VICOT: HERRAMIENTA VIRTUAL COLABORATIVA PARA EL DESPLIEGUE DE IMÁGENES EN LA WEB

Esmitt Ramírez[1]* y Fernando Álvarez[1]

[1]Universidad Central de Venezuela, Facultad de Ciencias, Centro de Computación Gráfica.

Recibido: 20/09/2016 Corregido: 16/10/2016 Aceptado: 17/10/2016

RESUMEN La visualización de datos ayuda a mostrar información de forma interactiva, facilitando a los usuarios la interpretación de dichos datos. En el área de Medicina existe un gran incremento de esto, donde las imágenes representan una pieza fundamental en los diagnósticos. Además, la presencia de datos tridimensionales requiere que existan dispositivos de gama alta para manipularlos. En este artículo, presentamos Vicot, una herramienta web para el despliegue de imágenes 2D/3D en un ambiente colaborativo empleando una configuración de red simple. Su principal objetivo es desplegar imágenes médicas, permitiendo a los usuarios su manipulación de forma transparente y sencilla usando una comunicación no presencial entre los pares involucrados. Vicot está desarrollado bajo una arquitectura modelo-vista-controlador, adquiriendo los datos desde un repositorio DICOM con una interfaz sencilla. Las imágenes son desplegadas en alta resolución, y los volúmenes son desplegados bajo un enfoque de Ray Tracing empleando la GPU. La salida final es mostrada en un browser con soporte a HTML5 en un equipo de gama baja. Nuestra propuesta es ideal para la enseñanza y discutir casos clínicos. Las pruebas realizadas en cada etapa de Vicot permiten obtener tiempos adecuados para maximizar su rendimiento. Además, al emplear una interfaz web se logra una independencia del sistema operativo, ampliando su alcance a diferentes dispositivos como PCs, tabletas, o teléfonos inteligentes.

PALABRAS CLAVE Visualización, despliegue de volúmenes, trazado de rayos, sistema colaborativo.

VICOT: VIRTUAL COLLABORATION TOOL TO RENDER IMAGES ON THE WEB

ABSTRACT Data visualization helps to show the information interactively, being easy for users the interpretation of relevant topics. A huge increase has been noticed in Medicine where images represent a keystone in diagnosis. Also, in presence of 3D data, it may requires a high-end device to manipulate and control them. In this paper, we present Vicot, a remarkable web tool to render 2D/3D images in a collaborative environment over a simple network configuration. It main goal is rendering medical images and volumes, allowing users the easy and transparent manipulation of data using a non-presential communication between pairs. Vicot is developed under a model-view-control architecture, acquiring data from a DICOM (Digital Imaging and Communication in Medicine) repository with a friendly web interface. Images are displayed in high-quality resolution; volumes are rendered using a GPU Ray Tracing approach. Final output is shown in a HTML5-based web browser under any standard low-end device. Our proposal is ideal to teach and discuss medical cases. Tests performed on each stage of Vicot allows to obtain suitable limits to maximize its performance. Also, using an input/output web interface allows the independence of operative system, spreading its scope to different devices such as PCs, tablets, and smart phones. **KEYWORDS** Visualization, volume rendering, ray tracing, collaborative system.

INTRODUCTION

Nowadays, Medicine has been influenced by different software able in the digital world, improving the day-to-day clinical practice. Aided diagnoses, analysis of diseases, preoperative planning, assisted surgeries, and others, are remarkable examples of this influence. Thereby, there are different hardware machines for acquisition, manipulation and visualization of medical data such as 2D images or 3D images. For this instance, the 3D images are called volumes or volumetric data. In this way, this data is used by software to enhance the practical operations in several health care centers around the world[7]. Developing new techniques and tools to manage medical data become necessary. Integration of these techniques with the visualization process is a research field to ease the work to physicians.

At the same time, physicians also require the support of collaborators in order to obtain insights, comments or diagnosis about a specific medical case of a particular patient. Most of these collaborators are not attended in the same healthcare center, being a problem in communication between pairs. Then, managing the medical data and providing the acquisition hardware as transparent for physicians, also allowing the opinion of collaborators which are not presented in the same place, would be a very useful tool. Taking this as motivation, in this paper we present **Vicot**, a **Vi**rtual **co**llaboration **t**ool to render medical images on the Web.

Vicot manages medical data in a transparent way integrated into an existing PACS - *Picture Archiving and Communication System*. It allows the acquisition of a medical study to visualize join with other physicians, in the same virtual place. Basically, there are two main roles: **presenter** and **attendee**. A presenter manages a medical case, loading and presenting data to be shown to other physicians. An attendee takes part in a session where can visualize the presented data and comment interactively through a chat.

This paper is organized as follows: The next section presents a basic background associated with this research and we summarize the related work. Next, an overview of Vicot is presented to introduce the relevant points of our research. Following, all details in depth of our proposal are described. In order to test our approach, a set of experimental tests are performed and the observed results are discussed. Finally, we concluded in the last section with an outlook to future work.

BACKGROUND

In this section, we present some basic concepts that define the problem of

fundamentals on DICOM related to our work and web-based environments to volumetric data.

Dicom

The digital imaging system PACS (Picture Archiving and Communication System) involves an image management and communication, retrieval, processing, distribution and its rendering[4]. This system allows the interconnection between several physical workstation where patients are attended, this with the goal to improve the patient care and outcome. Thus, the digital image database and storage devices are required as main component of the architecture of any PACS. The digital images handle by PACS are defined as the standard format named as DICOM (Digital Imaging and Communication in Medicine)[2]. DICOM is the standard to describe different formats for medical images of different modalities (e.g. MRI - Magnetic Resonance Imaging, US - Ultrasound, CT - Computed Tomography, X-rays, IVUS - Intravascular Ultrasound) that can be exchanged with data and quality necessary for clinical use in several equipment (e.g. Phillips, General Electric, Siemens).

