2019

Web-based Medical Data Visualization and Information Sharing Towards Application in Distributed Diagnosis

Qi Zhang
Illinois State University, qzhan10@ilstu.edu

Informatics in Medicine Unlocked 14 (2019) 69–81

Contents lists available at ScienceDirect

Informatics in Medicine Unlocked

journal homepage: www.elsevier.com/locate/imu

Web-based medical data visualization and information sharing towards application in distributed diagnosis

Qi Zhang

School of Information Technology, Illinois State University, 100 North University Street, Normal, IL, 61761, United States

ARTICLE INFO

Keywords:
Web rendering
Medical information
Information extraction
WebGL2
Internet
Information sharing
Medical data handling
Voxel classification
Distributed disease diagnosis

ABSTRACT

Network based medical data computing and collaborative visualization have been commonly used in remote medicine and distributed diagnosis, where visualizing 3D medical data on web browsers and sharing medical information on internet are critically important. However, due to the lack of efficient algorithms and compatible graphics hardware support, there are still some major technical challenges in web based medical data visualization and information exploration on internet. To address these practical issues, we created a new network based medical data rendering and information sharing system, where an Apache HTTP Server was applied to handle data information, and MySQL and PHP were exploited for data storage and management. In this system, medical data rendering and computation were supported with GPU and WebGL 2.0 (WebGL2), and a novel data information extraction algorithm was designed to optimize data storage and management. Taking advantage of the new 3D features of WebGL2, a web based raycasting algorithm was developed to deliver real-time data visualization on web browsers, in which a novel voxel classification method was integrated for color mapping and high-quality image generation.

The developed medical information system can deliver 60 ± 2 frames per second rendering performance for high-quality medical data visual exploration on modern browsers as well as medical information communication on internet. The system has been seamlessly integrated with web server, database and client computers equipped with modern graphics hardware, which has wide applications in the fields such as internet based computer-aided medical decision and education, as well as distributed disease diagnosis.

1. Introduction

The evolution of web technologies and browser features in recent years has made it possible to develop interactive medical data solutions on internet and improve the delivery of healthcare to the whole society [1]. Thanks to the development of graphic accelerators and web application tools, interactive medical data visualization can be shared on internet and is thus easily and widely accessible [2–4]. In addition, as an enhancement of internet based medical visualization, volume rendering on the web is an important technique to visualize 3D medical datasets and provide critical information about the spatial relationships of different tissue structures [5–7]. Web based volumetric data visualization can grant doctors and medical researchers the capability to collaboratively perform disease diagnosis and preoperative surgery planning from anywhere around the world where internet is available [8,9].

The importance of web based data visualization particularly in medicine is increasing due to the advancements in technology and healthcare systems [10,11]. There are many interesting research and applications of web based medical data visualization, for example, Congote et al. presented a raycasting algorithm implemented with WebGL and HTML5, which was based on 2D texture composition and post-color classification method [12]. Mobeen and Feng designed a single-pass rendering pipeline for mobile devices, where they used 2D texture to simulate 3D texture for volume rendering [13]. Mahmoudi's group developed an interactive 2D and 3D medical image processing and visualization software package using AJAX technology in web-user-interface [14], and a software architecture was presented by Marion and Jomier for visualizing 3D medical datasets on internet using WebGL and WebSocket, which was applied for collaborative data exploration [15]. Kaspar et al. [16] developed a web-based system, i.e., CoWebViz, for medical data stereoscopic visualization. Besides the application of medical data rendering, Rego and Koes [17] used WebGL and JavaScript to display molecular structures on the web, and Jaworski et al. [18] implemented a raycasting platform to show composite materials microlevel structure models on network. WebGL-based raycasting was also implemented to visualize protein structures by Sherif and his colleagues [19], and the similar technology was implemented to display macromolecular structure in the application of computational drug discovery by Yuan's group [20]. To allow users to explore the medical data in virtual

E-mail address: qzhan10@ilstu.edu.

https://doi.org/10.1016/j.imu.2018.10.010
Received 7 October 2018; Received in revised form 27 October 2018; Accepted 27 October 2018
Available online 12 November 2018

reality (VR) environment, Kokelj and his colleagues integrated VR into a web-based medical visualization framework [21].

WebGL, JavaScript and HTML5 were also employed to visualize 3D surface and volumetric neuroimaging data in modern web browsers [22], dynamically display and analyze genome on internet [23], build web interface for 3D medical data interactive segmentation [24], and develop surgical teaching tools through integrating interactive web-based 3D models derived from patient-specific data [25]. The similar technology was also used to render whole-body anatomy via a web interface [26], interactively visualize and analyze 3D fractal dimension of MRI data on the web [27], implement web applications to visualize dynamic molecular data [28], interactively browse and segment medical images on remote client computers [29], and collaboratively represent neurological images using Google Drive Realtime API and web browsers [30]. Recently, Shi et al. [31]. introduced a 3D modeling program, i.e., Web3DMol, which was a web application focusing on protein structure visualization in modern web browsers. Tiwari et al. [32] presented a web based rendering platform using image streaming, where Wamp Server was employed to show 3D images on client machines. Campoalegre et al. [33] designed a client-server framework for interactive data visualization on standard mobile devices, where a compression scheme was designed to improve data transmission on internet. Furthermore, a performance optimization model for web-based interactive visualization was proposed by Halic et al. [34]. Through integrating remote volume rendering and surface rendering algorithms, Qiao et al. [35] built a web-based platform that allowed users to access large-scale 3D medical data on internet using a Master-Slave interaction mode, while Pienaar et al. developed a cloud-based medical image data and information management software platform called CHIPS for clinical data sharing and exploration [36].

