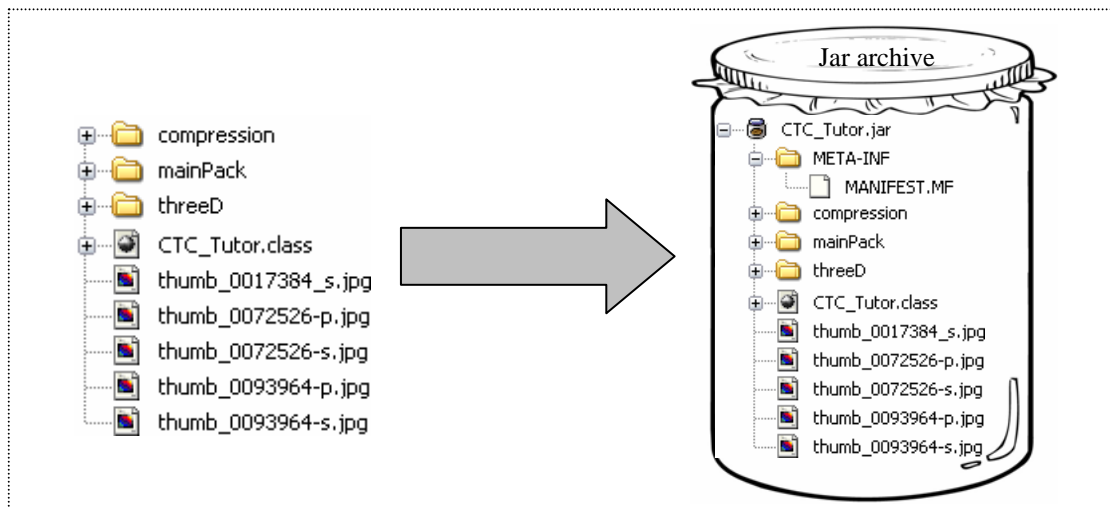


Figure 4-19. Pack the Applet files into a Jar archive.

4.2.3.3.2. The hash algorithm (Secure Hash Algorithm) SHA-1

SHA-1 is a hash algorithm that has been designed by the national security agency (NSA). SHA-1 is used to “hash” the files present inside the Jar archive. More precisely when SHA-1 is applied to a file, it returns a 160 bits digest. An SHA-1 digest is also known as a hash or a digital fingerprint. In theory, SHA-1 will never create the same digest for 2 different files. Therefore SHA-1 digests can be used to check that two files are identical and that they contain the exact same Bytes.

4.2.3.3.3. To generate a RSA key pair

A RSA key pair must be created using the executable program Keytool. Keytool is also available in the Java Development Kits (JDK). A RSA key pair consists of a private key and a public key. The private key will be used to sign the SHA-1 digests related to the files present inside the Jar archive. The public key will be used by the client to cryptographically validate the signed SHA-1 digests. The following command is used to create a RSA key pair named “key_vsg” and identified by the alias vsg.

```
keytool -genkey -keyalg RSA -keystore key_vsg -alias vsg -  
keypass 8x5Lkdf4 -storepass 8x5Lkdf4 -validity 999 -dname  
"CN=Vision Systems Group DCU, OU=Java Code, O=vsg dcu,  
L=dublin, ST=, C=IE, EMAILADDRESS=luautev@eeng.dcu.ie  
DC=vsg, DC=dcu"
```

Following this step, the key pair “key_vsg” is stored in the server’s hard disk.

4.2.3.3.4. Digitally signing the Jar archive

The Applet should be normally signed by a certificate authority (CA) such as VeriSign and Thawte. A CA is responsible to verify our identity and assert that we are the authors of this program. However, our Applet has been self-signed as it would be costly and cumbersome to have each test Applet signed by a CA. The command below is used to self-sign the Jar archive.

```
jarsigner -storepass 8x5Lkdf4 -keystore key_vsg  
C:/CTC_Tutor/web/CTC_Tutor.jar vsg
```

The signature information is added to the manifest “MANIFEST.MF” of the Jar. Our digital signature is stored inside the file “VSG.RSA”. Also the signature file “VSG.SF” contains SHA-1 digests of every file present inside the Jar.

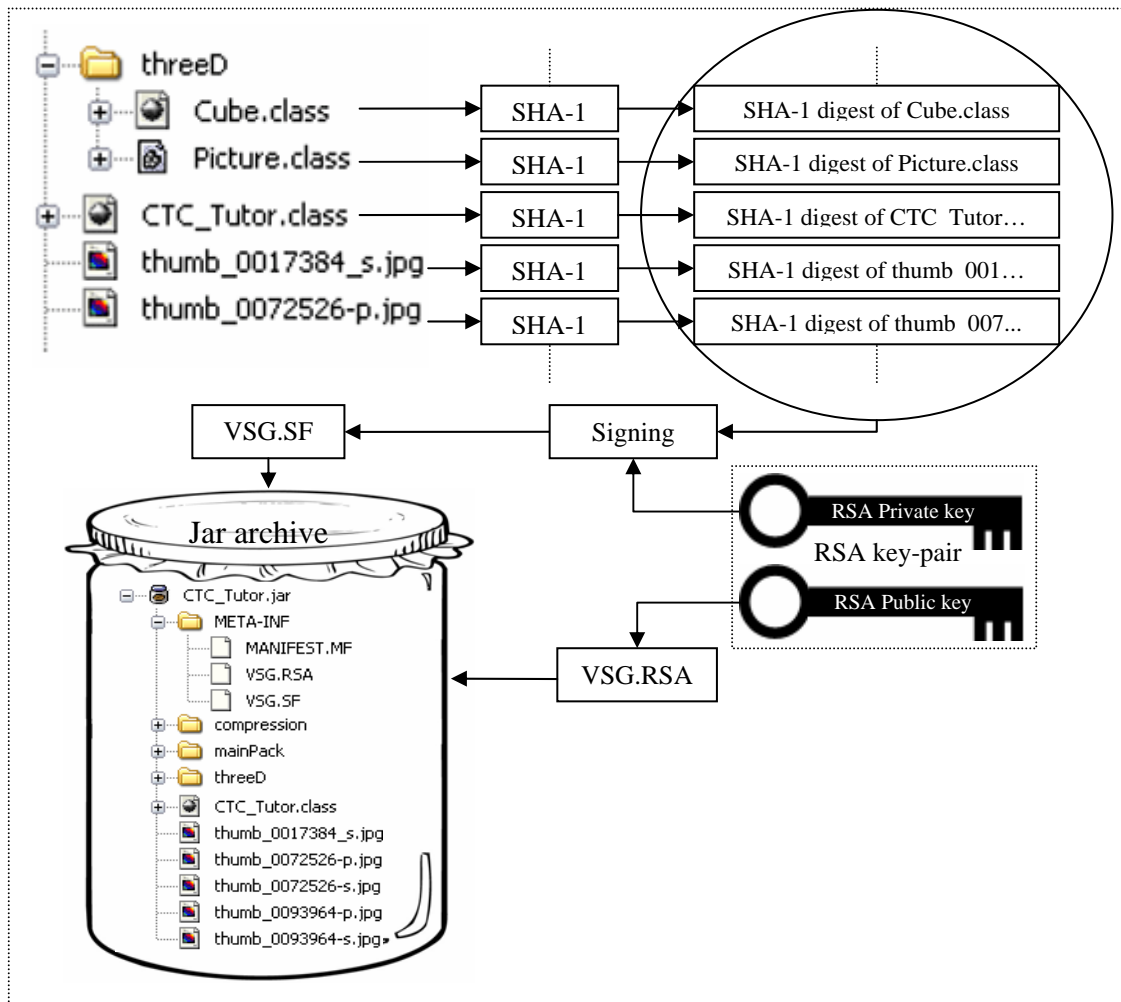


Figure 4-20. Digitally signing the Jar archive. A SHA-1 algorithm hashes each file of the Jar and the remaining digests are stored inside the file VSG.SF after being signed using the RSA private key.

As illustrated in figure 4-20, when signing the Jar archive, SHA-1 digests are calculated for every file present inside the Jar. The resulting digests are signed using the private key of the RSA key-pair. Finally, the signed digests are stored inside the signature file “VSG.SF” and the public key is stored inside the file “VSG.RSA”.

“jarsigner” is an executable programs that is available in the Java Development Kits (JDK). In order to test a new version of the system without having to type again all these command lines in the command prompt, a “.bat” file named as been created. The first line of this file set the path to the JDK directory:

```
set path=C:\Program Files\Java\jdk1.5.0_02\bin
```

Also, the file related to the previous RSA key pair is deleted before creating a new one.

```
del key_vsg
```

Finally, the following code is used on the HTML page to run the Applet:

```
<applet code=" CTC_Tutor.class" archive="CTC_Tutor.jar"  
width="956" height="615">
```

▲ The class and the Jar archive must have the same name or the Applet will not start. Using this code, the Jar archive “CTC_Tutor.jar” is sent from the server to the client side.

4.2.3.3.5. Receive and verify a digitally signed Jar archive

The public key present inside the certificate “VSG.RSA” is used to check the signature of the digests that are present inside the file “VSG.SF”. Finally, the decrypted digests are compared with SHA-1 digests calculated from the files present inside the Jar archive (see figure 4-21).

As we self-signed our Applet instead of having the Applet signed by a CA, when the Applet is started, a warning is displayed (see figure 4-22).

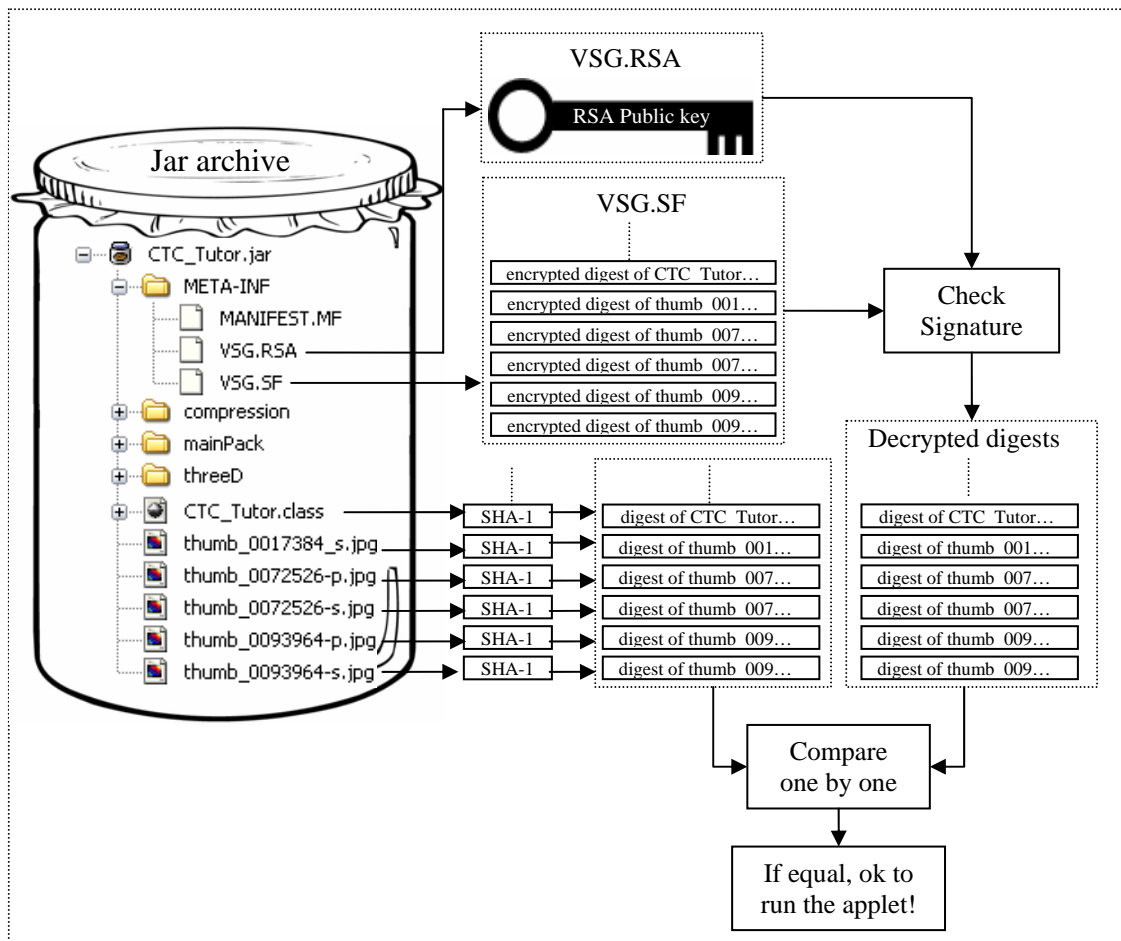


Figure 4-21. Verifying a Jar archive. When the Jar is received on the client side, digests stored in the file VSG.SF after decrypted using the RSA public key. These decrypted digests are then compared one by one with the digests calculated on each file of the Jar. If each digest is identical, it means that the Jar hasn't been altered during its transfer.



Figure 4-22. Security warning before starting the Applet. The user is informed that the Applet hasn't been digitally signed by a CA and asks the user if he/she wants to trust the named publisher.

4.2.3.4. Dataset selection

When running the Applet, the Jar archive is transferred from the server to the user's computer and thumbnails of each dataset's simulated scout x-ray are displayed on the first page of the interface (see figure 4-23).

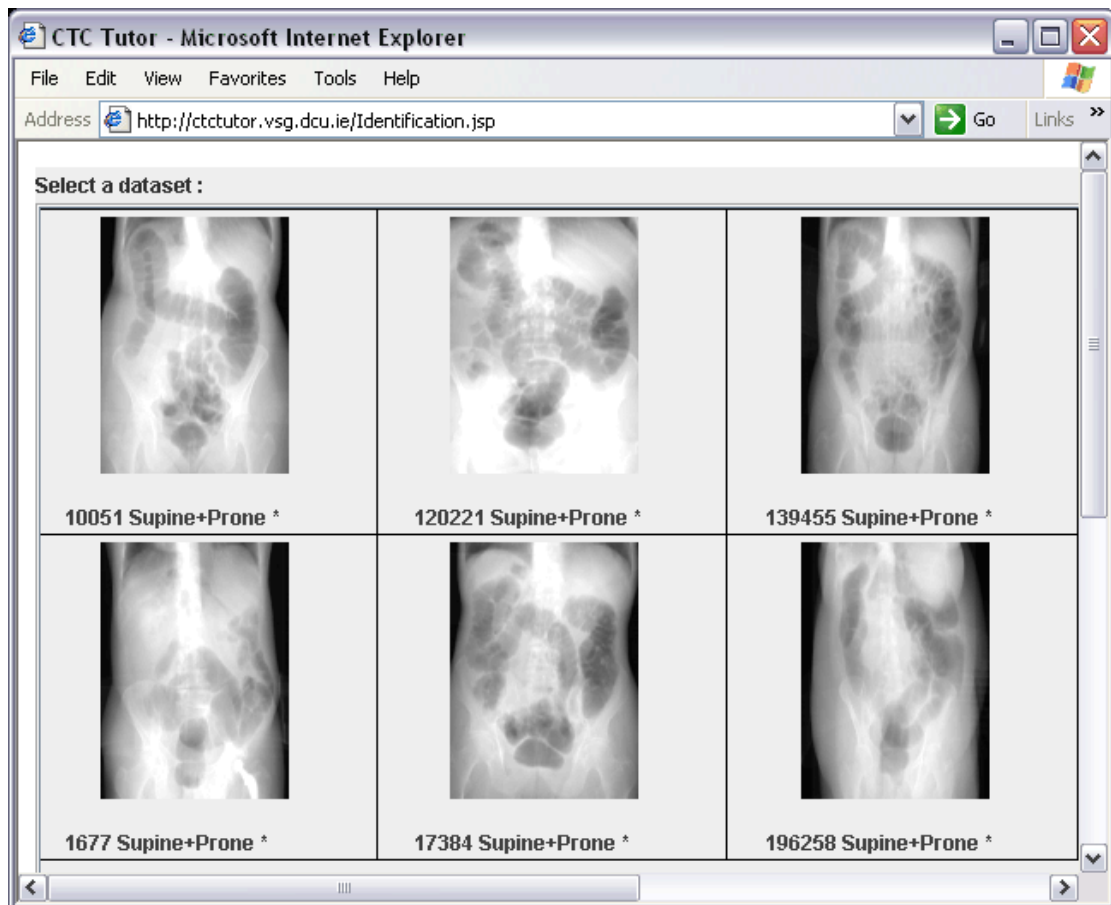


Figure 4-23. Dataset selection. On this page, the user can select the dataset he/she wants to train on.

The trainee can simply click on the thumbnail of the CT dataset he or she wants to train on. On the server side, each registered user has an individual directory. Each time a new CT dataset is studied, a new file containing the user's results is stored in the appropriate user's directory. The flowchart, figure 4-24 illustrates the process used to display the appropriate thumbnails according to the user profile. When a user starts the Applet, each CT dataset that has already been studied is marked with a star "*". The user can then visualise which of the CT datasets he or she has already trained on. In addition, the trainee may choose to train again on a dataset but his/her results will not be overwritten. Therefore, the monitoring of their training will be reliable.

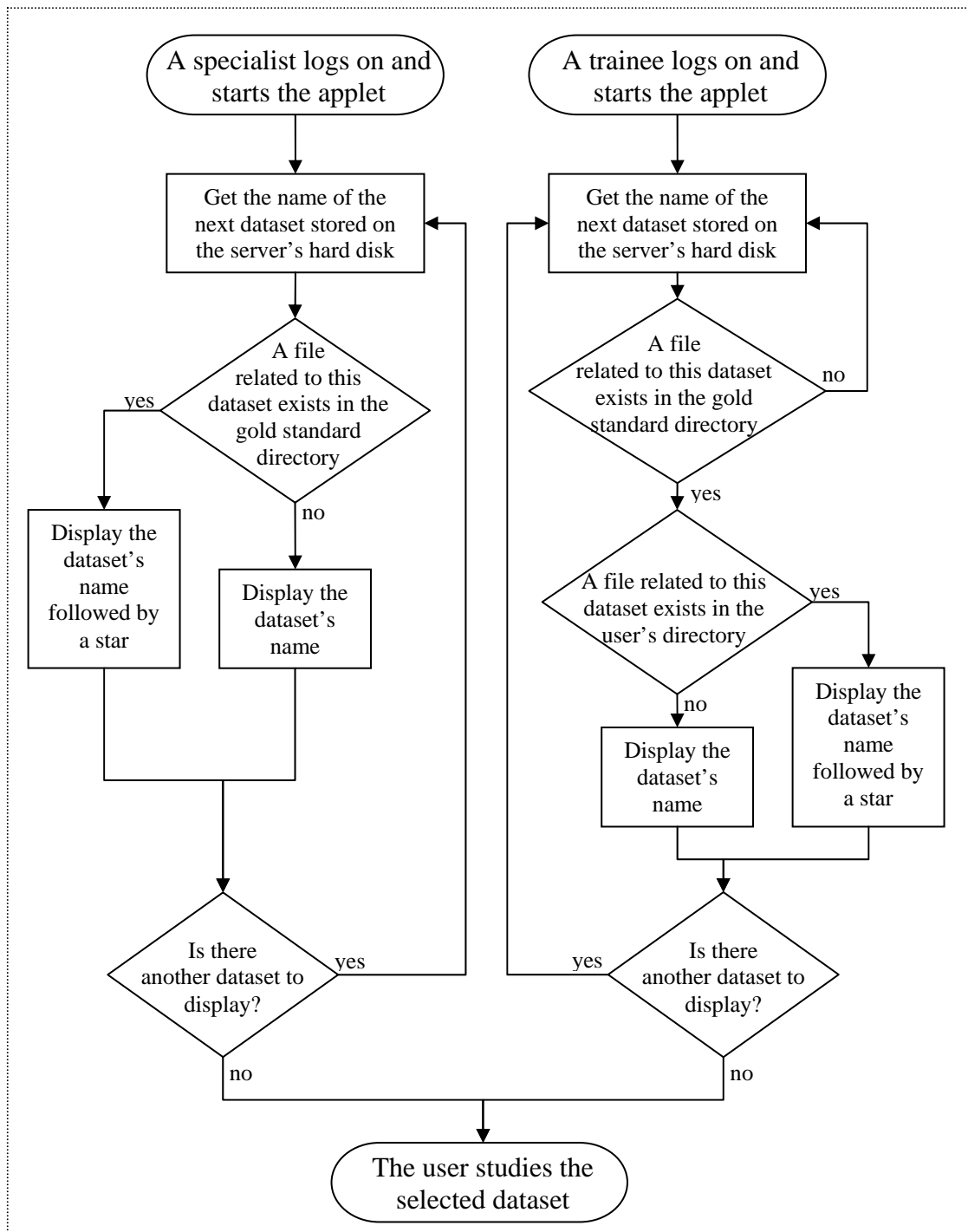


Figure 4-24. Dataset selection. This flowchart illustrates the process used to display the relevant scout x-ray.