Inside a clinical center working over a PACS, all real world objects are interpreted as objects in DICOM (e.g. patients, RX, plates, equipment). This objects have attributes and properties associated which define them. The IODs (Information Object Definitions) are the responsible for standardized each one, describing all particularly features of each object. For instance, information such as general data of patient, sex, age, weight, allergies, and other relevant clinical information are stored under a unique IOD. The current DICOM standard, the PS 3.1-2008, includes 18 related parts which follows the ISO directives. Two fundamentals parts are distinguished as components DICOM: information object and service class. Information objects define a set of images, and the service classes describes the way to manage these images. Taken into account only the information objects, the data stored can be defined as header information and byte data itself. An image is defined as a set of consecutive frames under a specific file format, being more technically, an image is formed by a header and the data itself and can be formed by several layers.

The data is captured as DICOM attributes (i.e. using a device in PACS), which could be transmitted and processed between DICOM devices formerly known as Application Entities (AE). Also, a program is called AE which allows the exchange of information between pairs into a PACS structure. This process is a client-server based scheme where software applications provided interchange services of data called SOP (Service-Object Pairs). For example, the storage of a CT image taken by a Siemens scanner to PACS,

corresponds with a CT storage SOP. In DICOM definition, it is necessary identifies the different services providers, requesting namely as SCP (Service Class Provider) and SCU (Service Class User) respectively. In the example the CT scanner is the SCU and the digital data-file is the SCU, this process is known as association. Each association starts with a DICOM handshake where 2 applications interchange information between pairs (i.e. similar as a single handshake process in computer networking).

There are more services provided by DICOM, the majority involves the data transmission over the network. That is the reason why the network infrastructure will be particular in a respective health care center.

Web-Based Medical Data Rendering

It is noticeable, that a SCU or a SCP is running over a specialized workstation inside the network infrastructure where PACS lies. However, these workstation are associated at a particular operating system or a particular setup. This could be a problem in order to incorporate a new workstation or just in an initial setup of whole system. If the managing, controlling and rendering of the PACS data is achieved on a regular non-dependent workstation then the process will be more acceptable in different health care centers.

In 2012, Ramírez and Coto[9] proposed four architectures to render 3D medical data on a web browser (truly, any 3D data). One of those schemes is based on render the 3D study in the server side using a web server to be accessed from clients as HTTP requests. Thus, a server complaint with high-end architectures on processing time and storage space.

Visualizations using 3D viewers on side client explodes the browser (whether plugins or no) based on HTML5, WebGL, X3DOM and others. A remarkable example is a tool presented by Birr et al.[1] where a real-time rendering is performed over medical data reaching an impact as teaching tool in surgical learning. That tool uses X3D model and WebGL in order to achieve the web application independent of the employed medical format. Another remarkable example is the presented by Settapat et al.[10].

After, studying the advantages and disadvantages of existing proposal we developed an approach where the rendering is performed in the server side. This sent the visual output results to web browser clients connected to this server. Following, we outline the complete process of Vicot.

OVERVIEW

Our proposed approach is presented as a networking collaborative system to display and share medical data in a fast way using session rooms. Vicot

contains session rooms where users can enter and interact over a medical case presented by the host of session. Notice that each user can be physically attend on be place on different workstation. This session rooms are similar to a conventional chat with particular properties.

Taking a particular medical case using Vicot, the treating physician selects the study of patient and visualize as a 3D image (i.e. volume) or a 2D image (i.e. single slice). This study creates a session room, where others physicians physically placed on other locations or just using other workstations, can enter to the created room and discuss over the case. In both cases, it is necessary using a HTML5-based web browser, and the treating physician also requires access to the medical data stored into a PACS. Generally, the data stored into the PACS and the treating patient must be in the same physical place. From this point to the end, we focus only in the 3D images rendering using Vicot.

The basic structure of Vicot is shown in Figure 1. The process starts when a client, using a web browser from a computer, tablet or phone (left-bottom side) request a 3D volume to explore. The request is handle by the Control Manager which generates an order to the PACS module. The volume requested is on the local cache (upper-right side) or on the global database (upper-left side); in both cases, the Control Manager gets a copy and it is used by the 3D Rendering module to perform the volume rendering process. Notice that PACS module connects to a DICOM file server where patients are stored.