2. Comparison with existing work

For current applications, high-level libraries such as Visualization Toolkit (VTK) [37] and Three.js are commonly used directly to visualize medical data on the web directly without developing new custom high-quality volume rendering algorithms, which result in inflexible scene rendering primitives and limited medical data rendering quality and speed, making it difficult for the software systems to satisfy modern clinical requirements [20,25,35]. Next, for publications that are based on custom developed volume data visualization algorithms, WebGL 1.0 was generally used for voxel sampling and data calculation without 3D texture support on graphics hardware, these algorithms used 2D texture to simulate 3D texture for data sampling and interpolation, which generated image artifacts and low rendering efficiency [12,13,16,17,26,31]. In addition, for data and information processing and visualization, using algorithms without efficient color mapping and voxel classification can result in inefficient clinical data exploration and information extraction due to the incapability of interactively updating the rendered medical and molecular data [23,28,30,34]. Many applications used server to render the data and send the result to the clients, which would increase the computing burden of server and slow down the system's performance when multiple clients connected with the server and requested data processing at the same time, and the there were also information security concerns when sending data to sever on internet without security protection [14,32,36]. Furthermore, the confliction of the internet speed and data transmission would further delay data display and exploration on remote client computers [24,27,29].

Due the challenges such as image quality, visualization speed, flexibility, data and information transmission efficiency and security, there are still many unsolved issues related to the network based collaborative medical data visualization and information sharing. In this paper, we described new algorithms and a software framework to address the mentioned issues. First, for medical data visualization, our algorithm was based on WebGL 2.0 (WebGL2) [38] and low-level graphics programming instead of using WebGL 1.0 or high-level libraries such as VTK and Three.js, which allowed us to exploit 3D texture for raycasting computation on graphics processing unit (GPU) directly instead of using the traditional solution of using 2D texture for voxel extraction and sampling in 3D space. Our algorithm conducted trilinear interpolation at each sampling step on GPU's vertex and fragment shaders. In addition, we integrated some efficient methodologies to improve the rendering speed and image quality, such as early ray termination and segment-based post color-attenuated classification, which enabled our system to render medical data in high quality and can interactively change the appearance of the rendered image for diagnosis and medical data exploration. Second, we implemented a novel image information extraction algorithm to save the storage space of medical data, so this system could effectively transmit data between server to client computers and launch the program on client computers in near real time. Third, Apache HTTP Server [39] was used to build web server and work with MySQL [40] database and PHP (HyperText Preprocessor) [41] server-side scripting language for data and information transmission and management. Users could interactively extract data information and update database on remote client computers. In addition, some security features, such as granting users different level of privileges to access and manage database, have been integrated into this system to protect user privacy and data information on internet. The developed algorithms and software can provide important functions for medical applications such as distributed diagnosis on internet.

3. Rendering data on web server

3.1. Server side programming

In this network based platform, a server scripting language PHP (Hypertext Preprocessor) was used for server side programming. PHP is a powerful tool for making dynamic and interactive Web pages, and can generate dynamic page content. In this application, PHP and HTML5 were employed to develop programs for the network based system framework. After the PHP code was interpreted and executed, the web server sent the resulting output to its client in the form of generated web page. The PHP code was utilized to generate a web page's HTML code, an image, or some other data. In this software platform, the PHP code ran on the server side, and emitted files that were then sent to the client/browser, in which WebGL2 was running and was called from JavaScript. So, our platform could emit WebGL2/JavaScript source code from within a PHP program running on the server.

3.2. Volumetric data visualization

Direct volume rendering is an algorithm for 3D data visualization, which can produce a projected image directly from volumetric data without intermediate constructs. In the visualization process, volumetric data is sampled on a rectilinear grid with a trilinear interpolation, and a transfer function is used to assign optical properties like color and opacity to each sampling point. To generate an image, the lighting optical properties should be integrated along each viewing ray, which affect the light passing through a volume due to absorption, scattering, or emission of light from small particles. The particles in an actual cloud occlude incoming light and add their own glow. Thus a realistic differential equation should include both source and attenuation terms:

$$\frac{dI}{ds} = \psi(s) - \tau(s)I(s) \tag{1}$$

where s is a length parameter along a ray in the direction of the light flow, and $I(s)$ is the light intensity at distance s. The source term $\psi(s)$ is a function of position specified by an independent transfer function. The common optical model for volume rendering is the absorption plus emission model, in which the volume itself emits and absorbs light, and the absorption and emission are evaluated at each voxel throughout the volume. Eq. (2) is the solution of Eq. (1):

$$\frac{d}{ds}\left(I(s)exp\left(\int_0^s \tau(t)dt\right)\right) = \psi(s)exp\left(\int_0^s \tau(t)dt\right) \tag{2}$$

Integrating from $s = a$ at the edge of the volume to $s = b$ at the eye, and then bring the I_0 to the other side, and multiplying by $exp(-\int_0^s \tau(t)dt)$, we can get Eq. (3) as the solution for $I(a, b)$.

$$I(a, b) = I_0 exp\left(-\int_a^b \tau(t)dt\right) + \int_a^b \left(\psi(s)exp\left(-\int_s^b \tau(t)dt\right)\right)ds \tag{3}$$

In Eq. (3), the first term represents light coming from the background and entering the volume, i.e., I_0 is the light intensity at $s = a$, which is multiplied by the cloud's transparency. The second term is the integral for the contribution of the source term $\psi(s)$ at each position s, multiplied by the transparency $T'(s) = exp(-\int_s^b \tau(x)dx)$ between s and the eye (position b on the casting ray). Define $\psi(s) = C(s)\tau(s)$, where $C(s)$ is a particle's color at the position s, and the quantity $\tau(s)$ is called the extinction coefficient, we can the following Eq. (4).

$$\begin{aligned} I(a, b) &= I_0 exp(-\int_a^b \tau(t)dt) + \int_0^D \psi(s)T'(s)ds \\ &= I_0 exp(-\int_a^b \tau(t)dt) + \int_a^b C(s)\tau(s)T'(s)ds \end{aligned} \tag{4}$$

For volume rendering calculation, the initial intensity of a ray of light before it hits the volume, i.e., the first item of Eq. (4), is set to zero. We can then get the volume rendering integral Eq. (5).