4.2.3.5. Multi-user server

Three approaches have been tested to establish connection between the server and the client in order to send CT datasets. The first idea is to simply include the CT dataset in a Jar file. The implementation of this approach is very easy but very limited. The second approach involves developing a dedicated server application intended to keep

listening to client request. This server uses a socket on a specific port. The socket cannot work on the standard HTML port (port 80) because this port is already used for the home page of the system. This aspect can be a problem with some firewalls which block port other than port 80. Another approach has been developed using a simple Servlet. These three approaches are described below.

4.2.3.5.1. Encapsulate a dataset into a Jar file

A CT dataset can be easily packed inside a Jar using the “jar” program that is included in the JDK. After a dataset has been included in the Jar, the dataset is compressed and its size is divided by approximately 2. However this approach has 2 drawbacks. First, loading a Jar archive containing the CT dataset takes an extensive amount of time. During this time, the user is forced to wait without any information related to the loading progress. Also a Jar archive must be created for each dataset. The dataset and the Applet classes are included inside the same Jar. Therefore, updating the Applet code involves generating the Jar archive for each dataset again.

4.2.3.5.2. Server using sockets

A CT dataset can be sent from the server to the client using an `ObjectInputStream` and an `ObjectOutputStream`. When the user clicks on the selected x-ray thumbnail of the dataset he or she wants to train on, the Applet sends the name of the selected dataset to the server. In practical terms, the Applet creates a new socket on port 5050 using the following code:

```
Socket socket=new Socket(this.getCodeBase().getHost(),5050);
```

A socket can be described as a tube connecting the client to the server. After that, the class `ObjectOutputStream` is used to send data through this socket from the Applet to the server and the class `ObjectInputStream` is used to receive data from the server. This new approach allows the Applet interface to display a 2-D slice as soon as it is received on the client’s computer.

On the server side, a class “Server” is responsible for welcoming the new clients. This class contains a thread dedicated to wait for new clients to connect. When the Applet creates a new socket to send the name of the selected dataset, the class “Server” accepts it and instantiates the class “HandleConnection”. The new socket is given as a parameter of the constructor of the class “HandleConnection”. Then,

“HandleConnection” can start sending the dataset to the new client (see figure 4-25).

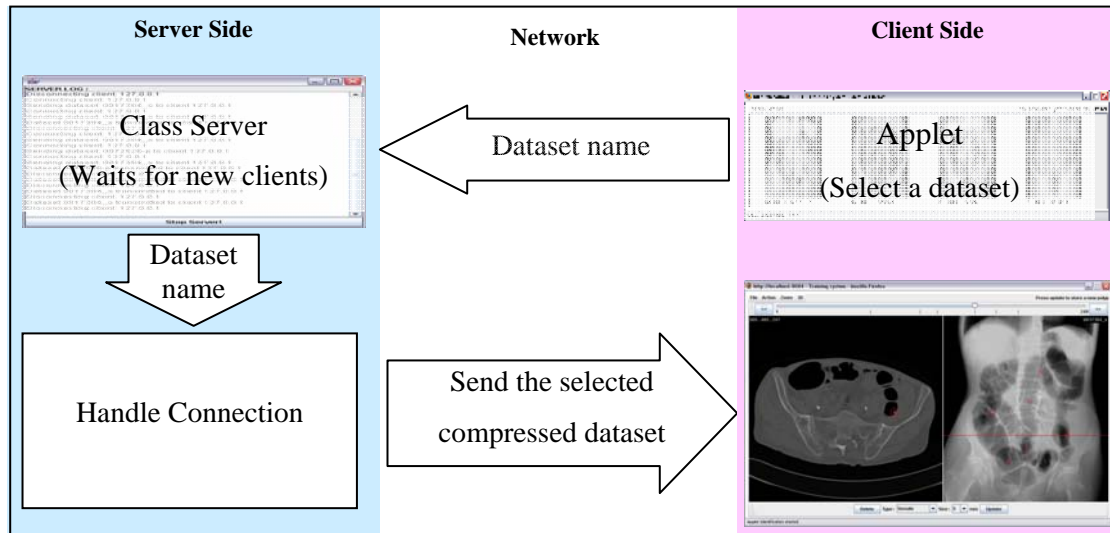


Figure 4-25. Sending a selected dataset from the server to the client.

As illustrated in figure 4-26, the class “Server” is only instantiated once and is responsible to wait for new clients. The class Handle Connection is instantiated each time a new client creates a socket.

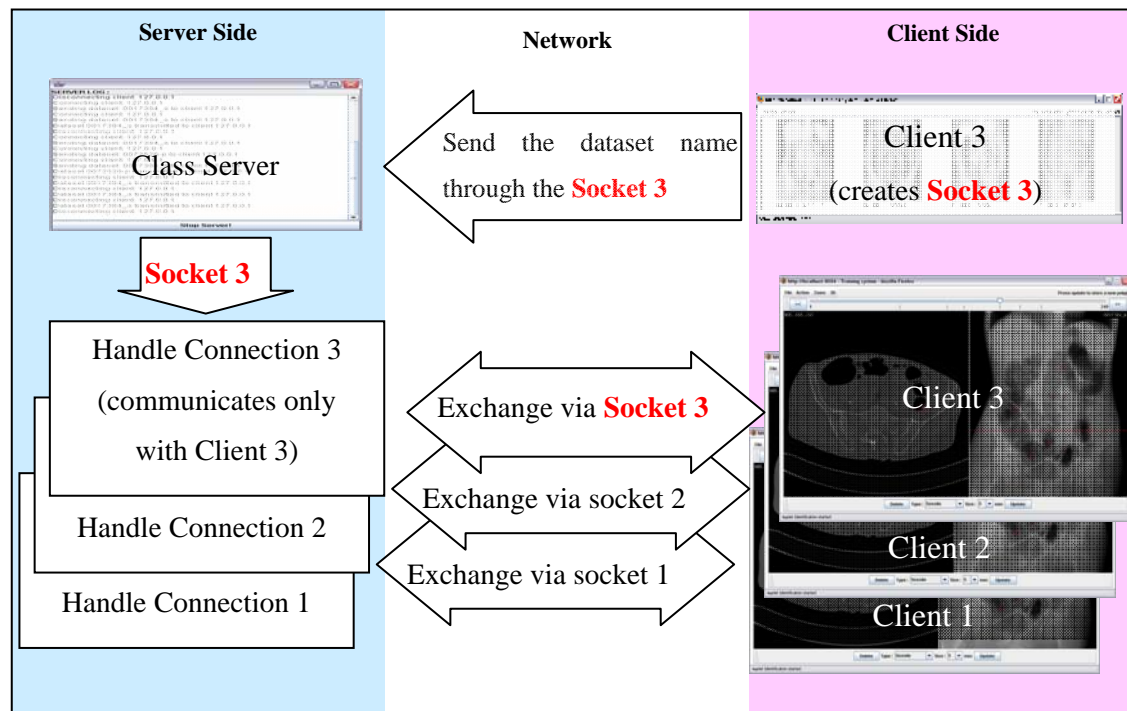


Figure 4-26. Handle connection of several clients. The server classes welcomes a new user and passes the corresponding.

A simple interface has been implemented on the server (see figure 4-27). This interface allows an administrator to start and stop the server and provides a monitoring of data exchange with the different remote computers.

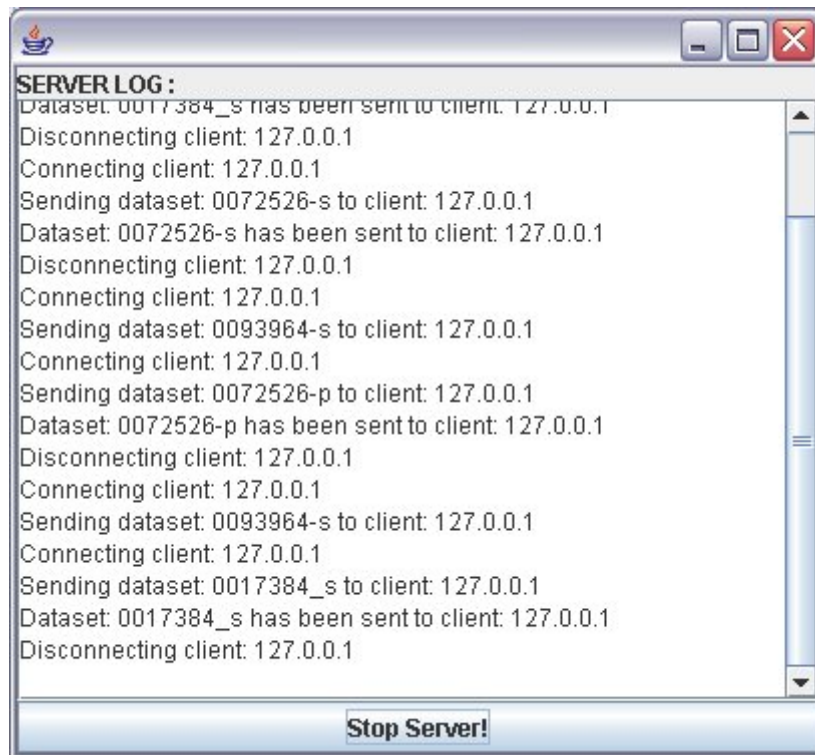


Figure 4-27. Interface of the server.

⚠ To use a scroll bar on a text area, we need to use the method `setCaretPosition()` of the class `JTextArea` each time a new line is appended.

```
JTextArea.append("Connecting client:"+ str_localIP+"\n");
JTextArea.setCaretPosition(JTextArea.getDocument().getLength());
```

4.2.3.5.3. Server using a Servlet

This approach consists of sending a CT dataset from the server to the client using the Java class `URLConnection`:

```
URL url=new URL(getDocumentBase(), "servlet/datasetProvider");
URLConnection conURL = url.openConnection();
conURL.setDoOutput(true);
```

When the user clicks on the scout x-ray thumbnail corresponding to the dataset that he or she wants to train on, the Applet sends the name of the selected dataset to the server. In practical terms, the Applet creates a `PrintWriter`:

```

PrintWriter pw=new
printWriter(conURL.getOutputStream(), true);
pw.println(str_datasetSupine);
pw.close();

```

Following this step, An *ObjectInputStream* is used to receive data from the server:

```

ObjectInputStream is = new
ObjectInputStream(conURL.getInputStream());

```

This approach also allows the Applet to display a 2-D slice as soon as it is received on the client's computer.

On the server side, a class “datasetProvider” is responsible to welcome the new clients. This class is a Java Servlet and is instantiated by the Applet. This Servlet uses an *ObjectOutputStream* to send each compressed slice of the dataset to the client (see

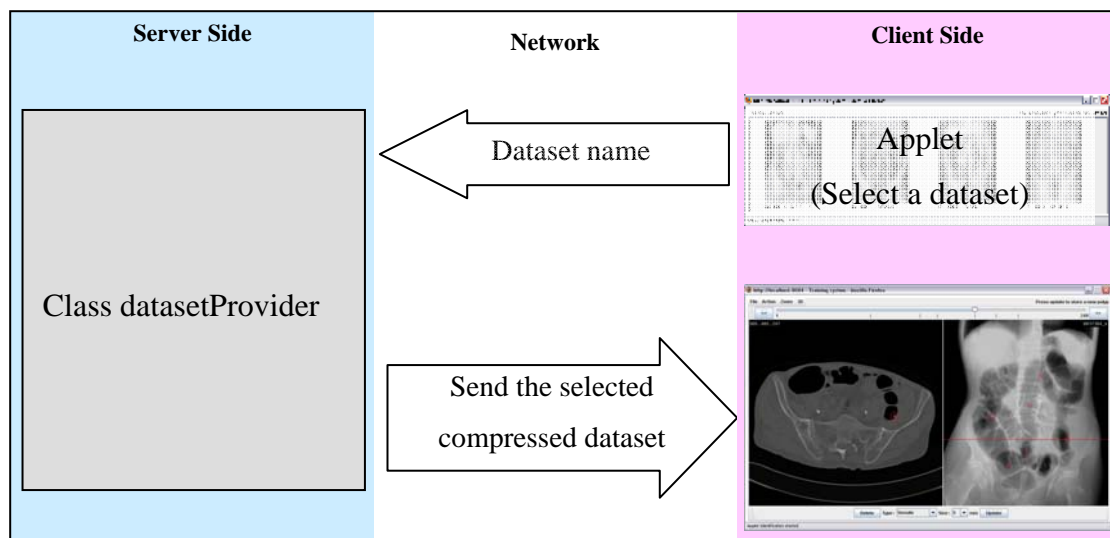


figure 4-28).

Figure 4-28. Sending a selected dataset from the server to the client.

In this implementation, a CT dataset is transferred through the standard HTML port 80. At the beginning, this approach was supposed to be the best solution to pass firewalls. The experience demonstrated that the firewall of the Mater hospital in Dublin blocks the Applets which transfer data through the port 80. On the other hand, computers inside the Mater hospital can download datasets via the Applet implementing the server with socket approach. As a result, both approaches (server

with socket and server using Servlet) have been implemented in the Applet. The user can choose the server he or she wants to use before starting the Applet.

4.2.3.6. A separate thread to display the downloading progress

Following the dataset selection, the Applet will start downloading the chosen dataset from the server to the end user's hard disk. The downloading progress is displayed on the Applet interface to inform the end-user. For the user registered as a specialist, a half transparent mask progressively reveals the simulated scout x-ray as illustrated in figure 4-29.

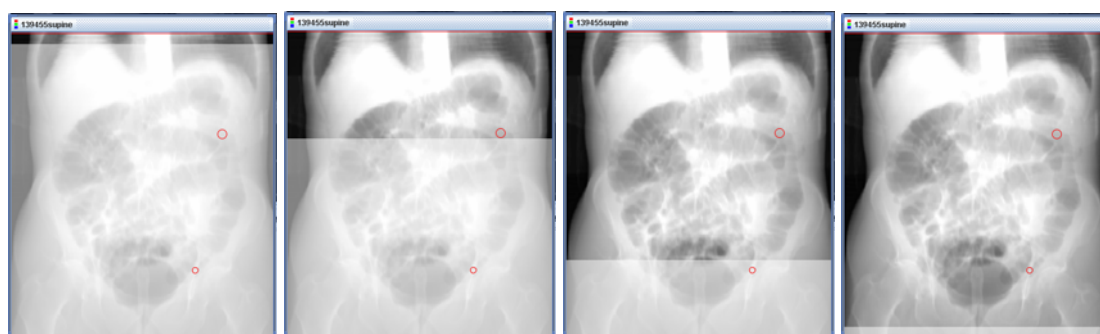


Figure 4-29. Downloading progress for a specialist.

To display the downloading progress while downloading the dataset, a new thread must be implemented by the Applet. The new thread is exclusively used to download the CT dataset while the main thread of the Applet keeps listening to the user's actions. Therefore, the specialist can start working on the initial images while the rest of the dataset is being transferred. The type of the *BufferedImage* used to create the mask must be *TYPE_INT_ARGB*. This type includes a transparency component that is necessary to create a half transparent mask.

For the user registered as a trainee, the downloading progress is displayed in front of the interface and prevents the user from accessing the interface until both supine and prone datasets have been downloaded (see figure 4-30).

The main advantage of this approach is to be able to start timing the reading time at the same position for every trainee whether they have broadband or a slow connection. The reading time is one important factor to later establish a learning curve of the CTC training.

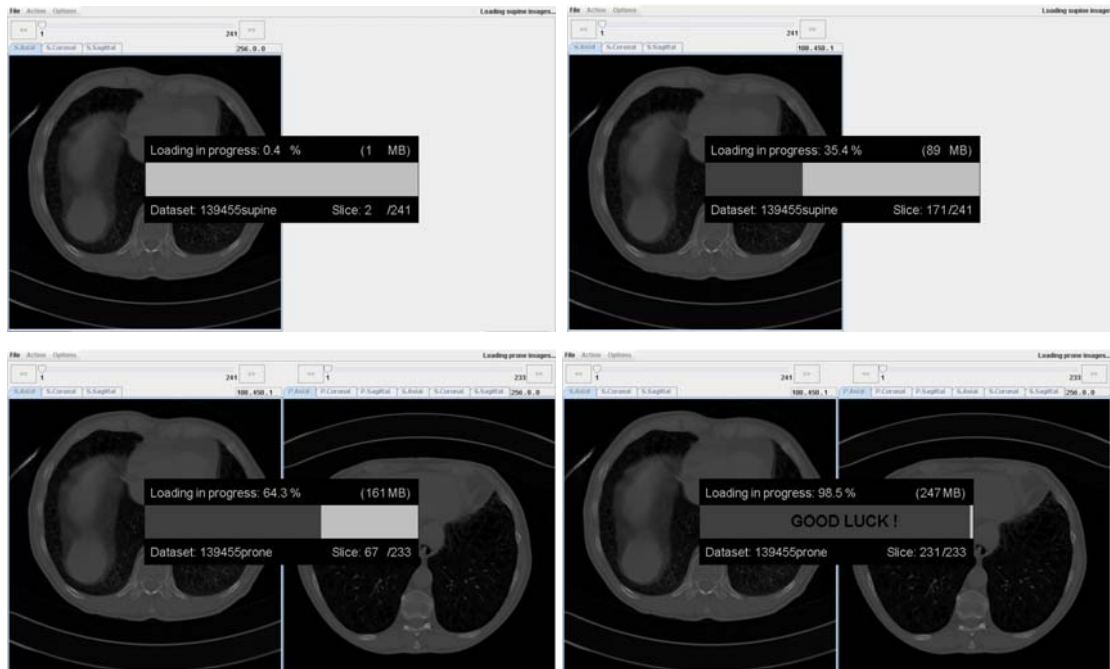


Figure 4-30. Downloading progress for a trainee.