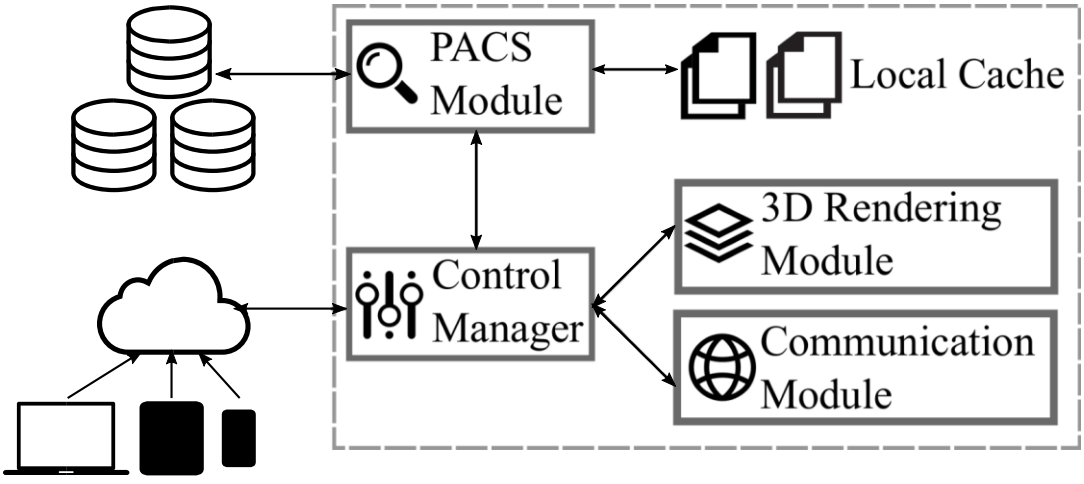


Figure 1 Basic structure of Vicot. The bottom-left side represents the user input interaction, follow by the control manager which is communicate with the PACS module. Both modules are connected with internal functionalities (3D Rendering and Communication modules, and Local Cache and Database).

Once the requested volume is rendered, the Communication module packs into a format to be sent to clients in an efficient way. Finally, the packed volume is received by a client and displayed. For instance, the Communication module allows the communication with a single final user or a group of users. In following sections we present in detail each module. First, we described the initial configuration required by Vicot for its proper behavior.

VICOT

From a technical point of view, Vicot is a web application based in the architectural pattern MVC (Model-View-Controller), being its main goal displaying medical data stored in an Image Manager/Image Archive manager. This is made using a collaborative approach based on a client-server. The server is developed in the C# programming language of Microsoft, and the client is HTML5-based (i.e. any supported browser).

Vicot was developed to be executed over a LAN or WAN network configuration. For a LAN network, a possible setup example is shown in Figure 2. Notice the internal IPs address, where it is necessary a DICOM server file (for instance, called DCM4CHEE) in a workstation, and other entity to connect with the DCM4CHEE (for instance, called DCMPal). The DCMPal contains a web server running the Internet Information Service (IIS) process, this machine (in the example, IP 192.168.1.104) is able to receive all incoming connections.

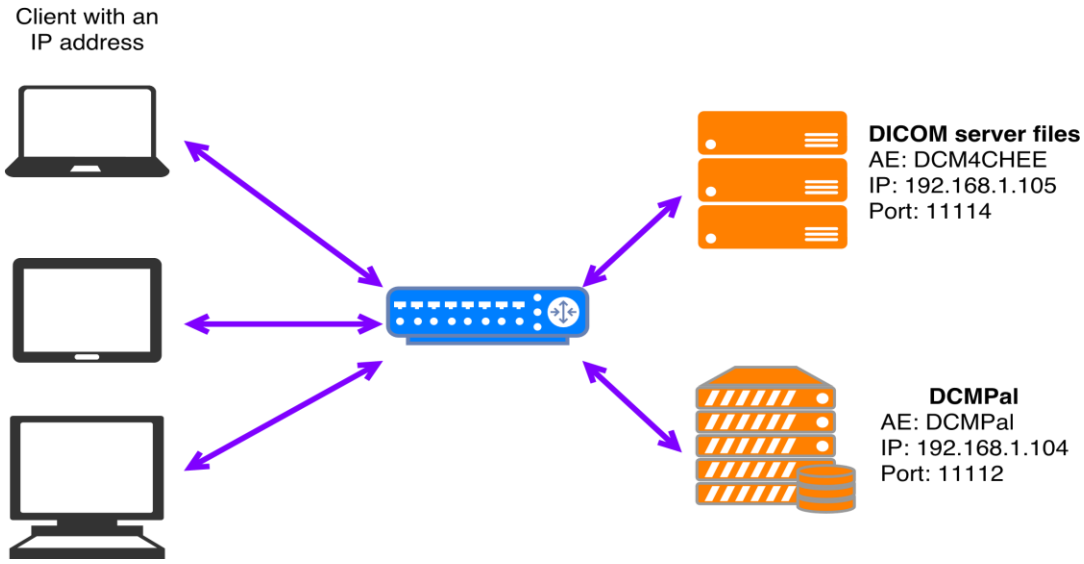


Figure 2 An example of the LAN networking structure. The example shows the DICOM server files and PACS server (DCMPal) over the same network interconnected with the workstations (i.e. clients with any valid IP address).

There are two libraries which are very useful to manage the DICOM files (i.e. grassroots), and to able the real-time connection server-client (i.e. signal R). Both are briefly explained before the Vicot's modules explanation.

Grassroots DICOM (GDCM)

The Grassroots DICOM, shortly named as GDCM, is a C++ open source implementation of a subset of DICOM definition[3]. GDCM offers functions to access the clinical data directly. A great advantage, is that GDCM includes a file format definition and a network communications protocol, both of which should be extended.

It offers wrapping other programming languages such as Python, C#, Java, PHP, and Perl. Also, it supports several image encoding such as RAW, JPEG (lossy 8 and 12 bits; lossless 8 and 16 bits), JPEG 2000 and RLE. For instance, in Vicot the GDCM was used as a C++ library mixing with C# implementation, over different images modalities offered by the DICOM standard.

SignalR

It is a library for ASP.Net to add real-time functionalities into web applications easily[11]. SignalR allows sent information from server to clients when these are available. This library offers an API to create remote procedure calls (RPC) linked with Javascript functions on client side. Also, it allows handling a set of client connections. In order to achieve the server-client communication, SignalR uses Websockets (if is available) or any other socket related connection, creating an abstraction layer of local implementations of each server to real-time connections.