$$I(a, b) = \int_a^b C(s)\tau(s)T'(s)ds \tag{5}$$

3.3. Numerical calculation

The integrals in Eq. (5) can be calculated using numerical integration. The Riemann sum $\sum_{i=1}^n \phi(x_i)\Delta x$ is used to approximate the integral $\int_a^b \phi(x)dx$. The interval from a to b is divided up into n equal segments, getting length $\Delta x = D/n$, and a sample x_i is chosen in each segment, so that $(i-1)\Delta x \le x_i \le t\Delta x$. Set $x_i = i\Delta x$, then $t_i = \tau(i\Delta t)exp(-\tau(i\Delta x)\Delta x)$ can be thought of as transparency of the i^{th} segment along the ray. Setting $C_i\tau_i = C(i\Delta t)\tau(i\Delta t)$ and approximate the transparency $\tau(i\Delta x)exp(-\int_{i\Delta x}^b \tau(x)dx$ between x_i and b by $\Pi_{j=i+1}^n t_j$. The Riemann sum for $\int_a^b C(s)\tau(s)exp(-\int_s^b \tau(x)dx)ds$ then becomes $\Sigma_{i=1}^n C_i \Pi_{j=i+1}^n t_j$. The final estimation of Eq. (5).

$$I(a, b) \approx \sum_{i=1}^n C_i \prod_{j=i+1}^n t_j = \sum_{i=1}^n C_i \prod_{j=i+1}^n (1 - \alpha_j) \tag{6}$$

In Eq. (6), α_j is opacity of the j^{th} segment, and the samples are evaluated from the sampling point to the eye on the casting ray, which is referred as back to front optical composition order.

3.4. WebGL based implementation

3.4.1. Data processing and loading

Commonly used 3D medical data sets, such as DICOM, NIfTI, or metadata, are composed of 2D image slices aligning in the x, y, or z direction. In this project, ImageJ [42] was used to load the 3D data into CPU main memory, and then transformed the loaded 3D image into a large 2D montage image, i.e., tilling each slice beside the other in a matrix configuration. The generated large 2D montage image can be used as a 2D texture, which can be loaded by WebGL 2.0, i.e., WebGL2 and save data storage space. Optimized data format and storage size are important for fast data transmission and processing on network. The following is the pipeline of 2D image data loading and 3D texture generation.

For WebGL based data rendering and visualization in HTML web browsers, the browsers cannot load 3D data directly into the WebGL rendering pipeline. The *HTMLImageElement* was used as an interface to provide special properties and methods for manipulating the layout and presentation of < *img* > elements, and the *HTMLImageElement.src* was employed to contain the loaded 2D data/image, which is a *DOMString* that reflects the *src* HTML attribute and contains the full URL of the image. The HTML based web browsers can only display 8-bit data, so JavaScript's array *Uint8Array* was employed to hold an array of 8-bit unsigned integers with the dimension 2D slices multiply the number of slices.

The data processing algorithm exploited the HTML canvas's *drawImage()* method to draw the loaded big 2D image to graphics frame buffer. Then, the *getImageData()* method of the WebGL's *CanvasRenderingContext2D* class was employed to create the data texture, which was then used with the function *texImage3D()* to generate 3D texture. Because commonly used web browsers can't deal with images that are very large in any one dimension, so we generated the 2D montage image with multiple rows and columns, as shown in Fig. 1, where there are total 172 slices, and there are 4 columns in the generated 2D montage MR brain image. The rendering system transfers the number of images per row, the number of rows, the total number of slices, and distance between z axis to the JavaScript image reader. Our program transformed the 2D image into 3D texture and then loaded the texture into the vertex and fragment shaders for raycasting computation on web browsers using WebGL2.

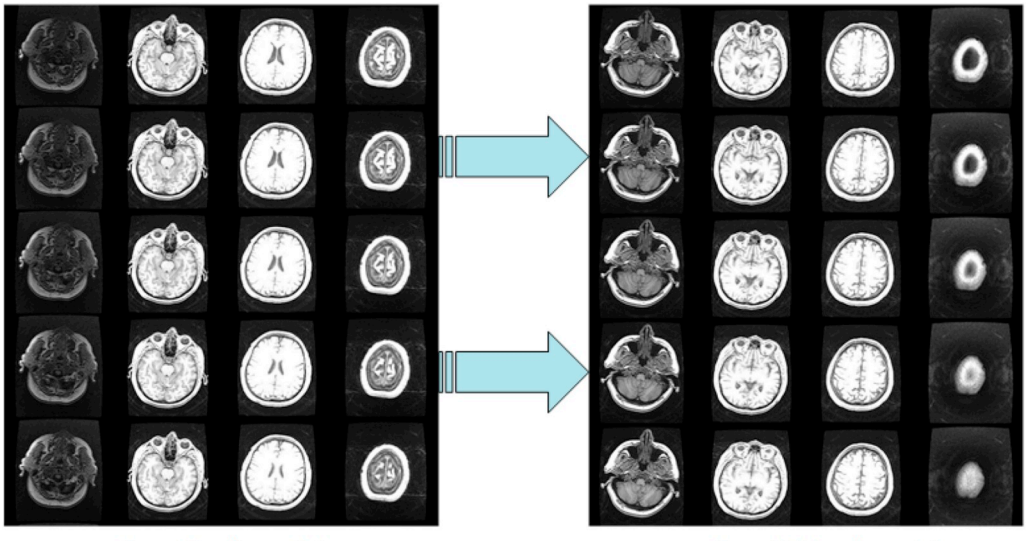


Fig. 1. Two image parts selected from a generated montage 2D MR brain image using a 3D MR data set. Total slice number is 172, and there are 4 columns in the montage image, so there are 43 slice images per column. The left side image is the top 5 rows, while the right side is selected from the rows 15 to 20 from the montage MR brain image.

Fig. 2. Color and opacity mapping for data visualization in a web browser interface. The bottom in the left side is the color at equally sampling points from 0 to 255, and the left top is the corresponding opacity values for each color point. The control points can be manipulated by users interactively, which will dynamically update rendered 3D image. The point values between the control points are acquired by linear interpolation. The right side is the rendered 3D brain image on a Firefox web browser using left side transfer function and the described WebGL2 based raycasting algorithm. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

3.4.2. Rendering pipeline

In this research, we took advantage of the new 3D texture support in the WebGL2 for volume raycasting calculation on the graphics processing unit (GPU). This image-based method renders a 3D scalar field into a 2D image by casting rays along the 3D volume. Each pixel on the screen is the result of a ray that is going through the cube and getting intensity samples from the voxels at regular intervals. Using 3D textures for volume rendering, the slices can be oriented perpendicular to the viewer's line of sight and will not lead to sampling errors. Furthermore, a 3D texture can naturally store normal textures, which can be used to add lighting effect to the rendered volume.