Technically, the main Applet class called “CTC_Tutor” must implement *Runnable*. After that, the class *Thread* is instantiated and the method *start()* is called.

⚠ It is possible to specify the priority of the thread. The priority of this new thread must be low in order to grant the maximum priority to the main thread of the interface in case a specialist wants to use the interface while the Applet is downloading the datasets.

```
Thread thread = new Thread(this);
thread.start();
thread.setPriority(Thread.MIN_PRIORITY);
```

When the method *start()* is called, the method called *run()* is actually executed. This method *run()* must be implemented on the Applet. In practical terms, this method holds the code used to receive, uncompress and store a dataset in the client’s hard disk. The dataset is stored as a file using a *DataOutputStream*. Each time a new slice is written in this file, the progress bar is updated.

4.2.3.7. Retrieve 2-D slices on the client's side

Each 2-D axial image of the selected dataset is received, uncompressed and stored in a file in the user's hard disk. The transfer of a compressed dataset can fall into three parts:

- The server starts by sending the header of the dataset as a *Hashtable* object.
- After that, the server sends the Huffman symbols.
- The server finally sends the compressed slices Byte by Byte.

From the last point, each bit of the received Bytes goes through the decoding process as illustrated in the flowchart, figure 4-31. This process starts by reading 2 bits in order to check if a "change length code" $(01)_2$ or a "Switch to Huffman" $(00)_2$ code is present.

1. If the value of the first 2 bits is equal to $(01)_2$, the 4 next bits are used to find the new length value. After that, the voxel value is retrieved by reading a number of bits defined by the new length.
2. If the value of the first 2 bits is equal to $(00)_2$, the program reads the next bits and retrieves the corresponding Huffman symbol (see table2).
 - a. If the Huffman symbol equals to "sign code", the next voxel value equal (- next Huffman symbol).
 - b. If the value of the Huffman symbol is "up 1", the 6 next bits will be used to retrieve the voxel value.
 - c. If the value of the Huffman symbol is "up many", the 4 next bits are used to find the new length value and the next voxel value is retrieved by reading a number of bit equal to the new length.
 - d. If the value of the Huffman symbol is different than "sign code", "up 1" or "up many", the next voxel equal this Huffman symbol.
3. After an axial image has been received and decoded with the combination of "limited length encoding" and Huffman coding, the original axial image is retrieved after applying the motion estimated and compensated technique.

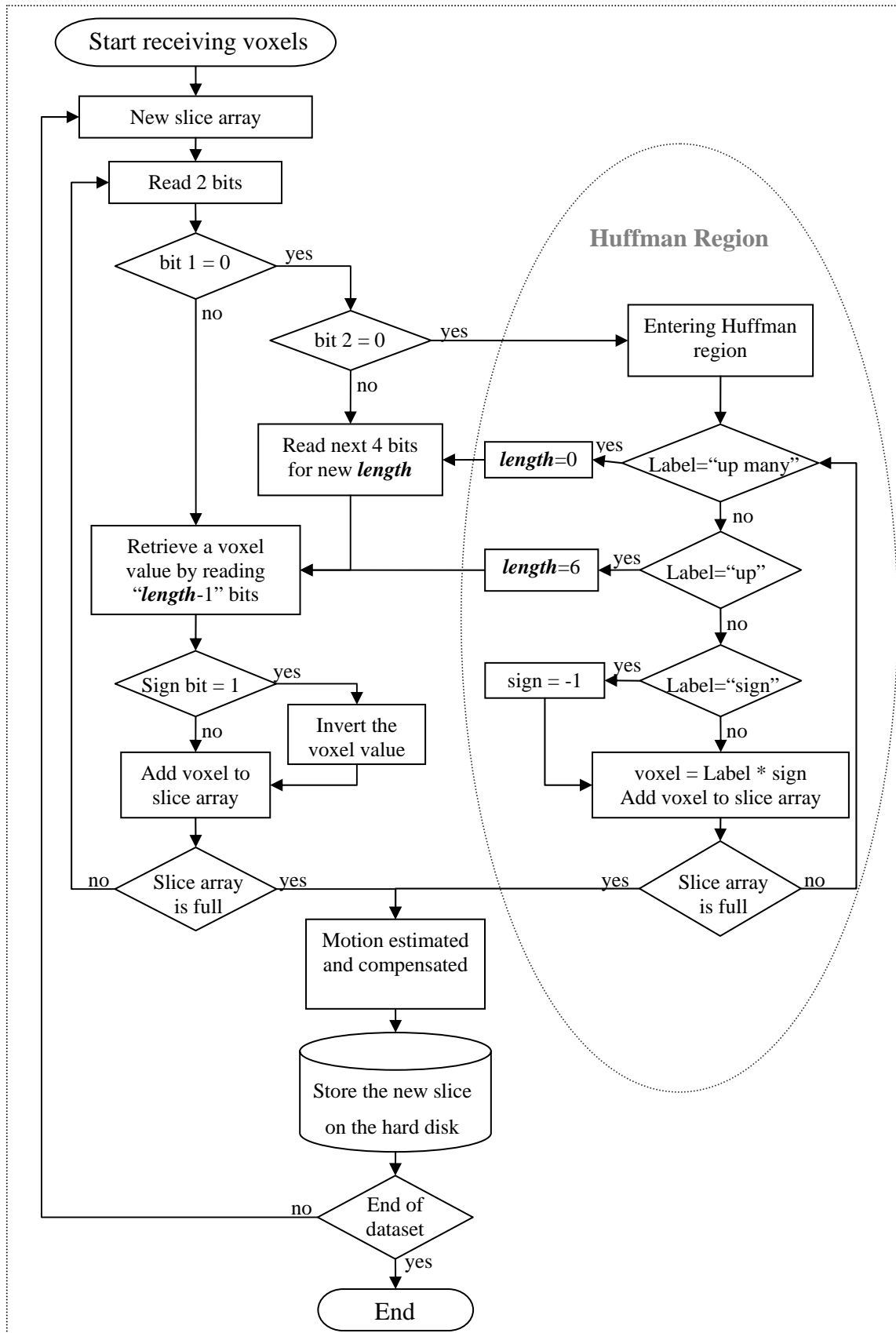


Figure 4-31. Decoding process. This flowchart illustrates how each axial image is decoded bit by bit when it is received on the client side.

4.2.3.8. Generate an axial image

Once a 2-D axial image of a dataset has been transferred from the server to the client's hard disk, the user can display it via the Applet interface. The user can specify the slice number that he/she wants to display using the navigation tool provided by the Applet. To read a dataset from the Applet, the class *DataInputStream* is used.

⚠ The method *readFully()* is used to read an entire axial image. This approach is faster than reading voxels using a *RandomAccessFile* class as we did in the first implementation.

This *BufferedImage* is down to a *JLabel* using the *setIcon()* method. The trainee is requested to highlight polyp candidates by flagging locations via circles superimposed on 2-D axial images (see figure 4-32). Also, trainees have to define their level of confidence, the size (in mm) and the type of the polyp e.g. "sessile" or "pedunculated". When flagging new polyps, ticks and circles are drawn on the navigation slider and on the simulated scout x-ray respectively in order to show which of the dataset slices contain identified polyps.

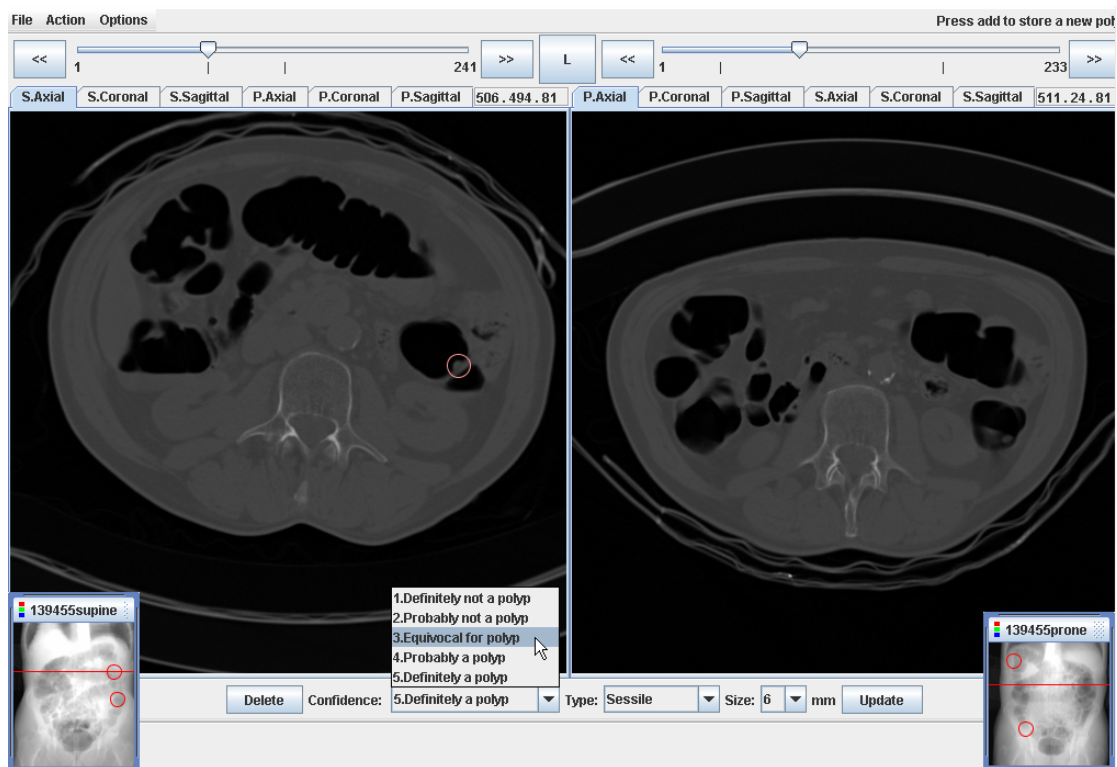


Figure 4-32. Flagging a potential colorectal cancer polyp. When the user clicks on a 2-D slice, a dedicated tool bar appears at the bottom of the interface and the user can select his/her level of confidence, the type and the size of the new polyp.

4.2.3.9. Reformatted images

4.2.3.9.1. Generate a coronal image

A recent study conducted by Sebastian (2006) has highlighted that radiologists using coronal reformatted images instead of axial images reports more findings and a higher degree of confidence in routine abdominal CT. This highlights the fact that the interface of our system must be able to generate coronal images in addition to axial images.

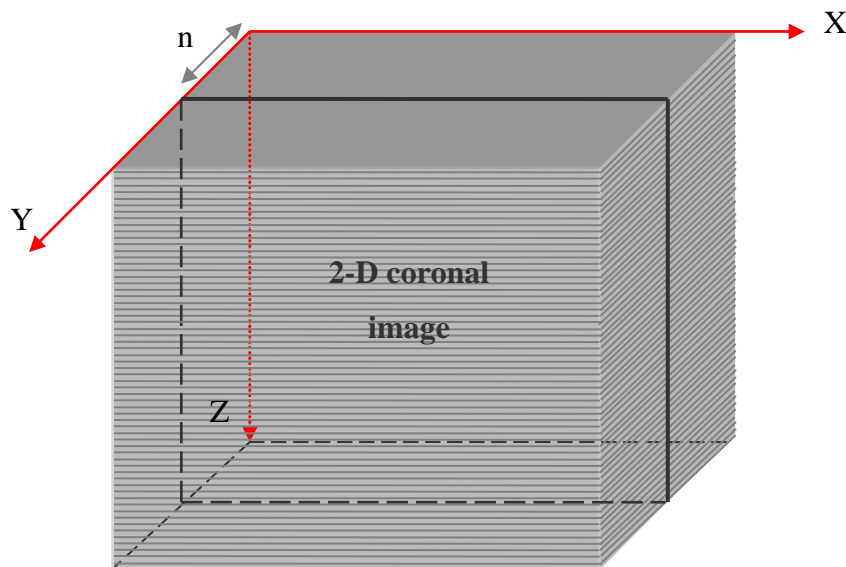


Figure 4-33. A 2-D coronal image in a CT dataset. There are as many coronal images as there are pixels along the Y axis.

Two techniques have been tested to generate coronal images. The most intuitive technique involves reading one specific width of each axial image in the dataset. For instance, to read the 2-D coronal image highlighted in figure 4-33, the program skips $2 \times n \times widthInVoxels$ Bytes and reads the next $2 \times widthInVoxel$ Bytes. After that, the program incrementally skips $2 \times (heightInVoxels - 1) \times widthInVoxels$ Bytes and reads $2 \times widthInVoxel$ Bytes until it reaches the end of the dataset. In this thesis, this technique will be referred to as “skip Bytes”.

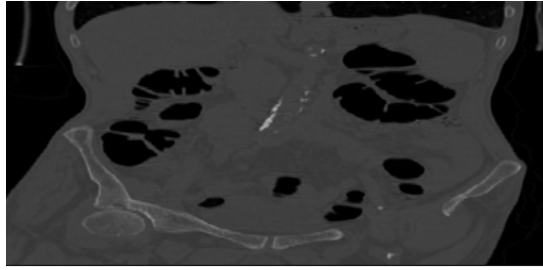


Figure 4-34. Coronal image.

The second technique involves using the method *readFully()* of the class *DataInputStream*. In this thesis, this technique will be referred to as “readFully”. Usually, a program runs much faster using data previously stored inside the RAM than reading the relevant voxels one by one in the hard disk. The limitation of the standard JVM prevents the program from storing a whole dataset inside the RAM. However, it is possible to incrementally store each axial image of the dataset inside the RAM and use the relevant width of each slice. Technically, the program incrementally stores an axial image ($2 \times heightInVoxels \times widthInVoxels$ Bytes) inside an array, reads the relevant voxels and overwrites this array with the next axial image.

There are as many coronal images as there are pixels along the Y axis of an axial image (512). Sebastian pointed out that the increasing number of axial images generated by modern CT scanners directly increases the interpretation time of axial images whereas interpretation time of coronal images remains the same. This is another reason why it is interesting to train radiologists for colorectal cancer screening using coronal images (see figure 4-34).

4.2.3.9.2. Generate a sagittal image

Modern analysis tools usually provide sagittal image visualisation in addition to coronal and axial images. Many studies of the colon are based on these three views e.g. (Aufort et al. 2005).

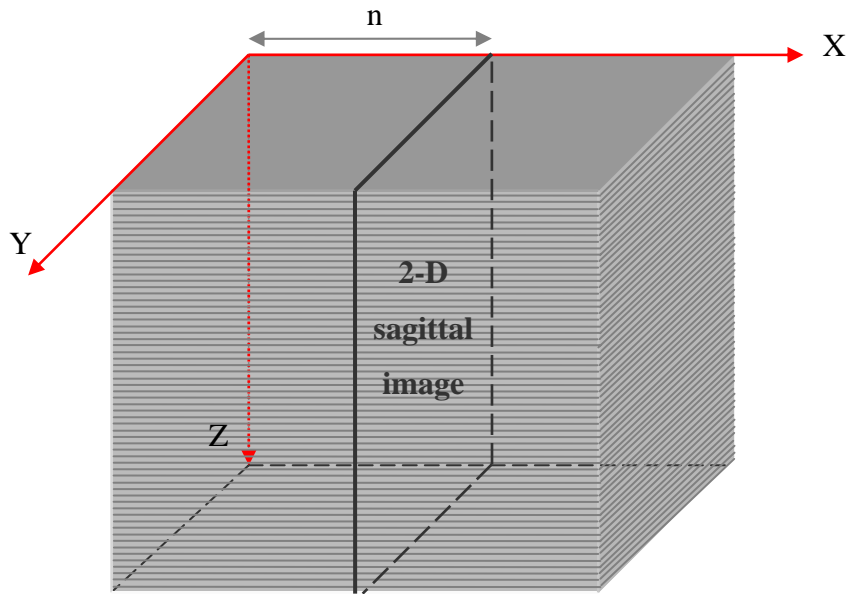


Figure 4-35. A 2-D sagittal image in a CT dataset. There are as many sagittal images as there are pixels along the X axis of an axial image.

The techniques “skip Bytes” and “readFully” previously discussed can be used to generate a sagittal image. This time, the skip Bytes technique consists of reading only one pixel of each axial image’s width. For instance, to read the 2-D sagittal image highlighted in figure 4-35, the program starts by skipping $2 \times n$ Bytes. Then, the program reads the next voxel and skips $2 \times widthInVoxel - 2$ Bytes. After that, the program will incrementally read one voxel and skip $2 \times widthInVoxel - 2$ Bytes until it reaches the end of the dataset.

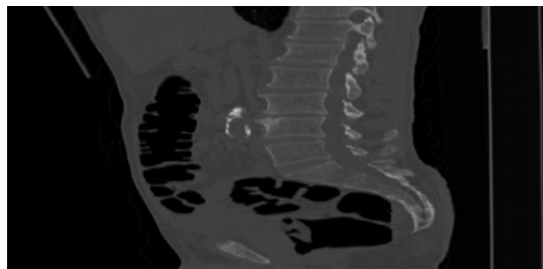


Figure 4-36. Sagittal image.

To generate a sagittal image using the “readFully” technique (see figure 4-36), each axial image is incrementally stored inside the RAM of the client’s computer and the relevant height of each axial image is used to generate the sagittal image.

4.2.3.9.3. Realistic reformatted image

The height of a reformatted image (coronal or sagittal) is equal to the number of axial images in the dataset. In the datasets that are used in this project, the number of axial images per dataset is about 250. Therefore the height of a reformatted image is 250 pixels. The width of a coronal image is equal to the width of an axial image (512 pixels) and the width of a sagittal image is equal to the height of an axial image which is also 512 pixels. The reformatted image must be stretched in order to be a realistic representation of the patient's body (see figure 4-37).

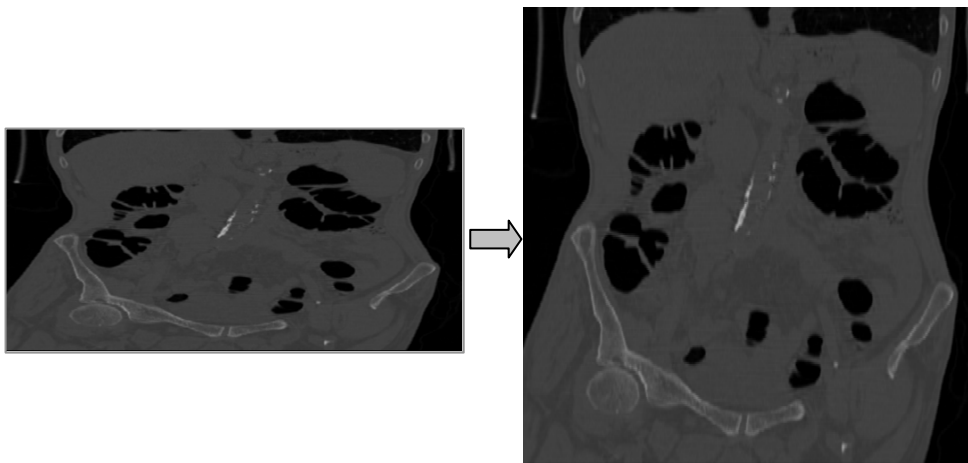


Figure 4-37. Stretching a coronal image to realistic proportions.