Vicot manages two client-server communication models: persistent and hub connection. The persistent represents a tool to send information to a single client, group or broadcast. This kind of communication is able using low-level access in SignalR. The hub is a major abstraction level than a persistent, it allows to clients the invocation of functions stored in the server as local functions.

The PACS Module

The PACS module contains the necessary information to read, pack and unpack DICOM files. Also, it allows the communication between the protocol used in a PACS system and Vicot. Notice that the PACS system on Vicot is the dcm4chee software. The GDCM is used to achieve the communication with the PACS server, requesting and managing the location of the files location of DICOM series of all patients (called patient's study) available on the PACS server. In the same way, once a particular study is located and

selected, a SCP and SCU are created to obtain a local copy of it. After, the local copy is stored into the Local Cache repository.

Control Manager

The Control Manager stores all procedures getting from the client to the server, and internal process in Vicot. Also, Control Manager performs the rendering process of acquired volume. Once the DICOM file is locally stored, a volume is created as an intermediate data structure to be render by the visualizing algorithm. This intermediate representation is read from a set of images (i.e. as part of a study), to create a single one in 3D (see Figure 3).

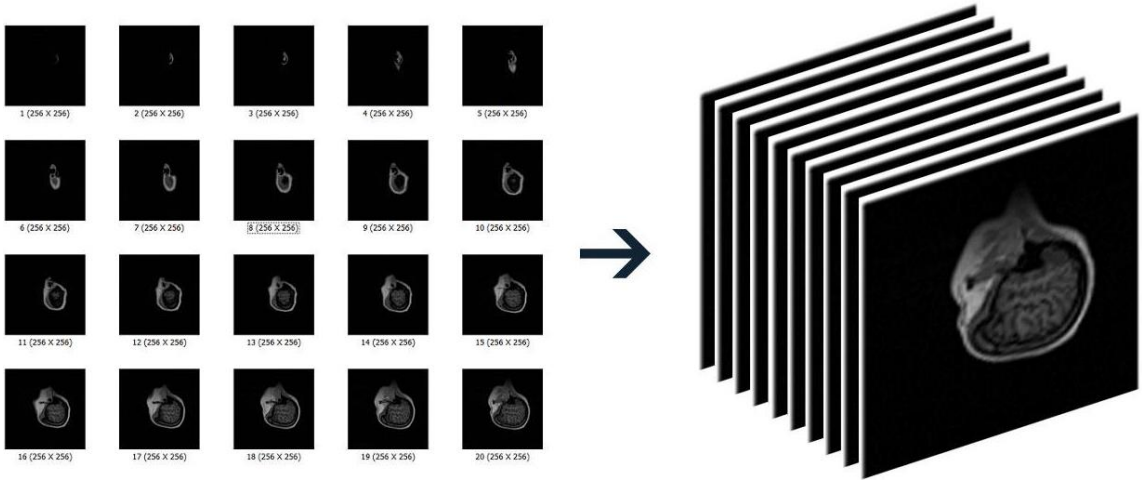


Figure 3 Representation of a volume data. The 3D is composed by a set of layers of 2D images. These are loaded into the internal GPU memory of server machine to be stored as texture in graphics hardware.

Vicot is limited to DICOM files without compression. Also, the transformation to be applied into stored volumes in the PACS server, are applied before create the intermediate volume. To this point, this representation is a 3D texture stored into the GPU memory of the server machine. Once the texture (internal representation of the 3D image) is loaded, a graphics library is used to perform the render to obtain an image as output, in Vicot we used OpenGL as rendered system.

3D Rendering Module

This module utilizes the Ray Casting on the GPU algorithm presented by Kruger and Westermann[5]. This was accomplished in the server side using C# as programming language, and the OpenTK library[6] to bind OpenGL. The mentioned algorithm address the integration of early ray termination and

empty-space skipping into texture based volume rendering on graphical processing units (GPU). Also, the algorithm requires GPU memory to load the 3D texture completely in order to speed up the performance of the rendering.

Communication Module

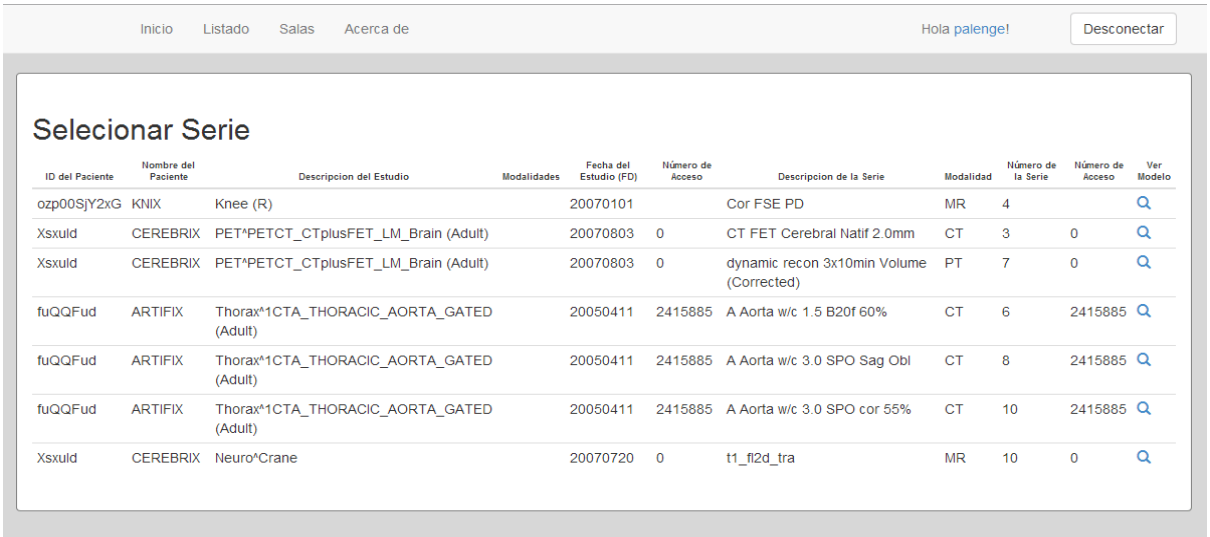
Each time a frame is generated by the 3D rendering module, it needs to be converted into an array of bytes to be displayed into a web browser. Moreover, this conversion generates a 2D image to be interpreted by a browser. To do that, a bitmap is generated to take the current frame from the rendering process. This bitmap is already in a format file image. The bitmap obtained is converted into an encoded string in Base64 to be sent at different web browser. The Base64 encode process is using due the characteristics of SignalR itself, and the easy conversion from this array of characters into a source tag into the canvas component of HTML5. This is achieved using a configuration into the HTML with the charset options.