To ease the process of low-level WebGL programming, we utilized the OpenGL Shadering Language (GLSL) [43] for GPU programming. This approach enables us to achieve a high-performance rendering of 3D data. In the following we will describe the pipeline of WebGL2 based volume visualization using graphics hardware accelerated raycasting algorithm.

- For every 3D data set or a series of 2D images acquired from a medical scanner, using the method described in subsection 3.4.1 to create a large 2D montage image, which includes all the 2D slice images of the source data. The corresponding imaging information such as the number of images per column, the number of columns, i.e., the number of images per row, the total number of slices, and distance between every two contiguous slices in x, y or z axis direction is stored in a text file.
- Load the 2D montage image to WebGL rendering pipeline using WebGL's image loading method *getImageData()*. At the same time, the text file containing all the related image information generated in the previous step is also loaded into the WebGL's rendering pipeline.
- The loaded big 2D montage image is used to build 3D texture using WebGL2's method *texImage3D()*, mapping 3D data dimension from data space to texture space, which can be used in graphics unit's vertex shader and fragment shader: $[1, h] \times [1, w] \times [1, d] \mapsto [0,1] \times [0,1] \times [0,1]$ (h, w, and d are the volume's height, width and depth respectively). The related image information was also loaded into the WebGL's rendering pipeline.
- Create the vertex and fragment shader objects, then attach the vertex and fragment shader source code to WebGL. Next, compile these two shaders and link the shader programs to the WebGL rendering pipeline.
- Load the 3D texture to the GLSL's vertex shader, compute vertex positions using method *gl_Position* and output the image coordinate *texCoord* to the fragment shader.
- The computing results in vertex shader are output to the GLSL's fragment shader, where the raycasting computation is performed. First, set the original position of the rendering system, such as (0.0, 0.0, 2.0, 1.0). Next, get the 3D image coordinate from vertex shader, and create casting ray direction, i.e., from the eye (camera) to the 3D image. For each casting ray, acquiring texture information at each equally sampled point along the casting ray inside the 3D texture, and then map the intensity value to color and opacity using loaded 2D texture generated by the transfer function. Our post color-attenuated classification algorithm [44] was used to build the transfer function's lookup table for optical mapping:

$$\alpha^{\psi}(s_f, s_b, d) = 1 - \frac{1}{2}\alpha(s_f, s_b, d) \tag{7}$$

$$\tilde{C}^{\psi}(s_f, s_b, d) = \frac{d}{s_b - s_f} \times \left(\int_0^{s_b} \tau(\omega)c(\omega)d\omega - \int_0^{s_f} \tau(\omega)c(\omega)d\omega \right) \tag{8}$$

where $\tilde{C}^{\psi}(s_f, s_b, d)$ and $\alpha^{\psi}(s_f, s_b, d)$ are color and opacity mapping for ray segment between s_f and s_b with the segment length d, and $c(\omega)$ and $\tau(\omega)$ are color and light attenuation function. The color and opacity mapping user interface and 3D rendering result are illustrated in Fig. 2.

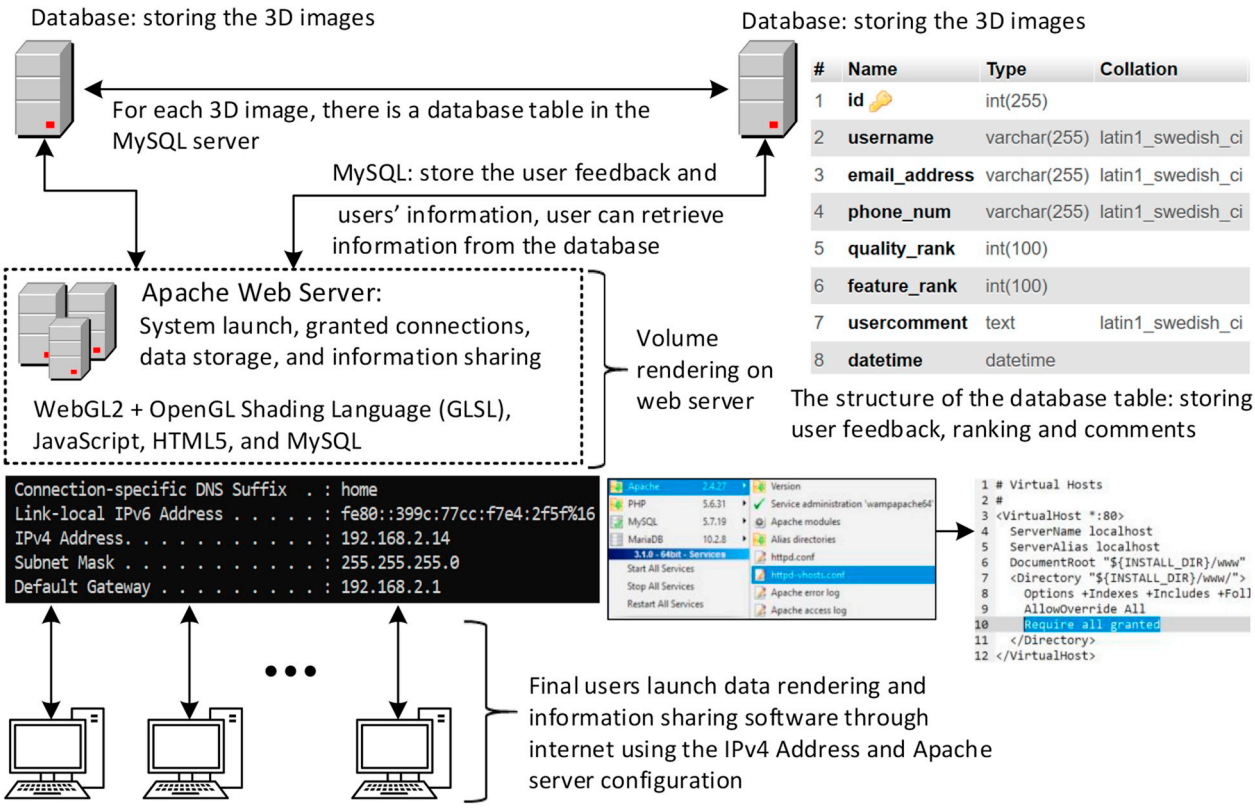
- During the calor and opacity calculation and mapping process, the data of ambient light, specular color, normal at each sampling point are utilized to add shading effect to the final 3D rendering result.
- The formula 6 is used to compose the mapped color and opacity at each sampling point. The raycasting calculation will be conducted at each casting ray in parallel on graphics hardware, i.e., GPU's vertex shader and fragment shader, and will be terminated at any casting ray if the next sampling point is outside the volume boundary or the accumulated opacity exceeds the preset threshold on the casting ray, i.e., early ray termination.
- Finally, the fragment shader computes the final color for each pixel of the primitive, and draws it on the output image inside the web browsers as the volume rendering result.