The realistic proportion of a reformatted image can be found by using the “voxelWidth” or “voxelHeight” and the “voxelDepth” which are present in the header of a dataset. The voxelWidth and voxelHeight are the number of millimetres that one pixel represents on the width and on the height of an axial image. The voxelDepth can be described as the thickness (in millimetres) of an axial image. For example, the actual width of a coronal image in millimetres is equal to the width of an axial image multiplied by the voxelWidth (see equation 4.2.4) and the height is equal to the number of axial image inside a dataset multiplied by the voxelDepth (see equation 4.2.5).

$$\text{coronalWidth} = \text{widthInVoxels} \times \text{voxelWidth} \quad (4.2.4)$$

$$\text{coronalHeight} = \text{depthInVoxels} \times \text{voxelDepth} \quad (4.2.5)$$

The actual dimensions of sagittal images in millimetres (see figure 4-38) are calculated using the equation 4.2.6 and 4.2.7.

$$\text{sagittalWidth} = \text{heightInVoxels} \times \text{voxelHeight} \quad (4.2.6)$$

$$\text{sagittalHeight} = \text{depthInVoxels} \times \text{voxelDepth} \quad (4.2.7)$$

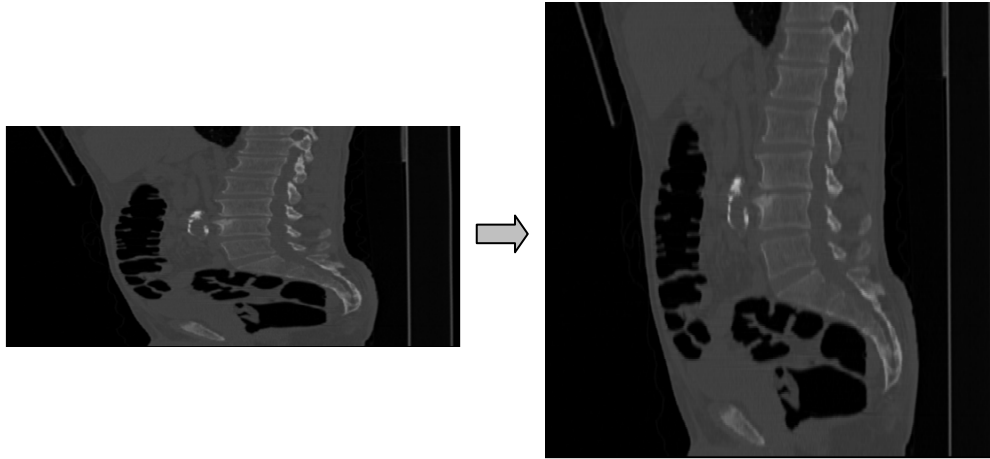


Figure 4-38. Stretching a sagittal image to realistic proportions.

The Java method `getScaledInstance()` is used to stretch the initial reformatted image. The last parameter of this method is the interpolation method. `image_SCALE_SMOOTH` is chosen as the interpolation method in order to give higher priority to image smoothness than scaling speed. Time spent to resize a `BufferedImage` is approximately 0.5 seconds using a PC with an Intel Pentium 4, 2.8GHz and 512MB of RAM.

4.2.3.9.4. Region of interest for reformatted images

Computation time spent to generate reformatted images with realistic proportions was calculated using a PC with a 2.8 GHz Intel Pentium 4 processor, 512 MB of RAM and XP. Forty reformatted images have been generated with the “skip Bytes” method and forty reformatted images have been generated with the “readFully” method.

		Coronal image		Sagittal image	
		Skip Bytes	Readfully	Skip Bytes	Readfully
Full image	Before buffering	1.4	7.7	4.7	8.7
	After buffering	0.8	1.1	1.2	1.1

Table 4-4. Average time (in second) to generate reformatted images with “skip Bytes” and “readFully” methods.

After generating a first group of reformatted image, the computer that was used for this test automatically buffered the dataset inside the RAM and the subsequent reformatted images was generated in approximately 1 second with each method. The computer stores inside the RAM memory, data that are repeatedly read in the hard disk. Unfortunately, the first group can vary from 10 to over 30 images. This number depends of the availability and the size of the RAM. Also, some computers with 128MB of RAM will never be able to store a dataset which is usually about 150MB.

Table 4-4 shows that the “skip Bytes” method is more efficient than the “Readfully” method to generate a reformatted image before the dataset has been buffered inside the RAM. After the buffering of the dataset, table 4-4 shows that the “Readfully” method is slightly more efficient to generate a sagittal image. However, 0.1s difference is not significant for this test and it is possible to consider both methods equivalents for generating a reformatted image after buffering.

A response time equal to a few seconds is not comfortable for our end-users. To tackle this issue, a solution is to generate a region of interest of the reformatted image (see figure 4-39).

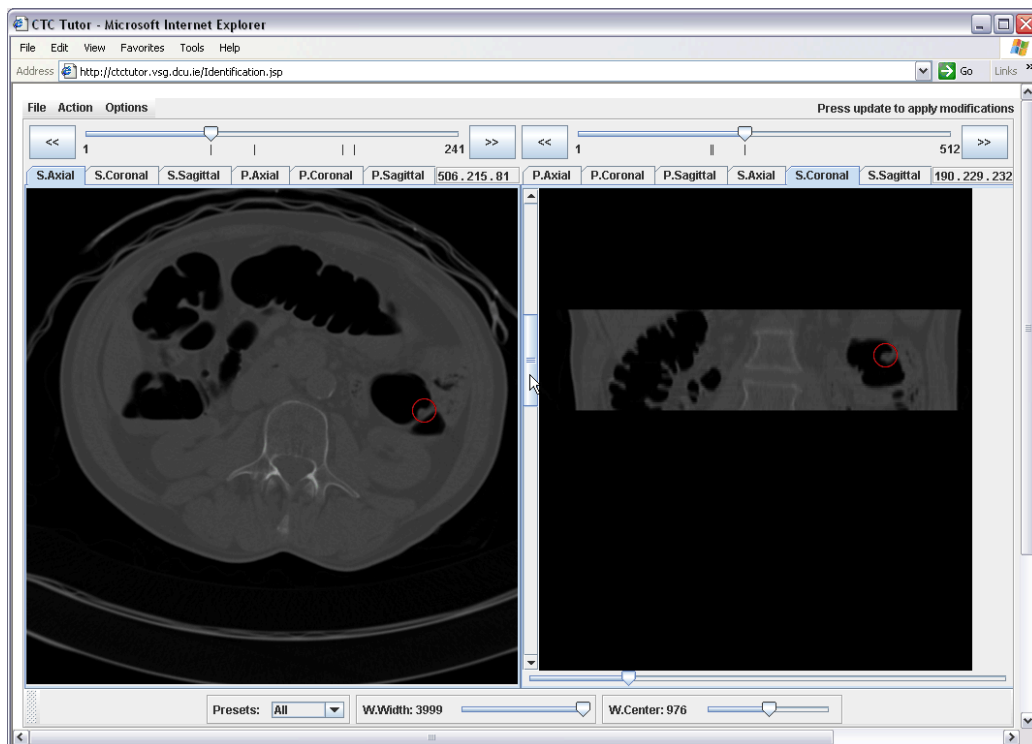


Figure 4-39. Axial image and region of interest of a coronal image.

Table 4-5, shows that time taken to generate the region of interest of a coronal image with the “Skip Bytes” is slightly more efficient than the “Readfully” method. Time spent to generate the region of interest of a sagittal image is similar with both methods. As a result, the method “Skip Bytes” is more efficient in most cases and this method has been implemented on the Applet.

	Coronal image		Sagittal image	
	Skip Bytes	Readfully	Skip Bytes	Readfully
Region of interest (using 100 axial)	0.5	0.6	0.6	0.6
Region of interest (using 25 axial)	0.3	0.4	0.4	0.4

Table 4-5. Average time (in second) to generate different region of interests.

On the interface, a horizontal slider and a vertical scrollbar surround the reformatted image. The user can drag the horizontal slider to increase or decrease the height of the region of interest and can drag the vertical slider to display another section of the image.

4.2.3.10. Dataset navigation

Using the Applet interface, the navigation through 2-D slices can be performed by dragging a slider or clicking on a forward/backward button. Also, to meet the requirement of our medical colleagues, the rotation of the mouse wheel can be used to scroll images. The listener method simply retrieves the number of mouse wheel notches (which may be positive or negative) and adds it to the currently displayed picture’s number.

The interface also provides a resizable window displaying the simulated scout x-ray of both supine and prone datasets. The height of the scout x-ray image represents the depth of the dataset (along z axis). Therefore, if an axial image is displayed, the user can directly click on the associated scout x-ray to jump to the desired slice. The interface will display the axial image that is equal to the y coordinate of the mouse click on the scout x-ray. A horizontal red line illustrates the position of the current axial image in the dataset (see figure 4-40).

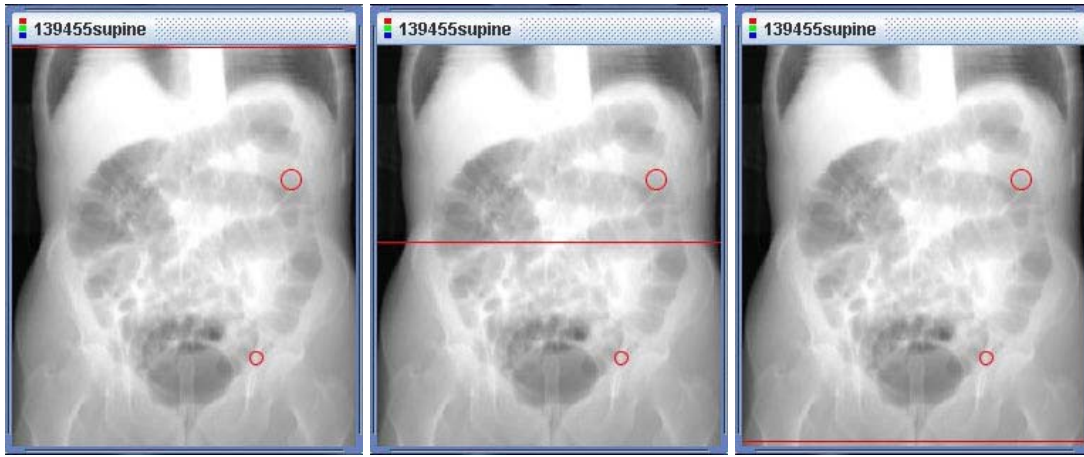


Figure 4-40. Simulated scout x-ray illustrating the position of one of the first, middle and last axial image in the dataset

If a coronal slice is displayed, a rectangle drawn on the simulated scout x-ray illustrates the position of this coronal slice in the dataset (see figure 4-41). The first coronal image is represented as a large rectangle whereas the last one is represented as a small rectangle in the centre of the scout x-ray.

If the user clicks in the centre of the scout x-ray, the interface displays the last coronal slice in the dataset whereas if the user clicks on the border of the scout x-ray, the interface jumps to the first coronal image of the interface.



Figure 4-41. Simulated scout x-ray illustrating the position of one of the first, middle and last coronal images of the dataset.

The simulated scout x-ray corresponding to a sagittal image illustrates the position of the current sagittal slice as a vertical red line (see figure 4-42).



Figure 4-42. Simulated scout x-ray illustrating the position of one of the first, middle and last sagittal image of the dataset.

If the user clicks on a polyp circle drawn on a simulated scout x-ray, the corresponding image of the interface will jump to the relevant slice and the interface will display the appropriate panel used to modify or delete this specific polyp.

4.2.3.11. View tabs and synchronisation

The system is designed to handle 2 different images at the same time; one on the left and another one on the right of the interface (see figure 4-43). The user can choose among 6 possible views on the left and 6 possible views on the right by clicking on one of the tabs located between the navigation slider and the image. For each image, 3 tabs refer to the supine dataset and 3 tabs refer to the prone dataset. The name of the tabs referring to the supine dataset starts with a S whereas the tabs referring to the prone dataset starts with a P. Using these tabs it is possible to choose between supine-axial, supine-coronal, supine-sagittal, prone-axial, prone-coronal and prone-sagittal.

By default, the left image is a supine-axial and the right image is a prone-axial. This configuration has been chosen by our medical collaborators. Our end-users can also display 2 images of the same dataset. In this case, when the user clicks on one image, the system gets the coordinates of the mouse click and determines the most relevant slice that should be displayed on the second image. For example the user can choose to display a supine-axial on the left and a supine-sagittal on the right (see figure 4-43). In this case, if the user clicks on the supine-axial, the interface will display the sagittal slice that is equal to the x coordinate of the mouse click. If the second image is a coronal image and the user clicks on the axial image, the interface will jump to the coronal slice that is equal to the y coordinate of the mouse click. This synchronisation

works with every image of the same dataset. This way if the user flags a polyp on an image in one view, the interface displays the image for the relevant slice in the second view and a circle is drawn around the coordinates of the polyp that has been reported on the second image.

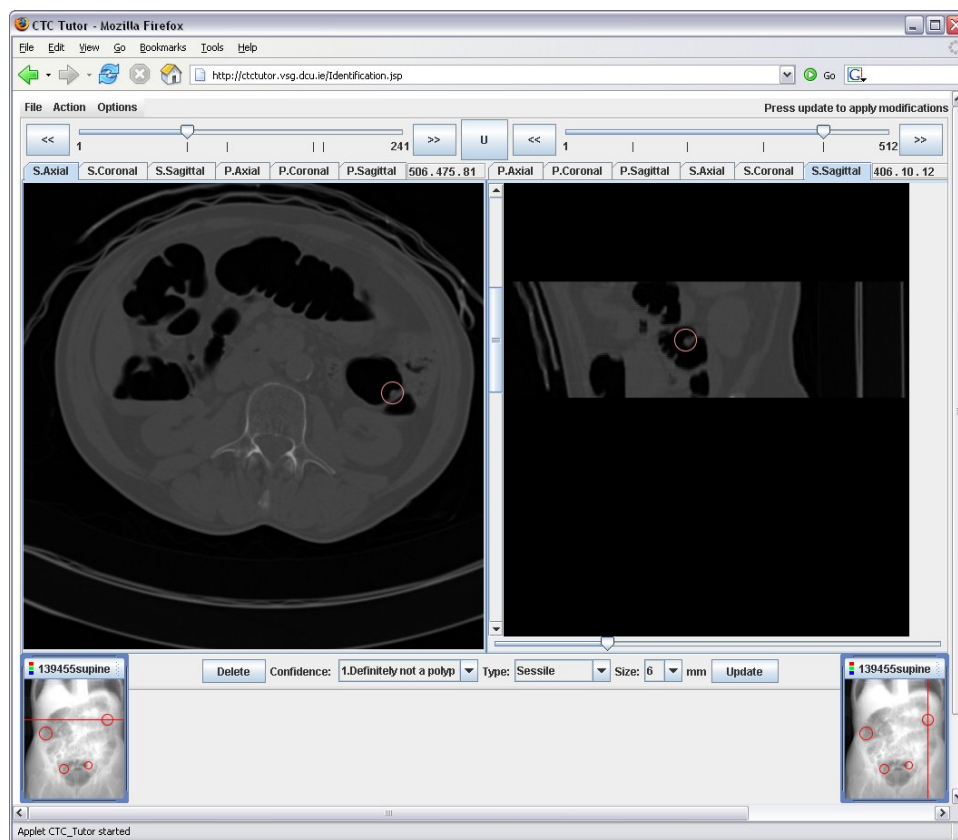


Figure 4-43. Image synchronisation. A polyp flagged on a supine-axial image is automatically reported on a supine-sagittal image and the system focuses on the relevant region of interest

If the second image is a reformatted image such as a sagittal or a coronal view, the interface automatically focuses on the relevant region of interest.

4.2.3.12. Measurement

A measurement tool has been implemented in the Applet interface (see figure 4-44). This feature is accessible from the “Action” menu. The user clicks on the picture to specify the first point and click once again to specify the end point. Therefore, we calculate the differences between these points on the x and on the y axes. Then, we convert these pixel differences into millimetres. To do so, we multiply the difference on x by the voxelWidth and the difference on y by the voxelHeight (voxelWidth and voxelHeight are written on the header of the dataset). At last, we simply deduce the length between the two points using the Euclidean distance (see equation 4.2.8).

$$length = \sqrt{[(endX - startX) \times voxelWidth]^2 + [(endY - startY) \times voxelHeight]^2} \quad (4.2.8)$$

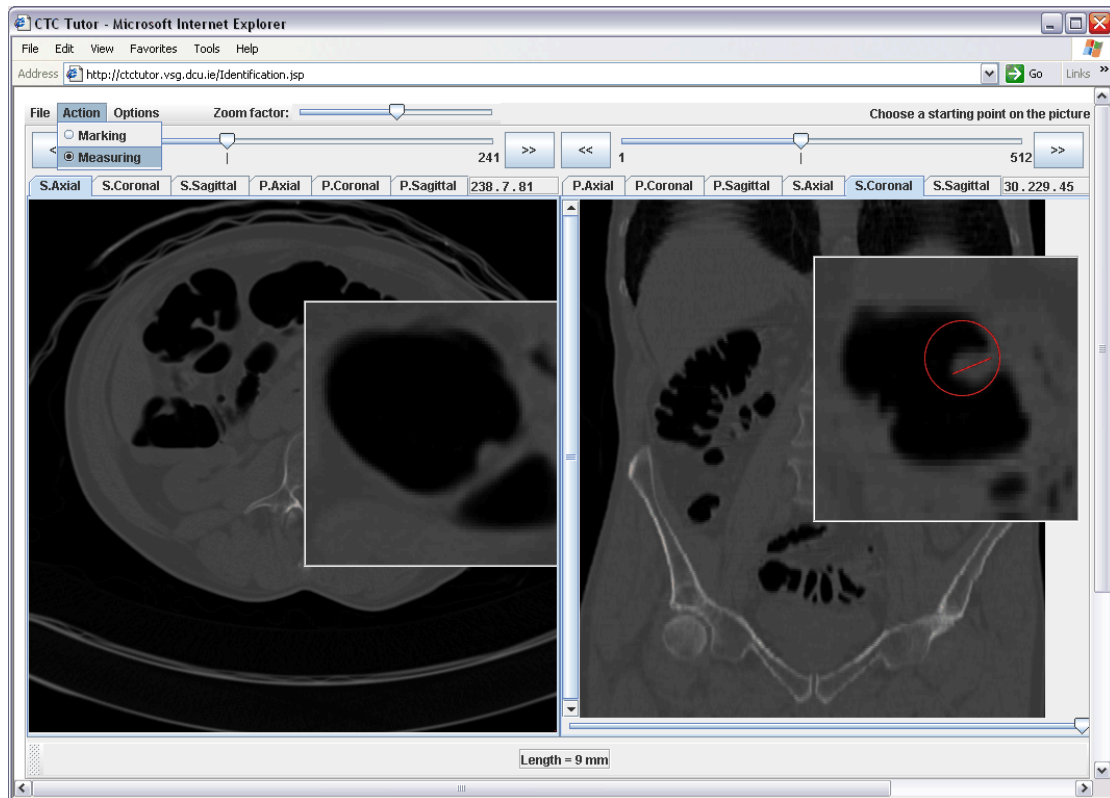


Figure 4-44. Zoom window and measurement tool. The length between two mouse clicks is indicated at the bottom of the interface.