Views

The views are defined as the content seeing by users. This is developed using ASP.NET and it contains the standard views such as landing page, login, about, register, and other. Particularly, there is a remarkable page available for users: List of Rooms.

The List of Rooms allows to join users into a created Room. A Room is defined as the place where user can collaborate. As mentioned in the introduction, there are two types of users: presenter and attendee. The presenter is whom will guide a session and control the meeting with the attendees. An attendee is a collaborator whom will join into a Room, in order to participate in the collaborative session. Only the presenter has the ability of manage the volume; attendee only can view and interact in a selected Room.

In server side, there are a list of active Rooms. Each Room is associated to a patient's study. The presenter can view a list of available studies, each study contains a set of “Series”. Then, before starts a Room the presenter might select a Serie as shown in Figure 4.



ID del Paciente	Nombre del Paciente	Descripcion del Estudio	Modalidades	Fecha del Estudio (FD)	Número de Acceso	Descripcion de la Serie	Modalidad	Número de la Serie	Número de Acceso	Ver Modelo
ozp00SjY2xG	KNIX	Knee (R)		20070101		Cor FSE PD	MR	4		
Xsxuld	CEREBRIX	PET^PETCT_CTplusFET_LM_Brain (Adult)		20070803	0	CT FET Cerebral Natif 2.0mm	CT	3	0	
Xsxuld	CEREBRIX	PET^PETCT_CTplusFET_LM_Brain (Adult)		20070803	0	dynamic recon 3x10min Volume (Corrected)	PT	7	0	
fuQQFud	ARTIFIX	Thorax^1CTA_THORACIC_AORTA_GATED (Adult)		20050411	2415885	A Aorta w/c 1.5 B20f 60%	CT	6	2415885	
fuQQFud	ARTIFIX	Thorax^1CTA_THORACIC_AORTA_GATED (Adult)		20050411	2415885	A Aorta w/c 3.0 SPO Sag Obl	CT	8	2415885	
fuQQFud	ARTIFIX	Thorax^1CTA_THORACIC_AORTA_GATED (Adult)		20050411	2415885	A Aorta w/c 3.0 SPO cor 55%	CT	10	2415885	
Xsxuld	CEREBRIX	Neuro^Crane		20070720	0	t1_fl2d_tra	MR	10	0	

Figure 4 Example of selection of a clinical case "Serie" . The view is only available for the presenter which selects the "Serie" to analyze into a Room, inviting another collaborators to participate.

In the other side, an attendee can enter into a created Room and chat with other attendees or with the presenter using a writing chat. Figure 5 shows the view of the attendee once enters at a Room. In the right side, there is the message chat which is synchronized with all attendees and presenter. In the left side, is the displayed volume rendering model manage by the presenter.