4. Network setup

4.1. System architecture and data management

The system was implemented using web-based tools to address data rendering and information sharing on internet. As described in Fig. 3, the client and server based system architecture enables us to provide an easily accessible tool for data exploration over the web, which allows users to simply open a web browser to start a data rendering and collaborative analysis. To setup the network of this system, the WampServer [45] was used to create the system architecture, including configuring software packages such as Apache [39], PHP and MySQL database. WampServer is a Windows web development environment, which allows users to create HTML5 based web applications with Apache, PHP and a MySQL database.

A main computer running Apache HTTP Server and Windows 10 was used as a server for web engine and data management, which could be requested by client computers for granted connections, information sharing and launching data visualization software. The server side of the system running Apache was implemented with PHP and MySQL for data storage and information communication management, while WebGL2, OpenGL Shading Language (GLSL), and JavaScript are executed client side on the browser for data real-time visualization as well as information input and extraction. In this system, the communication between the server and clients was done with the PHP, Ajax [46], jQuery, and JSON (JavaScript Object Notation). Ajax is a set of web development techniques, which use many Web technologies on the client side to create asynchronous Web applications.



#	Name	Type	Collation
1	id	int(255)	
2	username	varchar(255)	latin1_swedish_ci
3	email_address	varchar(255)	latin1_swedish_ci
4	phone_num	varchar(255)	latin1_swedish_ci
5	quality_rank	int(100)	
6	feature_rank	int(100)	
7	usercomment	text	latin1_swedish_ci
8	datetime	datetime	

```
# Virtual Hosts
#
<VirtualHost *:80>
  ServerName localhost
  ServerAlias localhost
  DocumentRoot "${INSTALL_DIR}/www"
  <Directory "${INSTALL_DIR}/www/">
    Options +Indexes +Includes +Foll
    AllowOverride All
    Require all granted
  </Directory>
</VirtualHost>
```

Fig. 3. Flowchart of the system network architecture, data rendering and exploration engine, and database management: internet-based data visualization, manipulation, management, and storage, information sharing, network permission granting and connections, as well as 3D medical data exploration using WebGL2, OpenGL Shading Language (GLSL), Apache HTTP Server, HTML5, JavaScript, and MySQL database.

The phpMyAdmin [47] was employed for database management, which is one of the most popular PHP applications and MySQL administration tools. It is a free and open source tool written in PHP intended to handle the administration of MySQL or MariaDB [48] with the use of a web browser. In this platform, system administrators can use the administrative interface of phpMyAdmin to manage MySQL database: creating, modifying or deleting databases, tables, fields or rows; executing SQL statements; and granting users' privileges and permissions. Through using SQL script programming language, the system administrators can add, remove, edit, backup and view databases. The right side of Fig. 3 describes the structure of the database table: storing user feedback, ranking and comments for a specific 3D data rendering on the client side web page, as well as granting system connections through configuring phpMyAdmin's configuration file, i.e., httpd-vhosts.conf. On the client side of this system, all data sets are rendered inside modern web browsers with the help of the client computer's GPU using WebGL2, GLSL, HTML5 and javaScript.

4.2. Network connection

As shown in the left side of Fig. 3, the main volume rendering and data manipulation engine was set on the Apache web sever. The client computers can connect with the web sever using the IP address of the sever computer. When the address was set, the end users can access this address and launch the web sever's data exploration software. For Apache server, the system should change the vhosts configuration file, granting the end users' access to the web server, i.e., modifying "require local" command to "require all granted" command in this configuration file, so the clients from anywhere on internet can access the Apache server through using verified user name and password granted by the system administrators. The end users can use web browser such as Firefox and Chrome to render the selected 3D data using the IP address (192.168.2.14 in our experiment) and the project location on the server (vr_proj/web_render/viscomput_proj/webgl22/in this experiment).

To access the web based visualization software in public domain, the public IP of the network should be acquired first. The router's login page is usually accessed via a private IP, and the router's admin interface shouldn't be made available to the public internet. The following steps are the procedure of creating public domain for our software system running on server and how can it be accessed from public internet.

- First, find out the networking's router's public IP address.
- Second, setup port forwarding on the router. The connections made to the routers public IP will be forwarded to the Apache HTTP server. For example, if the router's public IP is 162.31.30.56 and the server PC's private IP is 192.168.0.100. We need to configure a rule which says that any connection made to 162.31.30.56:80 needs to be forwarded to 192.168.0.100:80.
- Finally, setup a domain pointing to the public IP with a DNS (Domain Name System) registrar and configure an Apache virtual host for that domain.

Through using the above internet setting, users can request and explore medical data and send back feedback to web server from client computers on internet. Users enter a Uniform Resource Locator (URL) to point to a web server by means of its Fully Qualified Domain Name (FQDN) and a path to the required resource through entering granted user name and password, and then the software for data exploration can be launched and connections will be created for data communication between the web server and the granted client computers on public internet.