4.2.3.13. Zoom Window

The Applet also offers the possibility to display a zoom window. This window works as a magnifying glass which can be moved with the mouse over the picture. To generate this view, we firstly crop the original bufferedImage thanks to the `getSubimage()` method:

```
BufferedImage buff = buffImage.getSubimage (x, y, zoomWidth
/ zoomFactor, zoomHeight /zoomFactor);
```

After that the cropped image is enlarged using the method `getScaledInstance()`:

```
Image im = buff.getScaledInstance(zoomWidth, zoomHeight,
Image.SCALE_SMOOTH);
```

At last, the image is drawn to the `JLabel` using the class `Graphics2D` and the `setIcon()` method (see figure 4-44).

4.2.3.14. 3-D Visualisation

3-D visualisation is also supported and is intended to be used for “problem solving” i.e. differentiate between polyps and folds. The algorithm used to generate 3-D images is based on the work of Zhanlin (2005) and is using a volume rendering technique called “ray-casting” (Levoy 1988). Ray-casting has two advantages:

- Ray-casting offers a high processing speed which ensures real-time volume rendering.
- With regards to the accessibility, ray-casting is a very interesting technique as it can be implemented by using standard Java classes.

As explained earlier in this chapter, a CT dataset is a stack of 2-D axial images. The combination of these 2-D images creates a volume that can be rendered using a 3-D algorithm. Ray-casting is an image order algorithm intended to render 3-D scenes to 2-D screens. The approach of ray-casting can be described as drawing a 2-D representation of a volume by calculating the distances between the eye of the observer and the voxels of the volume (see figure 4-45). When explaining the operation of ray-casting, these distances are usually referred to as “rays”.

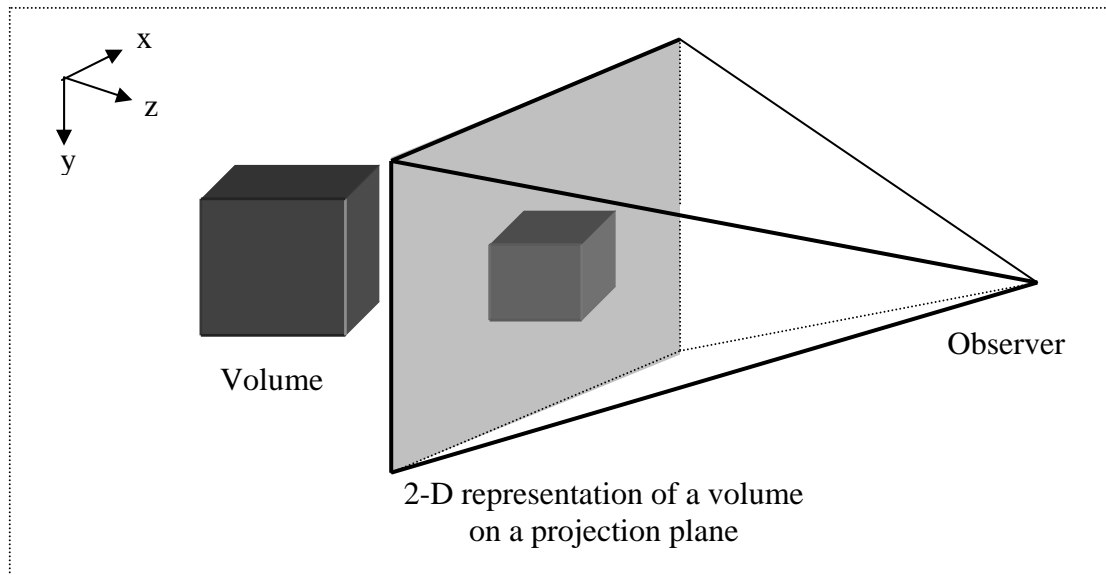


Figure 4-45. Observer watching the projection of a volume on a projection plane.

A 3-D subsection of the dataset is stored in a 3-D array. A loop is used to cast rays from left to right. As it was said earlier, rays go from the observer to the volume. When a ray hits a voxel of the volume, this voxel is reported on the projection plane (see figure 4-46).

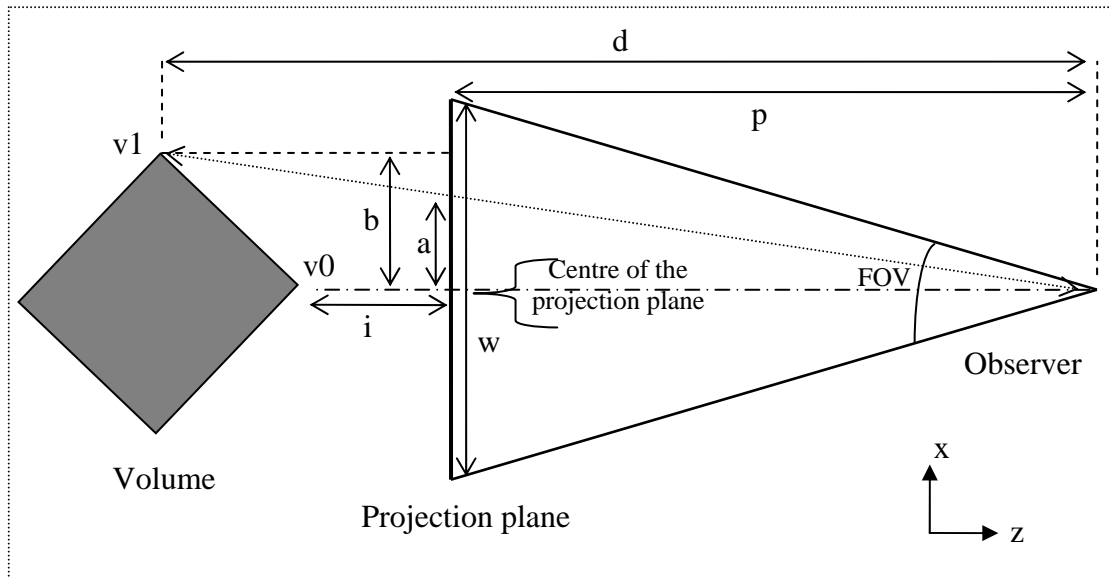


Figure 4-46. Projection of a cube on a projection plane (viewed from above). Ray-casting is used to represent a volume on a 2-D image using distances between the observer and the volume's voxels.

The figure 4-46 illustrates the projection of the corner of a volume on the width of a projection plane. On this figure, the distance “b” is the distance between the centre of the projection plane and the corner of the volume. The distance “a” is the projected distance of “b”. The x coordinate of the voxel's projection is equal to $\frac{w}{2} + a$. The distance “p” is calculated in relation with the width of the projection plane; “w” and

the angle called the field of view (FOV) is calculated as follows:
$$p = \frac{\frac{w}{2}}{\tan\left(\frac{FOV}{2}\right)}$$

Experience shows that 60 degree is a suitable value for the FOV. The width of the projection plane can be selected by the user. Therefore the distance p is calculated and stored in a variable. The user also selects “i”; the distance between the projection plane and the volume is also selected by the user. By observing figure 4-46 it is

possible to write: $\frac{a}{p} = \frac{b}{d}$ and $d = i + p$. Each voxel of the volume is reported on the

width of the projection plane using the following equation: $x = \frac{w}{2} + \frac{b \times p}{d}$. The

distance “b” is given by the position of the voxels v0 and v1 inside the volume.

The same operation is done on the height of the projection plane to find the y coordinate of each voxel's projection plane.

The projection plane is stored in a single dimension array called “pixels[]” and this image is created using the class *MemoryImageSource*:

```
MemoryImageSource source = new MemoryImageSource(width, height,  
pixels, 0, width);  
Image image = Toolkit.getDefaultToolkit().createImage(source);
```

A 3-D subsection of a CT dataset is equivalent to a solid cube. Each voxel of this cube has a density value. Therefore, in order to see the inner surface of the colon, voxels referring to air or gas must be removed from the 3-D representation (see figure 4-47).

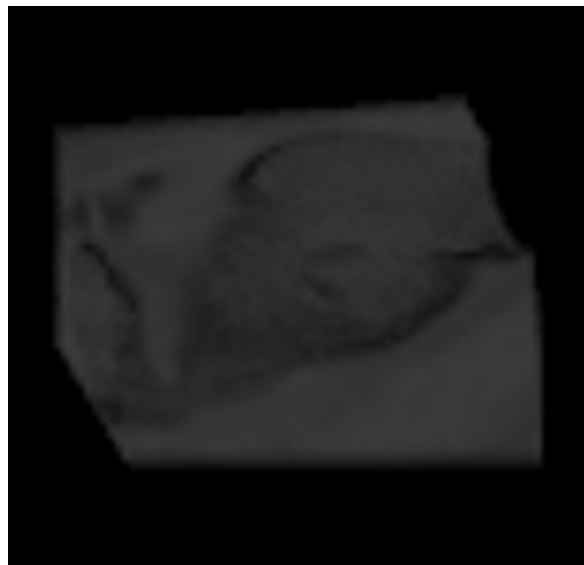


Figure 4-47. A 3-D rendering of the inner surface of the colon. This image has been generated with a threshold equal to -200 HU. A polyp is visible in the middle of this image.

This is achieved by exclusively taking into consideration, voxels that have a density value above a specific threshold. Subsequently, the threshold can be selected by the user to display information related to a specific density. For instance, if the threshold is set to 350 HU the resulting 3-D image represents a section of the skeleton of the patient (see figure 4-48).

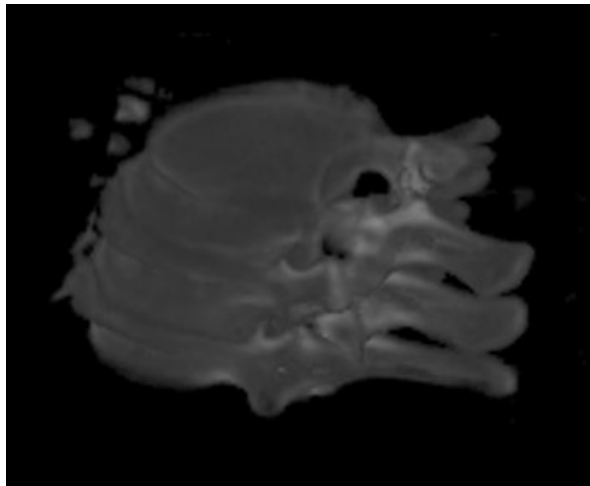


Figure 4-48. 3-D rendering of a dataset subsection with a threshold superior to 350 HU. With this threshold value, the 3-D algorithm offers a view of the skeleton e.g. a section of the spine.

To run the 3-D visualisation, the user uses the mouse to draw a 2-D area on one of the 2-D views e.g. axial, coronal or sagittal and the 3-D view will be generated using several slices on either side of the original 2-D image (see figure 4-49).

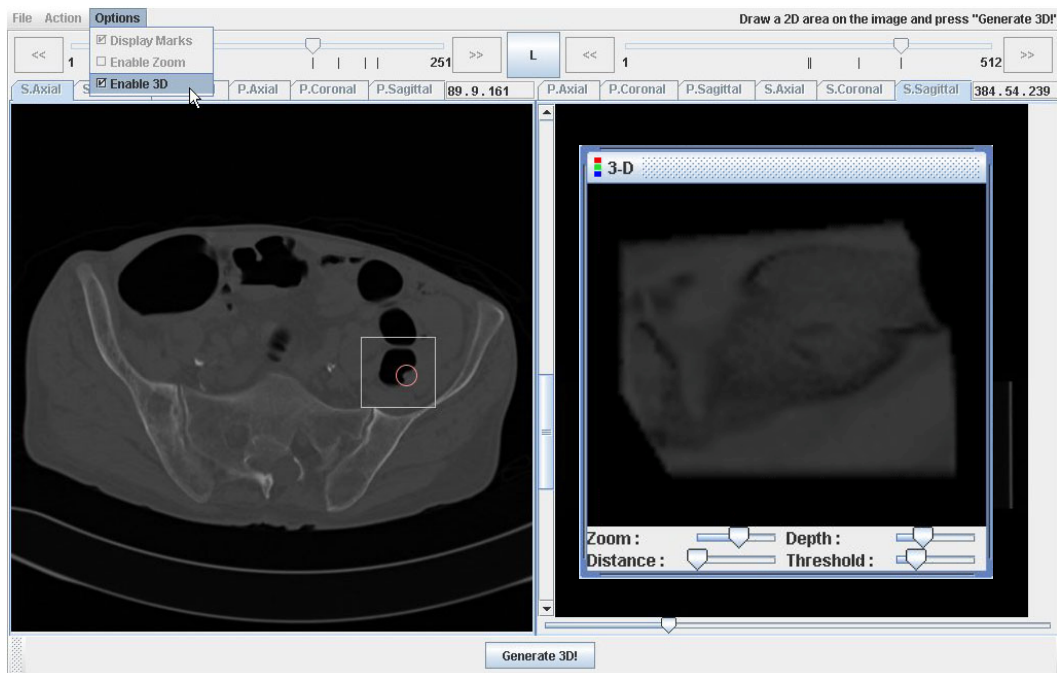


Figure 4-49. The 3-D interface. After the user has specified a region of interest, a volume is displayed in a floating window.

The system also allows our end-users to turn the zoom window into a 3-D view. In this case, the zoomed area is used as the region of interest. Again, the 3-D view is built using a few slices on either side of the zoomed axial image as illustrated in figure 4-50.

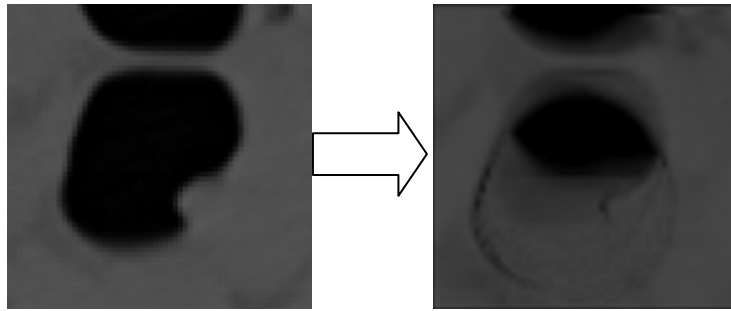


Figure 4-50. Generate a 3-D view from a zoomed area.

The 3-D view fits into a cube and it is possible for the user to drag the mouse over this cube to rotate in the x and y directions. Several parameters can also be adjusted in real time by dragging individual sliders. These parameters include depth, distance and opacity. Depth is the number of 2-D slices used to create the 3-D view. Distance can be described as the distance between user's eyes and the 3-D representation. Finally, the opacity is a parameter based on density values and can be used to highlight information associated with the bones or lungs.

4.2.3.15. Gold standard Evaluation

Upon completion of their work, the trainee can run an automatic evaluation based on gold standard information. This gold standard information can be gathered from specialists using our system. For the user who is registered as a specialist, the interface will provide more choice to define a polyp's type. Where trainees can only choose between "sessile" and "pedunculated" polyps, specialists can choose between "sessile", "pedunculated", "ileocecal valve", "stool tagging", "fold" etc. It is therefore possible for a specialist to highlight potential false positives. For instance, if a trainee marks polyps as sessile while it is the ileocecal valve, the evaluation panel will mark this polyp as a special false-positive and will display the correct type. Consequently, the evaluation process is intended to determine four possible types of polyp:

1. true-positive: polyps found by trainee which match with gold standard.
2. false-positive: polyps found by the trainee which don't match with gold standard.
3. Special false-positive: polyps found by the trainee that have been identified by the specialist as a potential false-positive.
4. False-negative: gold standard polyps which weren't found by the trainee.

In order to distinguish true-positive from false-positive polyps, the system compares each polyp's coordinates to the gold standard using the Euclidean distance:

if

$$(\Delta \max \geq \sqrt{(X_{gold} - X_{trainee})^2 + (Y_{gold} - Y_{trainee})^2 + (Z_{gold} - Z_{trainee})^2})$$

polyp = True-Positive;

else

polyp = False-Negative;

On the interface a colour code is used to highlight the different polyp categories. For instance, a red circle drawn on the 2-D axial image identifies a false-positive polyp whereas a green circle identifies a true-positive polyp. Coloured ticks are also drawn on the navigation slider and on the simulated scout x-ray image to show the location of these specific polyps (see figure 4-51).

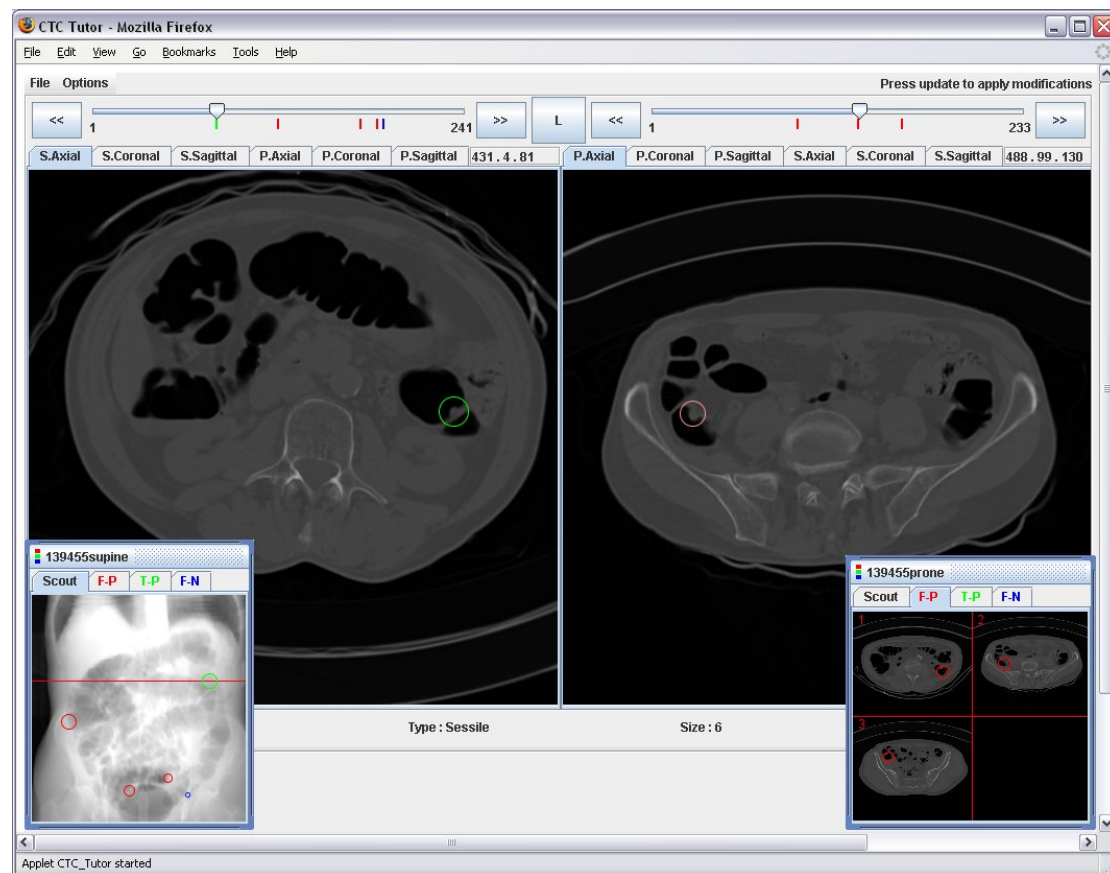


Figure 4-51. The evaluation panel. On this panel, a colour code is used on the navigation bar, on the 2-D images and on the scout x-ray to indicate the true-positive (green), false-negative (blue) and false-positive (red) polyps.