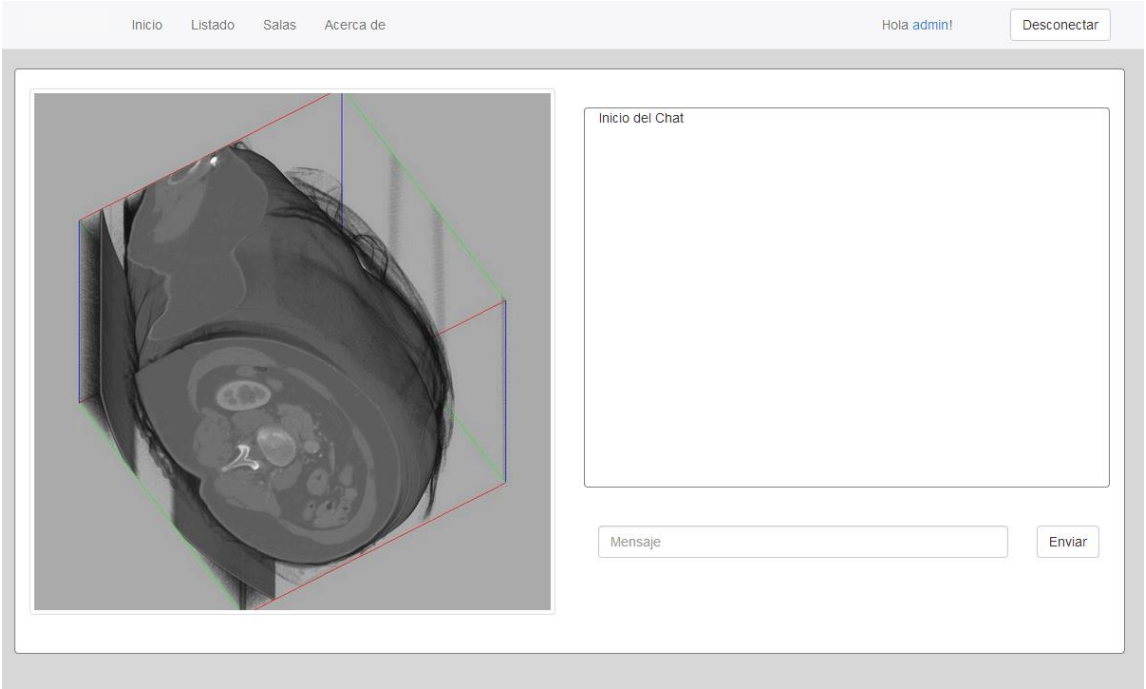


Figure 5 View for attendee to interact into a Room session. An example of the view displayed on the client browser for the attendee into a Room session, notice the volume in left side and the chat in the right side.

In the same Room, presenter have others views where control the volume performing operations such as rotation, scaling, contrast/brightness, and modification in the volume transfer-function (see Figure 6). Vicot is thought to run in a low-end hardware for clients, due rendering process is achieved in the server side. This is a remarkable aspect of Vicot to work on web browser clients.

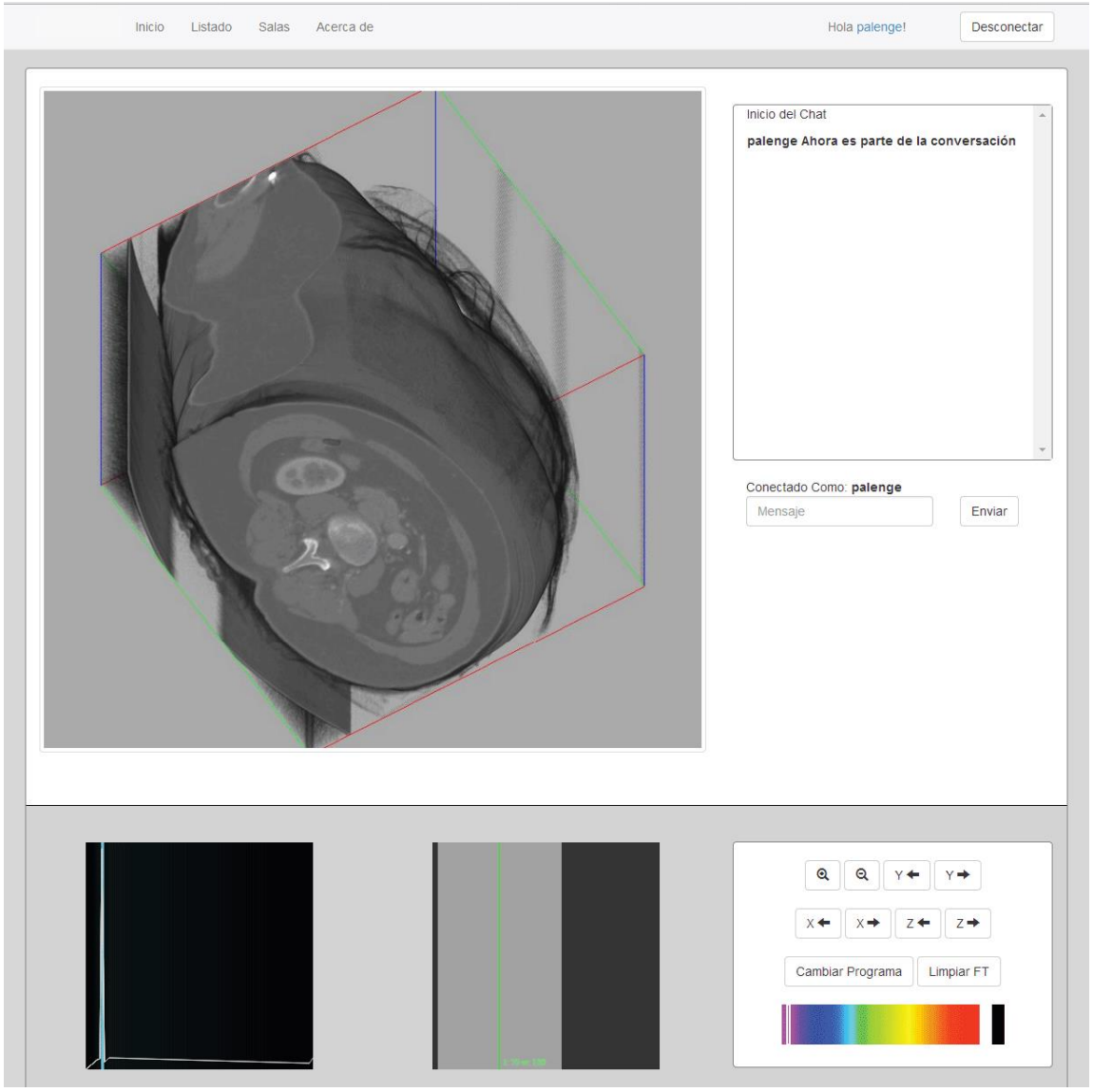


Figure 6 View for presenter to interact into a Room session. An example of the view displayed on the client browser for the presenter into a Room session, notice the volume in left side, the chat in the right side, and the transfer function and options in the bottom side.