5. Information exchange and sharing

5.1. User input and data update

In this web based data exploration and information sharing system, MySQL was used to build database to accept user input and store data. MySQL is a database system commonly used on the web, which can run on an Apache HTTP server. The data in a MySQL database are stored in tables. A table is a collection of related data, which consists of columns and rows. For each data set in our visualization system, an independent database table was created to store and manage the information such as user input, user information, was well as login name and password.

The name of the built database is vr_db, which includes multiple tables. As illustrated in Fig. 3, there are three tables in the experiment, i.e., *tablei*, $i = 1,2,3$, which are used to store user feedback for three independent 3D data sets. In addition, there is another table with the name *user_info* to store user name and password information. New tables can be added to the database when new 3D medical images are added to the system. The existing tables will be deleted when the corresponding 3D images are deleted from the system. As demonstrated in the top left of Fig. 4, all the tables (*tablei*, $i = 1,2,3$) have the same structure, including id, user name, email address, phone number, data quality ranking, feature ranking, user comments on the data, as well as the date and time of the user's input.

For every loaded 3D medical data set, the graphical user interface of the system has a corresponding user input interface, such as name, email address, phone number, quality ranking, feature ranking, and comments. Whenever users login the network system using granted user name and password, they can input all these information except the user id and date and time, which are generated by the system automatically. The users can explore the 3D image in web browser, manipulate the rendered 3D image, comment on and rank the selected image, as well as store all the feedback information into the corresponding table inside the MySQL database vr_db on the main HTTP Apache server.

When users click the submit button, the system will call PHP program with the name "userInfor.php". In the PHP program, the system first builds links to the Apache sever, then login with user name and password using MySQL link function *mysqli*. Each data set in the system has a unique table to store user's input data and information. When the login and connections are successful, the software use *$_POS* method to acquire users' input data and information, and then insert all these information input the database table corresponding to the selected 3D data. The user id and date and time are generated automatically. The user id is used for management, which can only be accessed by the database administrator.

The server will store the user input into the database table if the client connection and data transaction are successful. The other users on any client computers with granted connections to this system can retrieve the stored information out and review the feedback. In the next step, jQuery was used to build a serialize array for data update in the database table, send the message of "success" or "failure" to the system's user interface, which will be displayed at the location close to the submission feedback. Finally, the input information will also be cleaned by using a jQuery function *clearInput*. The whole interface of user input and data update are described in Fig. 4.

5.2. Information extraction

The Ajax (Asynchronous JavaScript And XML) was used to retrieve data from the database table, which requires a PHP file for server connection and database content extraction. For the PHP program, first, build a connection with the database vr_db using PI address, granted user name and password, and then retrieve data from the table, such as Table 1. Finally, all the rows of the query results are stored in an array, which will be returned by a encoding method of JSON.

The queried data are input into a Ajax's asynchronous function, and then the Ajax sends the output to the PHP file for database information extraction from a running Apache server. The connection and information sharing are asynchronous, but the user input information can be displayed to all the other users on networking who are using the same visualization system. The feedback information can be displayed

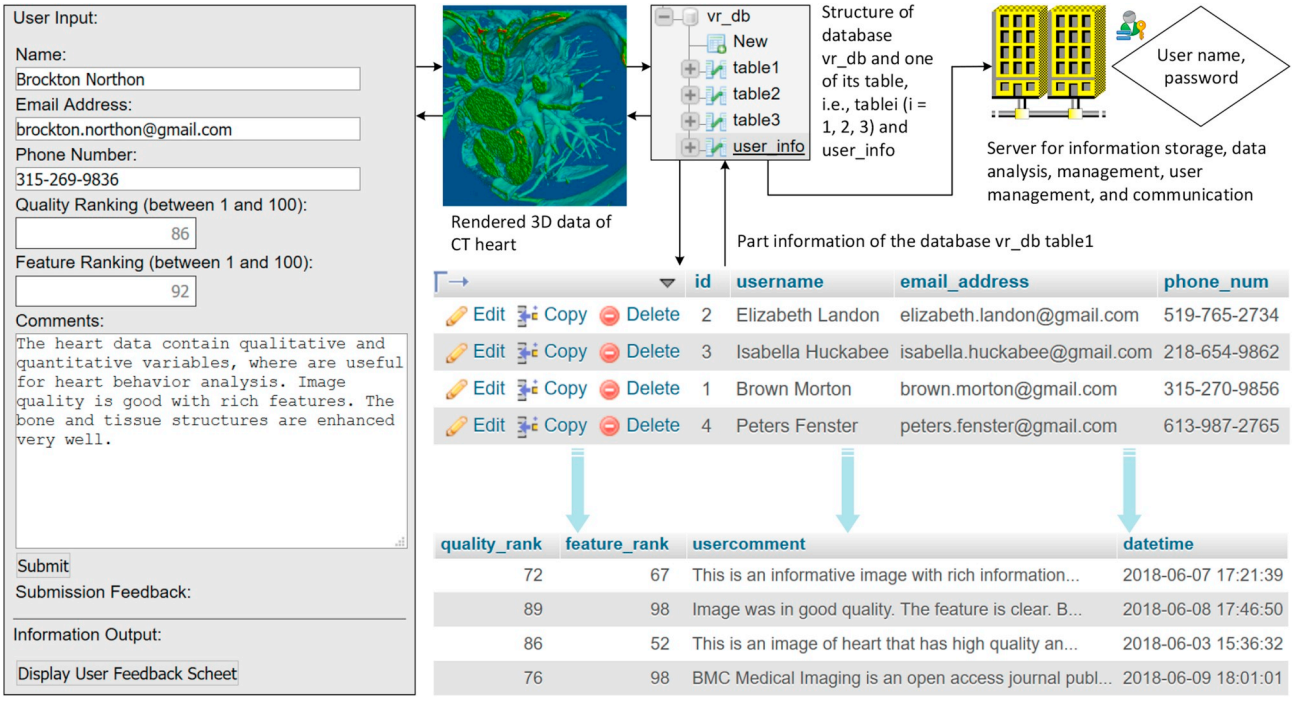


Fig. 4. Data storage and communication between the data exploration platform and MySQL database on Apache HTTP Server. User's input and feedback can be used to update the database table in real time. The information can be shared with multiple users on network who are using the same data visualization and information sharing system.