Thanks to this colour code, the user can instantly visualise his results by looking at the navigation sliders or the simulated scout x-ray images. With regard to the processing of the different polyp categories, we took advantage of Object Oriented Programming

(OOP) concepts. In practical terms, a Java class with private data and methods has been implemented in order to create a polyp category object (See figure 4-52). The private data are arrays used to store the 3 coordinates, type and size of polyps. The methods are used to set, get, modify and delete a polyp of a specific category. This object is used to maximise the flexibility of the user interface as it is possible to retrieve and highlight polyps of a chosen category.

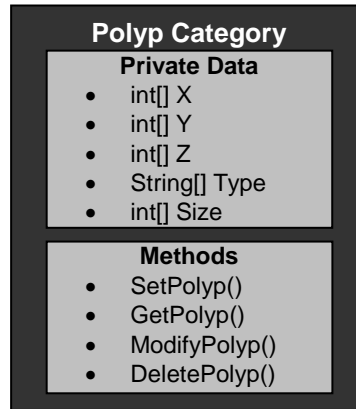


Figure 4-52. Polyp Category Object

To create a new polyp category, the class PolypCategory is instantiated: (*PolypCategory truePositivePolyps=new PolypCategory ()*). The polyp categories used in this system are:

1. Polyps flagged by a trainee on a supine dataset.
 2. Polyps flagged by a trainee on a prone dataset.
 3. Polyps flagged by a specialist on a supine dataset.
 4. Polyps flagged by a specialist on a prone dataset.
 5. True-positive polyps on supine dataset.
 6. True-positive polyps on prone dataset.
 7. False-positive polyps on supine dataset.
 8. False-positive polyps on prone dataset.
 9. False-negative polyps on supine dataset.
 10. False-negative polyps on prone dataset.
 11. Polyps from each category on supine dataset.
 12. Polyps from each category on prone dataset.
- } Polyps compared to gold standard polyps
 } Gold standard polyps
 } Polyps flagged by the trainee that match with the gold standard
 } Polyps flagged by the trainee that don't match with the gold standard
 } Polyps flagged by specialists that weren't flagged by trainees

4.2.3.16. Display all pictures of a specific polyp category

Regarding the result panel, another request of our medical colleagues is to display every true-positive, false-positive or false-negative at once on the same screen.

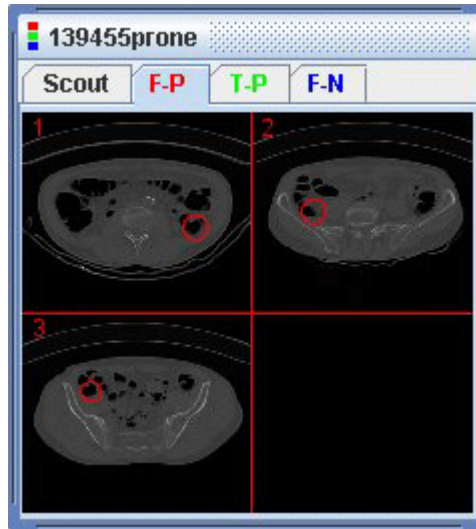


Figure 4-53. Floating window displaying polyp thumbnails of a specific category. After the running the automatic evaluation, the user can click on a tab at the top of the scout x-ray window to display thumbnails of a specific category.

When the user runs the automatic evaluation the system displays 3 new tabs on the scout x-ray window as illustrated in figure 4-53. These tabs correspond to the different polyp categories and are identified on the interface by their abbreviation e.g.: “F-P” for false-positive. The number of thumbnails displayed in a tab is equal the number of polyps of the corresponding category. For example if there are 2 false-positive polyps on the same axial image, two thumbnails will be displayed, one for each of these polyps. When the user clicks on a thumbnail, the system displays the associated image along with the information about the specific polyp.

To display the thumbnails of a polyp category, the method `getNbPolyp()` of the class `PolypCategory` is used to get the total number of polyps (`totalPolyp`) of a specific category. Then, the number of rows and columns is calculated as illustrated in the flowchart, figure 4-54.

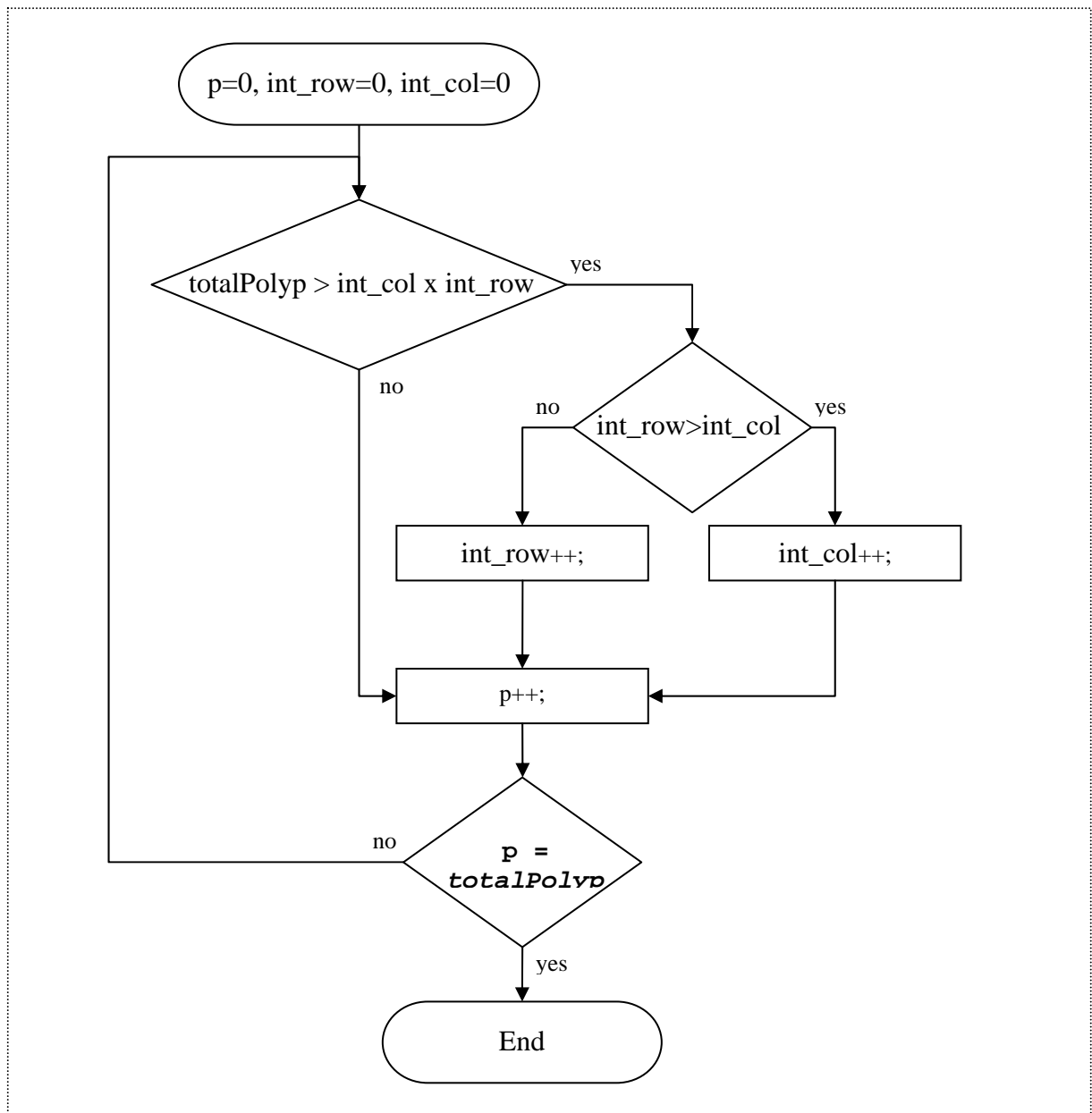


Figure 4-54. Calculating the number of rows and columns in the scout x-ray window.

Finally, each thumbnail is generated and displayed in the right cell. To find out the appropriate translation on the X axis of each thumbnail, the modulo¹ (symbolized as “%”) of the thumbnail’s number “t” by the number of columns is calculated and then multiplied by the width of a cell. Finally, the translation on the X axis is the absolute value of the result: $translationX = |width \times (t \% int_col)|$

1) The modulo is the remainder of a division.

To find out the appropriate translation on the Y axis, the thumbnail number is multiplied by the number of rows and the result is divided by the total number of thumbnails. Following this, the floor¹ of the result is multiplied by the height of a cell:

$$translationY = height \times \text{floor}\left(\frac{t \times int_row}{totalNumberThumb}\right)$$

When the user clicks on a specific thumbnail, the system compares the coordinate of the mouse click to each thumbnail's position. As illustrated in the following flowchart (figure 4-55), for each thumbnail "t", Four values; Xmin, Xmax, Ymin and Ymax are calculated using the modulo and the floor method. If the coordinates of the mouse click are included inside these four values, the selected thumbnail is found and the interface displays the associated image.

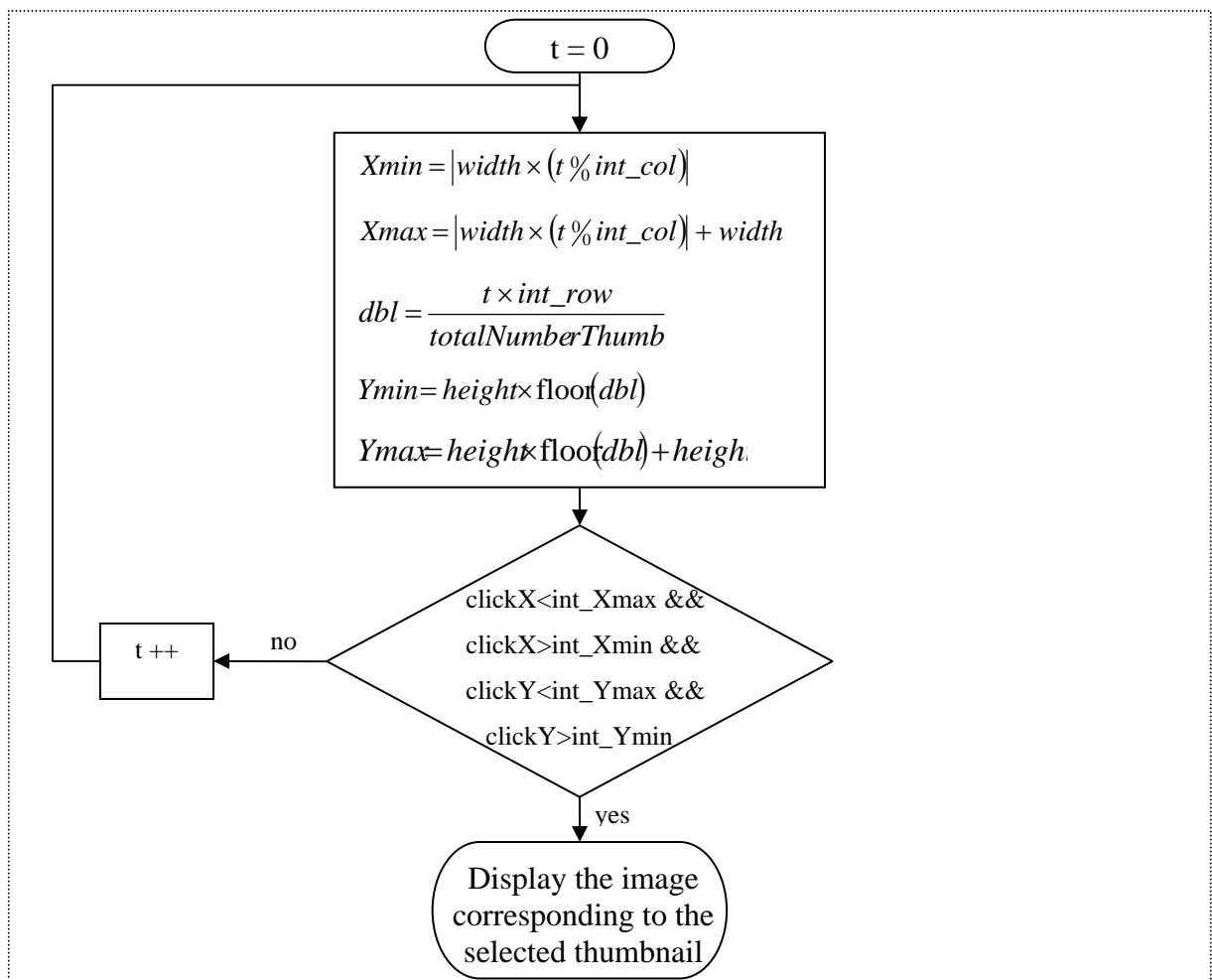


Figure 4-55. Find the selected thumbnail from the coordinates of the mouse click.

1) The "floor" of a number is the largest integer less than or equal to the number. Every integer is its own floor.

4.3 Multi-user architecture

4.3.1. Introduction

The access to the training system needs to be restricted to a list of authorised users. Gluecker (Gluecker et al. 2002) states that “before applying [CTC], the radiologists involved require a controlled learning period”. The main purpose of this multi-user system is to identify the user in order to store the training results and provide monitoring of the training. Several Java server pages (JSP) are used to make this multi-user system and help our end-users to manage their accounts. The following paragraphs give details of this system that has been developed in addition to the training system.

4.3.2. Login panel

To access the system, the user simply enters the appropriate URL in the address bar of his/her selected browser. The home page (see figure 4-56) is loaded and the user can run the identification process or access the registration page. If the user enters his/her login and password and presses the submit button “Enter!”, the Java Bean associated with this JSP compares the user entries with the logins and passwords stored in a text file on the server. If the user entries match the data stored on the server, access is granted to the user. If a wrong password is entered, the server saves the login and password that have been submitted, along with the date and the IP of the user. This data is stored in a text file that can easily be examined by the system administrator.

If the access is granted, the system reads the profile that has been assigned to this user. At this stage, there are 3 different scenarios in conjunction with the possible profiles:

- a. A trainee is able to train for colorectal cancer screening and run an evaluation of his or her work based on gold standard information.
- b. A specialist is able to flag polyps using the same interface as a trainee and can save his or her study as gold standard information.
- c. An administrator is able to create and delete user accounts and access to the trainee’s results.

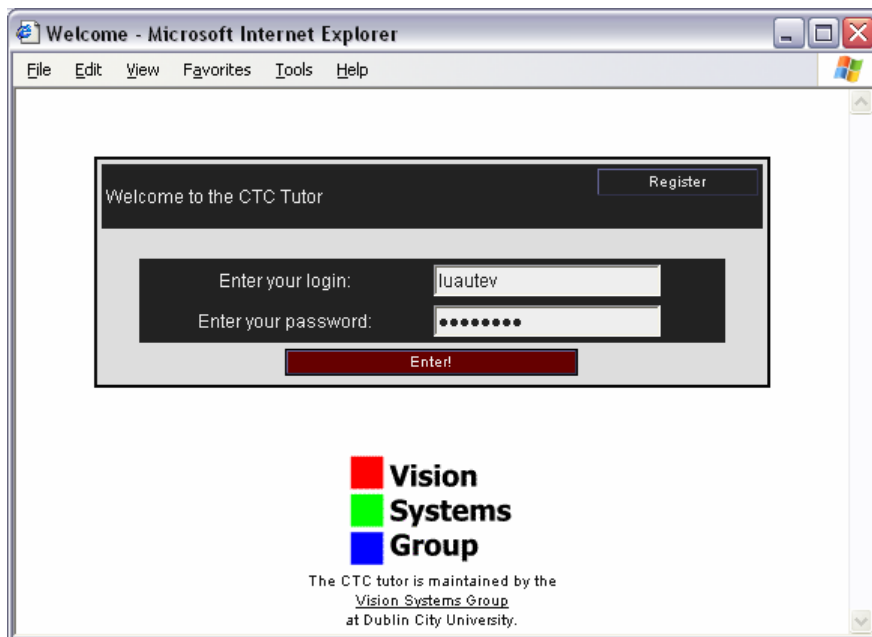


Figure 4-56. Home page of the system.

4.3.3. User registration

A registration panel has been implemented on the system (see figure 4-57).

Figure 4-57. The registration panel. A new user can register and enter the relevant information on this page.

This registration page is made up of a simple HTML form. This form is used to collect the basic information that we need to identify a user like the name, surname,

organisation and email. The user is also required to enter additional information related to his/her possible previous experience with abdominal CT. This additional information is important to later establish a relevant learning curve and measure the education ability of the training system.

When the user presses the submit button, information is stored in a text file on the server and an email is sent to the administrator. Methods from the libraries `javax.mail.*` and `javax.mail.internet.*` are implemented to the Java Bean. The following code is used to send an email and a specific subject to 2 different recipients:

```
Properties props = System.getProperties();  
props.put("mail.smtp.host", "pine.eeng.dcu.ie");  
Session session = Session.getDefaultInstance(props, null);  
MimeMessage msg = new MimeMessage(session);  
msg.setFrom(new InternetAddress("ctctutor@eeng.dcu.ie"));  
msg.setRecipients(MimeMessage.RecipientType.TO,InternetAddress  
ss.parse("ctctutor@eeng.dcu.ie", false));  
msg.setSubject("Subject of the email");  
msg.setText("text to send");  
msg.setHeader("X-Mailer", "LOTONTechEmail");  
msg.setSentDate(new Date());  
Transport.send(msg);
```

4.3.4. User possibilities

The flowchart figure 4-58 describes the different functionalities offered to the user.