TESTS AND RESULTS

This section shows a set of results getting from analyzed Vicot. It is important

to notice that large processing is on the volume rendering. Then, tests are mainly related to factors that are involved on the rendering. The input of Vicot are DICOM files, taken from Osirix project (available in http://www.osirix-viewer.com/), can be appreciated in Table I. Tests were executed on an Nvidia 650m SLI with an Intel i7 2.4 GHz, RAM of 16 GB running over Windows 8.

Table I Dataset employed

Name	Width	Height	Number of slices
Knix	512	512	20
Cerebrix	512	512	174
Artifix	512	512	374

The three types of volume dataset employed in our tests.

Textures 3D

As previous mentioned, to render a volume it is necessary create a 3D texture. The time required to display the volume depends of its dimension. For instance, the data type of a DICOM file requires an extra processing to convert it into the range of 0-1 (i.e. 0 represents darker color, and 1 the brightness).

The Figure 7 draws the times obtained in the test. To create the 3D texture on the Knix model was 2 ms, the Cerebrix was 11 ms, and the Artifix was 31 ms. It is clear the increasing of time depends of the number of slices (i.e. all volumes have the same size of 512 x 512 pixels). Notice that Artifix requires more time than others, more than twice.

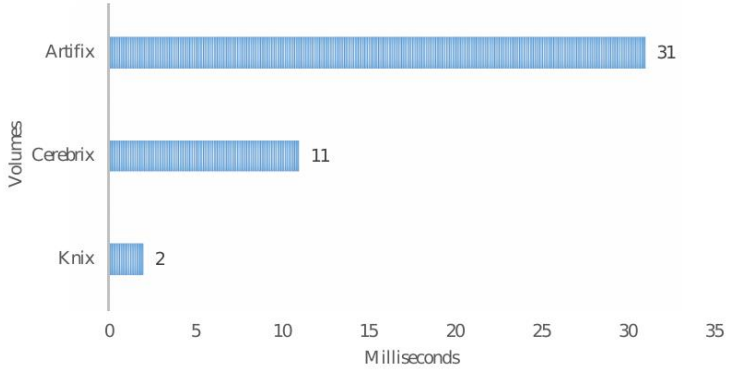


Figure 7 Times to create the 3D texture. Our tests include three different volumes to obtain an average on the time to create the 3D texture.

Output Window

Rendering the volume as itself is the keystone of Vicot. There are two impact factors on render: size of the volume and size of the displaying windows. The output window is associated with the space into the web browser to perform

the rendering (i.e. the canvas size). We test in 400 x 400, 600 x 600, and 800 x 800 pixels of canvas size, taking the Cerebrix as base model.

Table II presents the time in milliseconds to create the volume for three different window sizes. Given the technique employed to the volume rendering, number of pixels drawn into the output windows affects directly. For example, a size of 200 x 200 pixels can take the same time of a window of 800 x 800 pixels, due that it depends of the number of drawn pixels in the output window. Thus, the 80% of drawn pixels in 200^2 resolution could take the same time of the 20% of drawn pixels in 800^2 resolution.

Table II Time to create Cerebrix

Size of window	Time in ms
400 x 400 pixels	7
600 x 600 pixels	17
800 x 800 pixels	29

It shows different window size to create the Cerebrix model.

First frame

After loaded the volume the next step is visualize the model by the 3D Rendering Module. When the render is performed on the server side, the setup and initial configuration consume some time before displaying the volume. Taking the time of the first viewing frame is relevant in our study. Indeed, we consider a medium output window size (i.e. 600 x 600 pixels) to perform the test. The obtained results are shown in Table III.

Table III Time to obtain first frame

Volume	Time in ms
Knix	3
Cerebrix	17
Artifix	32

Using a window of 600 x 600 pixels, this table shows the time to obtain the first frame for all dataset employed.

The time to obtain the first frame of the Artifix volume is larger, due to it requires more GPU textures to represent the volume. This aspect can be noticed in the number if slices in Table I. To this point, the server has the first frame of the volume rendering already displayed. However, this image is only available to the server.

In order to send images of rendered volume to different clients, this will be

package to be received correctly on web browsers using the HTTP protocol.

Communication

Once a frame image is generated, an intermediate representation must be created. The image may be in different format (e.g. JPEG, PNG and GIF) and the intermediate representation is unique for all of them. We chose a string representation encoding in Base64 for the frame image.

It is noticeable that image have to be created in real-time. Accordingly, the processing time to generate the intermediate representation is considered. Table IV shows the time to create this representation from three different images format files to Base64 (2nd column). This test is using the Cerebrix volume dataset.

Table IV Conversion time from image to Base64 and sending time

Format Image File	Image to Base64	Server to Clients
PNG	3	20
JPEG	2	20
GIF	14	40

Using three file formats, the time to convert from image to Base64 and the time to send image to clients is presented, measured in milliseconds.

In our implementation, the format file impacts in the general performance. Convert a GIF file to its Base64 representation requires more time than other formats. However, this is not the only factor that affect the time to send image to clients, size of images impacts directly on the total time.

According the Table IV (3rd column), the time to send images to clients is indistinctly between JPEG and PNG format file. Now, the GIF format is not recommended to be used in Vicot. We decide to user select JPEG or PNG (e.g. configuration in the server) to convert the output of the volume rendering.

Web Browsers

The last test compares the time to load and display the output image obtained from the volume rendering in server to the client side. We used the Cerebrix model under the web browsers Google Chrome, Mozilla Firefox, and Internet Explorer. Figure 8 shows the results obtained.