Table 1
Hardware configurations used in the experiment for data rendering. Four systems, i.e., system i (i = 1, 2, 3, 4), are listed and used to test our software system's performance.

System number	Central processing unit (CPU)	Memory DDR3(GB)	Graphics card (memory)
1	Dual Intel Xeon X5690	96	GTX 480 (1.5GB)
2	Intel Core i7-3770K	32	GTX 480 (1.5GB)
3	Dual Intel Xeon X5690	96	GTX 980 (4.0GB)
4	Intel Core i7-3770K	32	GTX 980 (4.0GB)

in a independent web page, which has an organized format designed with HTML forms and tables.

6. Performance evaluation

The Apache server currently runs on two hardware systems, the first one is a workstation with dual processor, dual Intel Xeon X5690 (Westmere-EP, 3.47GHz, 130W Six-Core Processor), 96GB DDR3 memory (12 × 8 GB), and a Crucial MX300 1TB Internal SSD hard drive; while the second one is a computer with Intel Core i7-3770K Ivy Bridge Quad-Core 3.5GHz (3.9GHz Turbo), 32GB DDR3 memory (4 × 8 GB), and a Crucial MX300 1TB Internal SSD hard drive. Two graphics cards are used interactively, the first one is Nvidia GeForce GTX 980 4GB GDDR5 and the second one is Nvidia GeForce GTX 480 1.5GB GDDR5.

The hardware configurations provides enough computing power for our current purpose and future extensions like running data analysis algorithms on the server and distributing the results to all clients. Nvidia graphics driver 398.36 for windows 10 64 bit was used to test the system's performance.

6.1. User interface

The user interface of this network based medical data rendering and information sharing system is intuitive and allows the user to easily manipulate data rendering, tissue feature extraction and enhancement, data extraction from the system's database, and share information with other users. Fig. 5 demonstrates the system interface of WebGL based volume rendering of 3D medical data in Firefox web browser: the right image is the whole interface of a volume rendered whole 3D CT heart data with surrounding tissues and bones, while the left image only includes the displayed 3D medical image of a clipped heart data. The client computers can link to the Apache server with IP address 192.168.2.14 to launch the visualization software. The system's location on the main server is vr_proj/web_render/viscomput_proj/webgl22/.

Inside the user interface, users can select the data set rendered, interactively manipulate the transfer function editor through adjusting the control points, i.e., green points, using mouse, and the color mapping function can be selected and adjusted. The manipulation of the transfer function editor and color mapping function can interactively change the appearance and color of the rendered 3D data, helping users to capture the volume and feature of interest inside the rendered 3D medical data. In addition, the number samples along each casting ray can also be adjusted to balance the image quality and volume rendering speed on the web browser. The users can also add lighting effect to the rendering results, enhancing the reality of the visualized volumetric images.

As shown in the “user input” part of Fig. 5, user can input the user name, email address, phone number, quality ranking (between 1 and 100), feature ranking (between 1 and 100), and leave comments on the rendered image. When the user finishes the input, he/she can click the “submit” button to send the input information to the database on server through internet connection. The “Submission Feedback” will display the result of the submission, such as “success” or “failure”. The user can also click the button “Display User Feedback Sheet”, an independent user feedback sheet will be displayed in an independent web page with a data table including user name, email, phone number, quality ranking, feature ranking, user comments, and time of the input, which is illustrated in Fig. 6.

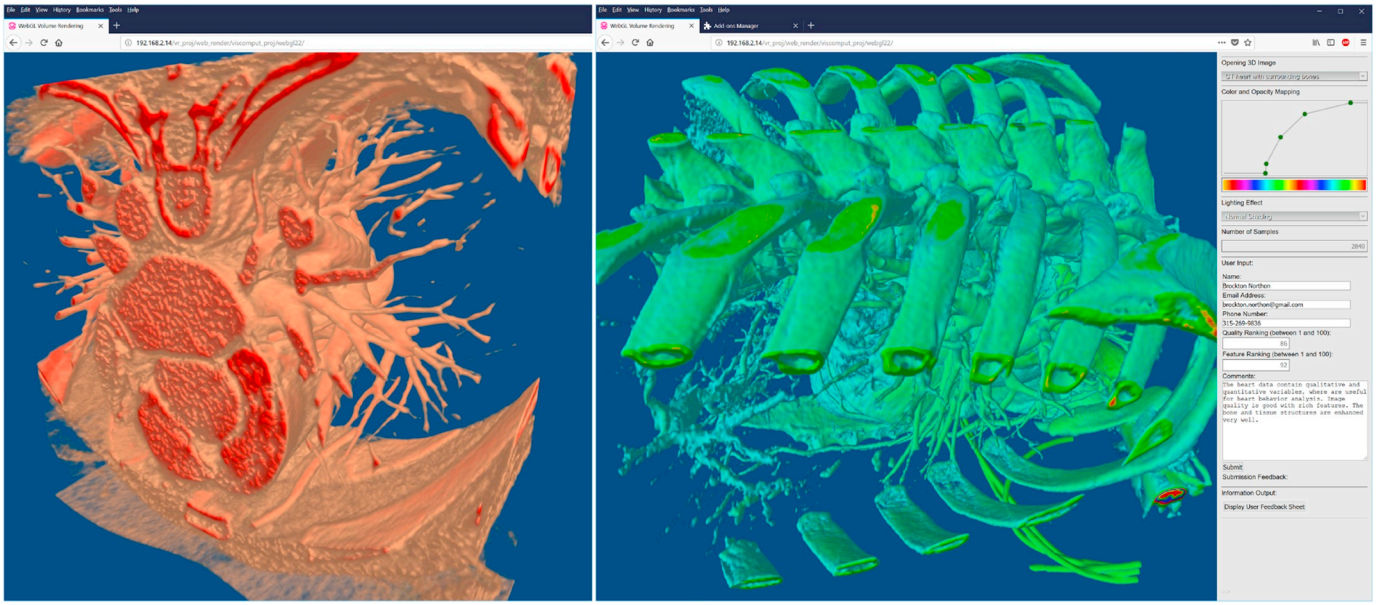


Fig. 5. The whole interface of the WebGL2 based volume rendering of 3D medical data in Firefox web browser. User can input ranks and leave feedback to the rendered image and extract feedback report of all the users on this image from database on the server. The left side snapshot is a rendered clipped heart data, while the right side snapshot is a visualized whole CT heart data with bones.