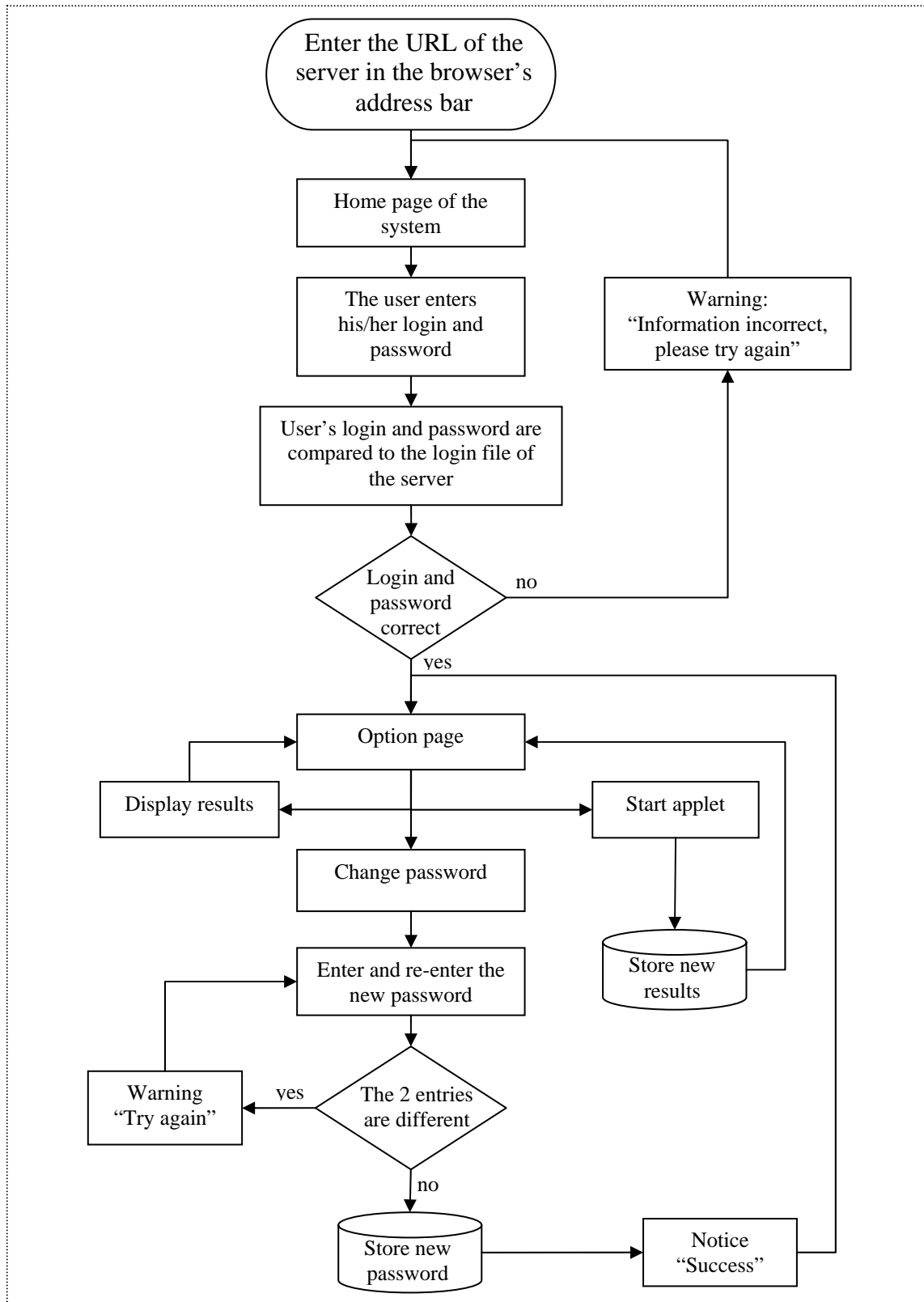


Figure 4-58. Multi-user system. This flowchart gives an overview of the operation of the system that surrounds the applet.

After running the identification process, the user reaches an option page (see figure 4-59).



Figure 4-59. Option page

From here the user can start the training Applet, display his/her previous results or change his/her password. Also a text box has been added to collect some feedback from our end-users.

4.3.5. Result page

A result page (see figure 4-60) is implemented to display the trainee's result as the number of true-positive, false-positive, false-negative and the reader time. This result page is accessible from the option page and from the training Applet. Precisely, once the trainee has run the automatic evaluation, the result page is accessible from the File menu of the Applet. The advantage of having this result page displayed via an HTML page instead of an Applet is the possibility for our end-users to select copy or print their results. Also, our end-users can access this result page without having to load the training Applet.

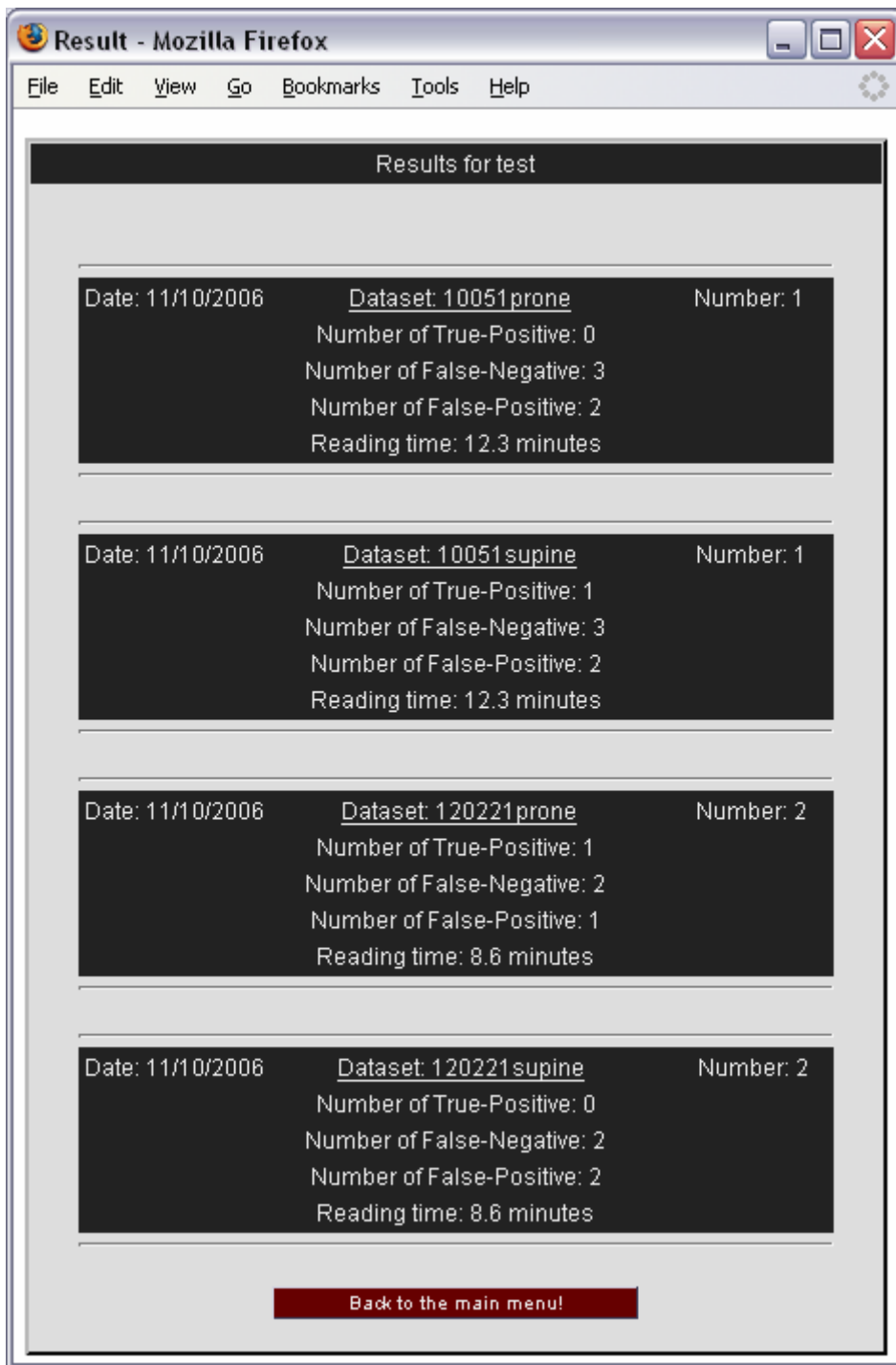


Figure 4-60. Result page. This page shows the number of true-positive, false-negative, false-positive and the reading time for each dataset previously studied.

4.3.6. Administration panel

If the user entries match with the administrator login and password, the JSP will generate the HTML page related to the administration panel (see figure 4-61).

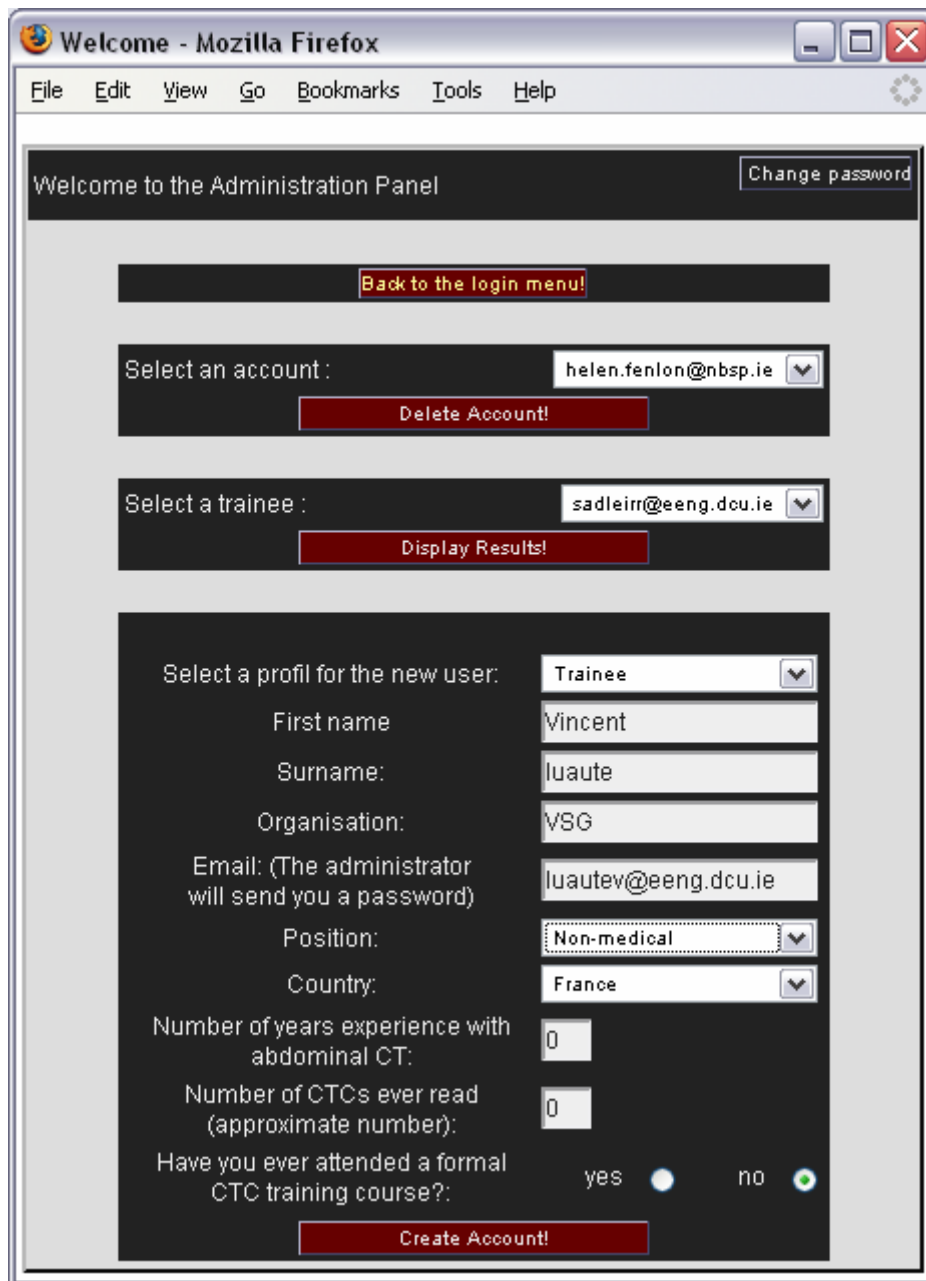


Figure 4-61. The administration panel. After a new user register on the system, an email is sent to the administrator. The administrator can then use this page to create the new account, delete an existing account or display the results of a trainee.

This panel allows an administrator to create a new user account. A first combo box is used to select the new user's profile e.g. trainee and specialist. On this panel it is also possible to delete an existing account and display the results of a specific trainee.

4.3.7. Conclusion for the implementation using Servlets and Applets

The main advantage of the second implementation is to transfer the relevant CT dataset on the client side in order to offer a short response time to the interface. This implementation provides new features such as the measurement tool and navigation using a simulated scout x-ray. Also, the development of 3-D visualisation and features involving mouse scrolling couldn't be achieved on the initial implementation that is exclusively made up of Servlets. The new possibilities offered by the Applet make the interface of the system more user-friendly.

Chapter 5. Testing & Results

5.1 Testing

5.1.1. Introduction

To be able to run the Applet, our end-users must use a recent browser that is compatible with signed Applets. Here are some examples of compatible browsers:

- Netscape Navigator, version 4.0 and above.
- Microsoft Internet Explorer on Windows, version 4.0 and above.
- Firefox 1.0 and above.

The server keeps a log as a text file of users connecting and downloading datasets. In this chapter, the transfer time of CT datasets from the server to the end-users has been extracted from this log. The server of this system is located in a lab of the Vision Systems Group in Dublin City University.

5.1.2. Test of the automatic evaluation

The primary function of the system is the automatic evaluation process, which informs the trainee about the number of TP (true-positive), FN (false-negative) and FP (false-positive) on his/her study. The system saves the results in a text file on the server machine in order to monitor the progress of the trainee. As described in chapter 4, the evaluation consists of comparing each polyp's coordinates to the gold standard using the Euclidean distance:

$$\Delta \max \geq \sqrt{(X_{gold} - X_{trainee})^2 + (Y_{gold} - Y_{trainee})^2 + (Z_{gold} - Z_{trainee})^2}$$

During our tests, the value of $\Delta \max$ was 6mm which corresponds to the length of a small polyp.

Gold standard polyp coordinates are stored in a text file on the server. When running the automatic evaluation, the server stores all polyp coordinates found by the trainee in another text file. Following a first test on 8 datasets, it is possible, to display, side by side, the coordinates of the polyps from the gold standard, the coordinates of the polyps found by the trainee and the result that has been given to the trainee via the evaluation panel, for each dataset.

Dataset	Gold standard			Trainee			Displayed result		
	Xg	Yg	Zg	Xt	Yt	Zt	TP	FN	FP
10051prone	No polyp			No polyp			0	0	0
10051supine	No polyp			No polyp			0	0	0
17384prone				114	238	138	4	0	2
	100	284	164	101	284	165			
				230	282	185			
	114	351	187	114	355	186			
	148	255	195	149	256	194			
	254	286	196	255	290	195			
17384supine				383	299	119	4	0	2
	384	263	161	385	263	162			
	255	244	178	256	245	176			
				323	314	185			
	258	257	196	259	257	194			
	325	328	204	324	327	204			
120221prone	171	277	229	172	272	229	1	0	0
120221supine	246	287	200				1	2	0
	265	381	199	263	338	205			
	268	336	205						
139455prone	none	none	none	117	211	82	0	0	1
139455supine	407	231	81	409	235	81	1	1	0
	355	144	188						

Table 5-1. Result displayed on the evaluation panel after comparing coordinates of polyps found by a trainee to the gold standard.

Using the data presented in Table 5-1, we can see that tests show that the results given by the system after running the automatic evaluation are correct. Therefore, the training system is reliable.

Dataset 10051prone:

- The gold standard indicates that there is no polyp.
- The trainee did not flag any polyp.
- Consequently the result 0 TP, 0 FN and 0 FP is correct.

Dataset 17384prone:

- The gold standard indicates that there are 4 polyps.
- The trainee found 6 polyps and 4 of them matched with the gold standard.
- Therefore, the two remaining polyps are false-positives and the result is correct.

Dataset 120221supine:

- The gold standard indicates 3 polyps.
- The trainee found one polyp and this polyp matches with the gold standard.
- Therefore, the 2 remaining polyps are false-negatives, the result is correct.

5.1.3. Response time of the interface

Response time of the user interface has been recorded using a PC with a Pentium 4, 2.8GHz and 512MB of RAM.

Axial	Coronal		Sagittal	
	Full image	Region of interest (60)	Full image	Region of interest (60)
0.06	1.4	0.4	4.7	0.5

Table 5-2. Response time (in seconds) for the different image.

5.1.3.1. Response time of axial images

As seen in table 5-2, the average time spent to generate an axial image is 0.06s. This response time is acceptable for our end-users. Navigation through axial images using mouse scrolling or dragging the navigation slider can be considered as real time operation.

Another relevant test of the interface consists of timing an iteration of lumen tracking. Using axial images, it takes approximately 45 seconds to visually navigate through the entire colon.

5.1.3.2. Response time of reformatted images

By default the Applet generates a reformatted image using 60 axial images. In other words, the default height of the region of interest is equal to 60 pixels. Thanks to the synchronisation between axial and reformatted images (as described in chapter 4), the region of interest automatically focuses on the relevant section of the reformatted image. Therefore the default height of the region of interest should be sufficient. As illustrated in table 5-2, time spent to generate a reformatted image is about 0.4s.

5.1.4. Transfer time of CT datasets between server and client

Transfer time of CT datasets from the server to the end-user has been optimised by the use of compression (as described in chapter 4). This compression reduces the size of CT datasets by 50%. The interface of the system displays prone and supine images of the same patient side by side. Therefore using compression, the transfer time of these 2 datasets is equivalent to the transfer time of an uncompressed dataset.

The transfer time of datasets using two different server programs has been measured over a local network. The same server computer and client computer were used for both tests. The size of the dataset used for this test is 120MB.

Server type	Transfer time
Server on port 8080	18.8min
Server on standard HTML port (port 80)	20.6min

Table 5-3. Transfer time of CT datasets.

Table 5-3 shows that the server on the standard HTML port which uses the Java class *URLConnection* is slightly more efficient than the other server program which uses a socket.

The log file in the server has recorded transfer time of 11 CT datasets from the server to the computer of end-users outside the university. These datasets have been transferred using the server on standard HTML port. Most of our end-users are accessing the system from Dublin.

Dataset name	Dataset size	Number of transfer	Minimum transfer time	Maximum transfer time	Average transfer time
10051supine	124MB	4	8.24min	37.55min	18.8min
10051prone	130MB	3	8.22min	41.18min	20.6min
120221supine	113MB	1	7.58min	7.58min	7.58min
120221prone	128MB	1	9.44min	9.44min	9.44min
166669supine	150MB	1	13.10min	13.10min	13.10min
166669prone	156MB	1	14.54min	14.54min	14.54min

Table 5-4. Transfer time of CT datasets using the server on standard HTML port.

Using the data in table 5-4, the average transfer time per dataset is equal to 16.5 min. As requested by our medical colleagues, the interface of the system displays a supine CT dataset and a prone CT dataset side by side. Table 5-4 shows that the average transfer time of two datasets from the server to the client is approximately equal to half an hour.

Over a local network, transfer time of a 150 MB dataset from the server to the user's hard disk is approximately 1 minute. In other words, users using the system inside the same building as the server can start working with the training system after about 2 minutes of data transfer.

5.2 Discussion

If our end-user wishes to perform colorectal cancer screening using reformatted images exclusively, the region of interest should be extended to the entire coronal or sagittal image. As shown in table 5-2 the average time spent to generate an entire coronal image is 1.4s and an entire sagittal image is 4.7s. This response time is not tolerable to our end-users. Therefore, this remote training system is dedicated to the colorectal cancer screening using axial images and the use of reformatted image is limited to problem solving purpose.