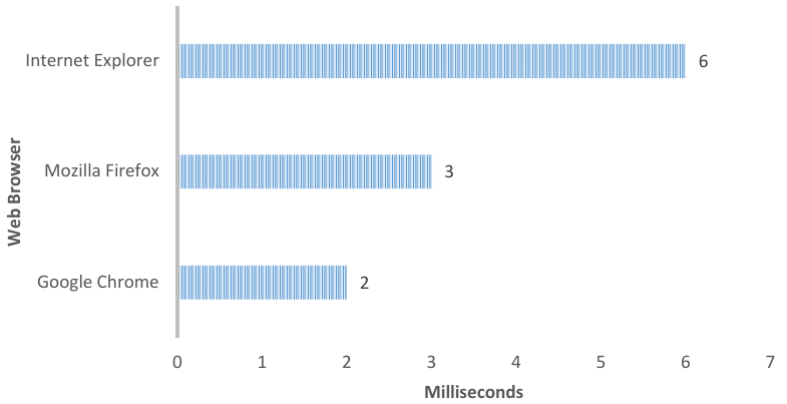


Figure 8 Times to render Cerebrix on different browsers Explorer, Firefox and Chrome were used to measure the time of the Cerebrix model.

Google Chrome and Mozilla Firefox offer a better performance than Internet Explorer on render over a canvas component, the Base64 encoding image. Despite of this test was only for Cerebrix, we tested using other dataset and the behavior is similar. The final output models can be appreciated in Figure 9: (a) Knix, (b) Cerebrix, and (c) Artifix.

(a) (b) (c)

Figure 9 Visual rendering output of models. The visual result, after the rendering process, is shown when is displayed into the canvas of the presenter or attended joined into a Session Room.

It is remarkable highlight over the final device to show the rendered models. Modern devices such as tablets or smartphones are suitable to be used with Vicot. However, the limitation of the canvas size is the only limitation, requiring at least 1000 width pixels to display the volume, also the chat panel. The height dimension is not considered as a problem, due the scrolling option on these devices.

CONCLUSIONS

In this paper, we presented a software solution called Vicot to render 2D/3D images on the Web using a collaborative approach. The architecture is based on a well-known software engineering approach in order to obtain the best structure (to include new features) and to obtain a better performance including external and very useful libraries into Vicot. Also, it allows the separation between presenter side (access to PACS server) and attendee side (access to HTML5-based web browser), offering a flexibility over the networking system implemented. Vicot can be used in clinical cases, teaching and training sessions, regular trails, among others. This allows intervention of collaborators not present in the same place. The collaborators only require a HTML5-based browser.

Also, according tests performed, the required hardware for attendees is thinking on low-cost devices. However, the presenter side (from the point of view of final user) or server side (from the point of view of architectural designer) should be a high-end workstation with a medium or high performance graphics card. This aspect is crucial to achieve the render completely based on the aspect that server realized the render, and client only receive a final rendered image. Moreover, the structure of the network is fundamental and it is linking to others factors (not considered in this study) such as bandwidth, network congestion, among others.

We plan extend Vicot to add more functions to manipulate images on Rooms. For instance, functionalities such as different views, predefined palette colors, effects over volumes (e.g. maximum intensity projection, illustrative rendering, etc.), and others. Also, getting lowest time in the rendering is a field to explode to reduce the total time in obtain an image.

Similarly, in the future we plan explode the Vicot composition to integrate it into a previous research[8] to support 3D images, also including the collaborative approach taking more aspects such as video calling, real-time drawing interaction and parallel session using the same dataset.

REFERENCIAS

1. **Birr, S., Mönch, J., Sommerfeld, D., Preim, B**. A Novel Real-Time Web3D Surgical Teaching Tool based on WebGL. *Bildverarbeitung für die Medizin, snformatik aktuell*, pp. 404-409, 2012.
2. **DICOM**. *National Electrical Manufacturers Association (NEMA)*. Retrieved: February 9, 2015. Available http://dicom.nema.org/
3. **GDCM**. *Grassroots DICOM library*. Retrieved: Aug 15, 2016. Available http://gdcm.sourceforge.net/wiki

4. **Huang, H.K**. PACS and Imaging Informatics: *Basic Principles and Applications*. John Wiley & Sons, 2nd edition, January, 2010.
5. **Kruger, J., Westermann, R**. *Acceleration techniques for GPU-based volume rendering*. In Proceedings of the IEEE Visualization, pp. 287-292, 2003.
6. **OpenTK**. *The Open Toolkit*. Retrieved: Aug 15, 2016. Available http://www.opentk.com/
7. **Preim, B., Botha, C**. *Visual Computing for Medicine*. Morgan Kaufmann. 2nd edition, 2014.
8. **Ramírez, E., Coto, E**. TRAUMAPLAN: para la planificación preoperatoria en traumatología. *Rev Venez Info, Tecnol Conoc*, **8(**2):61-78, 2011
9. **Ramírez, E., Coto, E**. *Web visualization of 3D medical data with open source software*. In Proceedings of Jornadas de Investigación Encuentro Académico Industrial Facultad de Ingeniería (JIFI), Bioengineering Section, Fac Ingeniería, Universidad Central de Venezuela, 2012.
10. **Settapat, S., Achalakul, T., Ohkura, M**. *Web-based 3D visualization and interaction of medical data using Web3D*. In Proceedings of the SICE Annual Conference 2010, pp. 2986-2991, 2010.
11. **SIGNAL**. *ASP.NET SignalR*. Retrieved: Aug 28, 2016. Available http://www.asp.net/signalr

**Correspondencia: ***Profesor Esmitt Ramírez. Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela, Los Chaguaramos, Caracas 1010-A. Correo electrónico: esmitt.ramirez@ciens.ucv.ve