This system uses different data transfer techniques between the server and the client over a network. For instance, data streaming guarantees the maximum speed for the transfer of a CTC dataset. Using the dedicated compression algorithm, the transfer time of a CT dataset from the server to the user's hard disk is divided by 2.

Unfortunately, the transfer time of datasets from the server to a distant computer is considerable. Before broadband connections become faster, the server and the end-user's computer should be located inside the same building e.g. hospital or university.

This system has been developed to handle the training of radiologist and gastroenterologist with a large number of datasets. In practical terms, the limited number of dataset that can be examined is set by the capacity of the server's hard disk.

Concerning the accessibility of this remote training system, two server programs using a different port have been implemented. The experience has demonstrated that the firewall of the server's local network sometimes blocks the server program running on a specific port while it allows the server program running on the standard HTML port. For example, when the server has been installed in the lab of the vision systems group, it was found that the firewall of the university blocks the ports other than the standard HTML port (port 80). Fortunately, the IT department of the university has accepted to open the specific port (port 8080) that is used by the second server program. Therefore, both servers are available to our end-users.

Besides, some firewalls of the client's network i.e. the firewall of the Mater hospital in Dublin, blocks the Applets which transfer data through the port 80 while computers inside the Mater hospital can download datasets via the server running on the specific port (port 8080).

Our study has demonstrated that a client-server architecture using a combination of Servlets, JSPs and Applets is a good solution to develop a remote access system for CTC training. This combination provides efficient data transfer performance. On the client side, an Applet provides a user friendly interface and can be loaded on every Java enabled browser. This approach offers great accessibility and avoids the need for any custom software installation on the client side.

Chapter 6. Conclusion

6.1 Initial implementation

This thesis describes two different implementations of the CTC training system. The initial implementation exclusively uses server side programs and allows a trainee radiologist to flag colorectal cancer polyps and run an automatic evaluation based on gold standard information. This approach provides the tools that are required to facilitate the identification of polyps. These features include windowing and zooming. Navigation through the 2-D axial images of a dataset can be performed by using forward/backward buttons, clicking on a navigation bar or choosing a slice number via a drop down menu. Although this research has yielded an operational system that meets the main requirements of our medical collaborators, speed and usability are major issues that are associated with this implementation.

6.1.1. Speed problem

CT images that are requested by the user are sent as he/she studies a CT dataset. Each axial image has to be extracted from the CT dataset and then converted into JPEG image before being sent to the client. This process takes a considerable time and the interface has a very slow response time (>5s). The evaluation of the system with our medical colleagues has revealed that slow response time renders this implementation unusable. However, this initial research has proved invaluable in developing a better approach. Discussion about the initial implementation with our medical colleagues revealed that it is more convenient for our end-users to wait for a whole CT dataset to be transferred from the server to the client instead of waiting for each image to be transferred as the user requests it.

6.1.2. Usability problem

The interface to the initial implementation is made up of HTML pages that are generated by server side programs (Java Servlets). The usability issue of this implementation is due to the limitation of HTML. For instance, our medical collaborators pointed out the need for mouse scrolling that is usually implemented on modern analysis systems.

6.2 Revised implementation

In response to the observations of our medical collaborators, the revised implementation consists of loading a whole CT dataset when the user starts the training system. To reduce this transfer time, a compression technique has been developed and is applied to CT datasets prior to transmission. As loss-less compression is used, received 2-D axial images remain the same as the original. Preserving the original density range and the original resolution ensures that the user can get full advantage of features such as windowing, zooming and 3-D visualisation. Thanks to the fast decompression process, the viewer Applet gives users registered as specialists, the ability to start working on the initial images while the rest of the dataset is being transferred.

Two server programs using different approaches have been implemented. The training system lets the user choose the server program he/she wants to use before starting the Applet. If the firewall of the client blocks the access to the first server, the user is free to try the second server program. At present, radiologists from the Mater hospital are able to remotely use the training system from the Mater hospital in Dublin.

6.2.1. Better response time

The second implementation of the CTC training system was developed using a combination of server side and client side programs. When the CT dataset has been sent to the client side, the interface of the training system exclusively uses the central processing unit (CPU) of the client's computer. This means that the response time only depends on the speed of the user's computer. With a standard, modern computer for instance a PC with a Pentium 4, 2.8GHz and 512MB of RAM), the interface offers a real time navigation using axial CT images. With this approach, the server is limited to provide CT datasets and other files to the clients. Every operation on the CT dataset is performed by the client's computer. Therefore, the server saves some resources and can handle more clients.

6.2.2. Advanced features

In response to the issue of usability, a Java Applet has been chosen as the client program. An Applet can be loaded on a standard browser, provides an advanced user interface and is able to exchange data with server side programs. For instance, the mouse scrolling that was requested by our medical collaborators can be easily

implemented, to navigate through slices of a dataset. In addition, sliders and floating simulated scout x-ray images have been developed to easily jump to a desired slice and automatically focus on one of the previously flagged polyps. These new features make the interface more user-friendly compared to the initial implementation. As a result, we have developed a dedicated system to analyse 2-D axial images in conjunction with modern analysis tools.

3-D visualisation is also supported and is intended to be used for “problem solving”. A volume rendering technique called “ray-casting” has been implemented in the system. One of the benefits of using ray-casting is the high processing speed which ensures real-time volume rendering. Also this algorithm can be easily implemented in the Applet code. Therefore, ray-casting is fully compatible with the basic Java Virtual Machine (JVM) and ensures compatibilities with the end-user’s system.

For users registered as a specialist, the system can also be used as a tool to analyse and annotate CTC datasets. Specialists can mark CT datasets using the visualisation features provided by the Applet interface.

6.2.3. Evaluation strategy

Upon completion of their work, the trainee can run an automatic evaluation. This evaluation is based on gold standard information previously gathered from colonoscopy records. The evaluation panel displays the true-positive, false-negative and false-positive polyps using a colour code. Specialists can use the interface of the system to flag potential false-positive polyps in addition to the true-positive polyps. For example, the ileocecal valve can be misinterpreted by trainees as colorectal cancer (Summers et al. 2004). Therefore if a trainee flags it as a polyp, the evaluation panel will be able to inform him/her that it is in fact the ileocecal valve.

The multi-user interface of the system allows monitoring of training and is secured by the use of login and password. Following the automatic evaluation, the trainee can choose to study a dataset again but his/her results will not be overwritten. Therefore, the monitoring of the training will be reliable. Finally, the use of our system will certainly help to determine the learning curve associated with interpreting CTC studies.

6.3 Future work

This research is ongoing due to the changing needs of our medical collaborators and the constant evolution of the CTC technique. The main weakness of this system is the transfer time of CT datasets from the server to the client. Hopefully, CT dataset compression can be improved in order to reduce this transfer time. Also, the performance of the user's computer and the connection speed between the server and the client could be determined. The server could optimise the dataset transfer by selecting the most appropriate compression ratio.

The security of the multi-user system can be improved. For example, an SSL connection could be used to encrypt the user's password into a database in the server. Concerning the trainee's result, a more comprehensive interface could display the progression of the training. This feature could certainly increase the user-friendly aspect of this system and maybe encourage the end-users by providing improved feedback regarding their training.

Recent developments in CTC research that will need to be dealt with by any training programme include fecal tagging for improved segmentation and polyp visibility (Lefere et al. 2002, Pickhardt et al. 2003). Finally, computer aided detection (CAD) (Summers et al. 2004, Yoshida et al. 2005) is a tool more and more used by specialists to pre-examine CT datasets and find potential true-positive cancer polyps. This feature could be also included in this training system as an assistive tool.

In order to meet the needs of our ultimate end-users (trainee radiologists in the area of CTC) we have been in regular consultation with our medical collaborators from the Gastrointestinal Unit and Department of Radiology at the Mater Misericordiae Hospital in Dublin.

Upon completion of their study, results from our end-users are saved in a text file on the server machine. Those results include the number of true-positive, false-negative and false-positive as well as the reading time for each dataset studied. Therefore, after the study of 50 to 100 cases by a group of trainee radiologists, it would be possible to determine the learning curve associated with the relevant CTC tutor. It would be particularly interesting to calculate the average evolution of the number of true-

positive, false-negative and false-positive polyps in order to determine the evolution of the sensitivity and the specificity. Sensitivity can be calculated using the following equation:

$$sensitivity = \frac{\text{number of true - positives}}{\text{number of true - positives} + \text{number of false - negatives}} \quad (6.3.1)$$

Specificity can be calculated using the following equation:

$$specificity = \frac{\text{number of true - negatives}}{\text{number of true - negatives} + \text{number of false - positives}} \quad (6.3.2)$$

References

- (Aufort et al. 2005) Aufort S, Charra L, Lesnik A, Bruel JM, Taourel P (2005) "Multidetector CT of bowel obstruction: value of post-processing." EUROPEAN RADIOLOGY 15 (11): 2323-2329 NOV 2005
- (Bohne-Lang et al. 2005) Bohne-Lang A, Groch WD, Ranzinger (2005) "AISMIG - an interactive server-side molecule image generator" Nucleic Acids Research, 2005, 33 (Web Server issue), 705-709
- (Campo et al. 2004) Campo J, Comber H, Gavin A T. (2004) "All-Ireland Cancer Statistics 1998-2000" Northern Ireland Cancer Registry/ National Cancer Registry 2004.
- (Cotton et al. 2004) P.B. Cotton, V.L. Durkalski, B.C. Pineau et al. (2004) "Computer tomographic colonography (virtual colonoscopy): A multicenter comparison with standard colonoscopy for detection of colorectal neoplasia" The Journal of the American Medical Association 291(14)1713-1719.
- (Dachman et al. 2003) Dachman AH, Yoshida H (2003) "Virtual colonoscopy: past, present, and future" Radiologic clinics of north america 41 (2): 377-+ MAR 2003
- (Fenlon et al. 1999) H. Fenlon, D.P. Nunes, P.C. Schroy et al. (1999) "A comparison of virtual and conventional colonoscopy for the detection of colorectal polyps" New England Journal of Medicine 341(20):1496-1503.
- (Fidler et al. 2004) J.L. Fidler, J.G. Fletcher, C.D. Johnson et al. (2004) "Understanding interpretive errors in radiologists learning computed tomography colonography" Academic Radiology 11(7):750-756.

- (Fisichella et al. 2006) Fisichella V, Hellstrom M (2006) "Availability, indications, and technical performance of computed tomographic colonography: A national survey" ACTA RADIOLOGICA 47 (3): 231-237 APR 2006
- (Fletcher et al. 2000) Fletcher JG, Johnson CD, Welch TJ et al. (2000) "Optimization of CT technique: prospective trial in 180 patients" Radiology 216:704-711
- (Gluecker et al. 2002) Gluecker T, Meuwly JY, Pescatore P, Schnyder P, Delarive J, Jornod P, Meuli R, Dorta G. (2002) "Effect of investigator experience in CT colonography" Eur Radiol. 2002 Jun;12(6):1405-9. Epub 2002 Feb 9.
- (Hill et al. 1978) Hill MJ , Morson BC , Bussey HJR (1978). "Etiology of the adenoma-carcinoma sequence". Lancet i : 245 - 247 .
- (Huffman 1952) D. Huffman, "A method for the construction of minimum redundancy codes," Proceedings of the IRE, vol. 40, no. 9, pp. 1098–1101, Sep.1952.
- (Klein 2003) Mark Klein, M.D. (2003) "Technology makes inroads into imaging practice" Diagnostic Imaging October 2003.
- (Konishi et al. 1982) F Konishi and B C Morson (1982) "Pathology of colorectal adenomas: a colonoscopic survey." J Clin Pathol. 1982 August; 35(8): 830–841.
- (Lefere et al. 2002) P.A. Lefere and S.S. Stefaan, S. Gryspeerdt (2002) "Dietary fecal tagging as a cleansing method before CT colonography: Initial results polyp detection and patient acceptance" Radiology 224(2):393-403.
- (Levoy 1988) M. Levoy. (1988) "Display of surfaces from volume data. IEEE Computer" Graphics & Applications, 8(5):29–37, May 1988.

- (Macari et al. 1999) Macari M, Berman P, Dicker M, et al. (1999) “Usefulness of CT colonography in patients with incomplete colonoscopy” *AJR* 1999;173:561-564.
- (Masseroli et al. 2004) Masseroli, M.; Bonacina, S.; Pincioli, F. (2004) “Java-Based Browsing, Visualization and Processing of Heterogeneous Medical Data from Remote Repositories. Engineering in Medicine and Biology Society, 2004.” *IEMBS* 04. 26th Annual International Conference of the IEEE Volume 2, Issue , 1-5 Sept. 2004 Page(s): 3326 – 3329
- (Mildenberger et al. 2002) P. Mildenberger, M. Eichelberg, E. Martin (2002) “Introduction to the DICOM standard”. *European Radiology* 12(4)920-927.
- (Morson et al. 1983) Morson BC , Bussey HJR , Day DW , Hill MJ (1983). “Adenomas of the large bowel”. *Cancer Surveys* 2 : 451 - 478 .
- (NEMA 2003) National Electrical Manufacturers Association. (2003) *Digital Imaging and Communications in Medicine (DICOM)*. Rosslyn, Va: National Electrical Manufacturers Association; PS 3.1-2003–3.16-2003.
- (O’Halloran 2005) John O’Halloran (2005) “Compression of Medical (CT) Images” Thesis presented as component of Master in Electronic Systems. School of Electronic Engineering, Dublin City University, Vision Systems Group.
- (Paulson 2005) Paulson, L.D. (2005) “Building rich web applications with Ajax” *Computer* Volume 38, Issue 10, Oct. 2005 Page(s): 14 – 17 Digital Object Identifier 10.1109/MC.2005.330
- (Pennebaker et al. 1993) *JPEG: Still Image Data Compression Standard*, Van Nostrand Reinhold, New York 1993.

- (Pickhardt et al. 2003) P.J. Pickhardt, J.R. Choi, I. Hwang et al. (2003) "Computed tomographic virtual colonoscopy to screen for colorectal neoplasia in asymptomatic adults" *New England Journal of Medicine* 349(23):2191-2200.
- (Pickhardt and Choi 2003) P.J. Pickhardt and J.H. Choi (2003) "Electronic cleansing and stool tagging in CT colonography: advantages and pitfalls with primary three-dimensional analysis" *American Journal of Roentgenology* 181(3):799-805.
- (Rex et al. 1997) Rex DK, Cutler CS, Lemmel GT, et al. (1997) "Colonoscopic miss rate of adenomas determined by back-to-back colonoscopies." *Gastroenterology*;112:24-28.
- (Sadleir et al. 2002) R.J.T. Sadleir, P.F. Whelan, N. Sezille, J.F. Bruzzi, H.M. Fenlon, A.C. Moss, P. MacMathuna (2002) "Automated detection and flagging of potential colorectal neoplasia at CT colonography" 3rd International Workshop on Multislice CT, 3D Imaging and Virtual Endoscopy, Rome, Italy, June 6th - 8th.
- (Sadleir et al. 2004 a) R.J.T. Sadleir and P.F. Whelan (2004) "Fast colon centreline calculation using optimised 3D topological thinning" *Computerized Medical Imaging and Graphics* (to appear)
- (Sadleir et al. 2004 b) R.J.T. Sadleir, P.F. Whelan, P. MacMathuna et al. (2004) "A Portable Toolkit for Providing Straightforward Access to Medical Image Data" *Radiographics* 24(4) 1193-1202.
- (Soto et al. 2004) J.A. Soto and M. Barish and J. Ferrucci "CT Colonography Interpretation: Guidelines for Training Courses" Presented at: RSNA annual meeting; November 28-December 3, 2004: Chicago.

- (Soto et al. 2005) Jorge A. Soto, MD, Matthew A. Barish, MD and Judy Yee, MD (2005) "Reader Training in CT Colonography: How Much Is Enough?" *Radiology* 2005;237:26-27.
- (Slomka et al. 2000) Slomka PJ, Elliott E, Driedger AA. (2000) "Java-based remote viewing and processing of nuclear medicine images: toward the imaging department without walls". *J Nucl Med.* 2000 Jan;41(1):111-8. PMID: 10647613 [PubMed - indexed for MEDLINE]
- (Summers et al. 2004) R.M. Summers, J. Yao and C.D. Johnson (2004) "CT colonography with computer-aided detection: automated recognition of ileocecal valve to reduce number of false-positive detections" *233(1):266-272.*
- (Sunit et al. 2006) Sunit Sebastian, MD (2006) Study presented in May 2006 at the American Roentgen Ray Society Annual Meeting in Vancouver, BC.
- (Taylor et al. 2004) S.A. Taylor, S. Halligan, D. Burling et al. (2004) "CT colonography: effect of experience and training on reader performance" *European Radiology* 14(6):1025-1033.
- (Villavicencio et al. 2000) Villavicencio RT, Rex DK. (2000) "Colonic adenomas: prevalence and incidence rates, growth rates, and miss rates at colonoscopy". *Semin Gastrointest Dis* 2000;11:185-193.
- (Vining 1994) D.J. Vining, D.W. Gelfand, R.E. Bechtold et al. (1994) "Technical Feasibility of Colon Imaging with Helical CT and Virtual Reality" *American Journal of Roentgenology* 162(Suppl):104.
- (Young et al. 2004) Young N, Chang Z, Wishart DS (2004) "GelScape: a web-based server for interactively annotating, manipulating, comparing and archiving 1D and 2D gel images" *Bioinformatics* 20 : 976 2004.

(Yoshida et al. 2005) H. Yoshida and A.H. Dachman (2005) “CAD techniques, challenges, and controversies in computed tomographic colonography” *Abdominal Imaging* 30(1):26-41.

(Zafar et al. 1991) Zafar S, Zhang Yq, Baras Js (1991) “Predictive block-matching motion estimation for tv coding .1. interblock prediction” *IEEE transactions on broadcasting* 37 (3): 97-101 SEP 1991

(Zhanlin 2005) Ji Zhanlin (2005) “A HTTP Compatible Medical Image Server” Thesis presented as component of Master in Electronic Systems. School of Electronic Engineering, Dublin City University, Vision Systems Group.

Publications Arising from this Research

This work has been presented at the following conferences:

V. Luauté, R.J.T. Sadleir and P.F. Whelan (2006) "A Remote Access Training System for the Detection of Colorectal Polyps at Computed Tomography Colonography" ECR 2006 - European Congress of Radiology, Vienna, Austria, March 3rd - 7th.

V. Luauté, R.J.T. Sadleir and P.F. Whelan (2006) "An automatic evaluation strategy for a remote access CT colonography training system" Biosignal 2006 - The 18th International EURASIP (European Association for Signal, Speech and Image Processing) Conference, Brno, Czech Republic, June 28th to 30th
Conference technically co-sponsored by IEEE EMBS

V. Luauté, R.J.T. Sadleir, H.M. Fenlon and P.F. Whelan (2006) "A Reader Training System for CT Colonography" IMVIP 2006 - Irish Machine Vision and Image Processing, Dublin, Ireland, August 30th to September 1st.

Vincent Luauté, Robert J.T. Sadleir, Paul F. Whelan (2006) "A remote access web based tutor system for reader training at CT colonography" RSNA 2006 - Radiological Society of North America, Annual meeting, Chicago, USA, November 26 – December 1.