Name	Email	Phone #	Q	F	User Comments	Time
Elizabeth Landon	elizabeth.landon@gmail.com	519-765-2734	72	67	This is an informative image with rich information for interactive diagnosis. The left side of the data can be visualized easily.	2018-06-06 17:21:39
Isabella Huckabee	isabella.huckabee@gmail.com	218-654-9862	89	98	Image was in good quality. The feature is clear. But I believe this is a heart image with weak left side muscle.	2018-06-10 17:46:50
Connor Graeme	connorgraeme@gmail.com	876-0912-765	34	54	This image is formed on the image plane of the camera and then measured electronically or chemically to produce the photograph.	2018-06-18 20:16:52
Brockton Northon	brockton.northon@gmail.com	315-269-9836	86	92	The heart data contain qualitative and quantitative variables, where are useful for heart behavior analysis. Image quality is good with rich features. The bone and tissue structures are enhanced very well.	2018-06-25 15:42:31

Acronym Note: Q: Quality Ranking; F: Feature Ranking

Fig. 6. Part of the web based table generated by the system's information extraction: the user name, email address, phone number, quality ranking, feature ranking, comments, and the time that the user input the information are displayed in the table on an independent web page.

6.2. Simulated medical study

Twenty two users evaluated this web-based medical information and data visualization system through entering rankings and comments on the right cardiac image of Fig. 5. The users input their names, email addresses and phone numbers (not real information used for privacy consideration), which will be used for the second step communication and diagnosis between users. The users ranked the image quality and information features using the 100 point based system. When they viewed the rendered cardiac data, users could get the patient's information from our server, which described the patient's cardiac condition. Then the users explored the 3D cardiac data on web browser through internet connection. In their comments, they added their observations and diagnostic options based on the data visualization and exploration. The patients's heart conditions usually can be identified in their medical images in 3D, such as coronary heart disease, unstable angina, heart attack, heart failure, valve disease, and congenital heart conditions. Usually, higher ranking images will give higher weight in the users' disease diagnosis comments.

The users' input information can be extracted for medical data analysis and distributed diagnosis. Fig. 6 illustrates part of the output information sheet of users' input information, comments, rankings and feedback. Through using phpMyAdmin running on the main Apache server, the system administrator can visualize and analyze the users' input and rankings, and can visually output the result in a graph format demonstrated in Fig. 7, which can help disease diagnosis through collecting diagnosis information from internet.

6.3. Visual evaluation

In this section the visual appearance of the volume rendered medical images will be evaluated. Effectively visualizing the medical

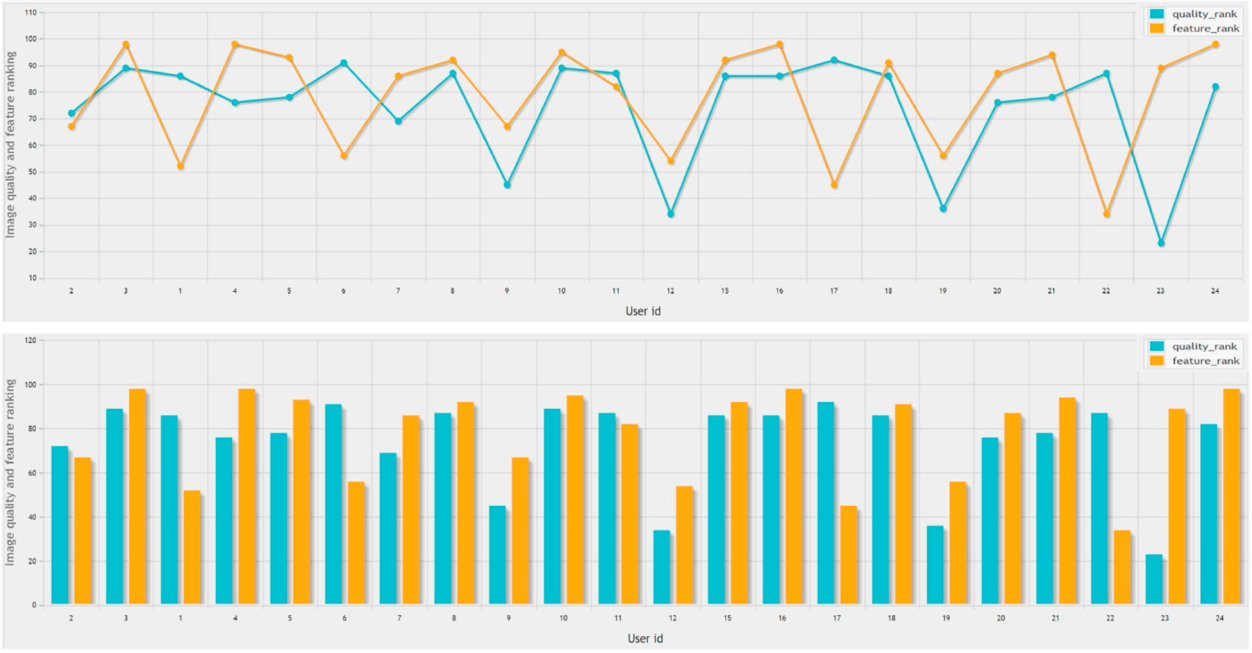


Fig. 7. Visualization and analysis of users' ranking for a specific 3D medical data rendering on a web browser, including image quality and feature. The ranking of all the user ids are displayed on two graph tables, the top one is the line links of users' ranking, while the bottom one is the column of the users' ranking, which will give the analyst different perspective of the data information.

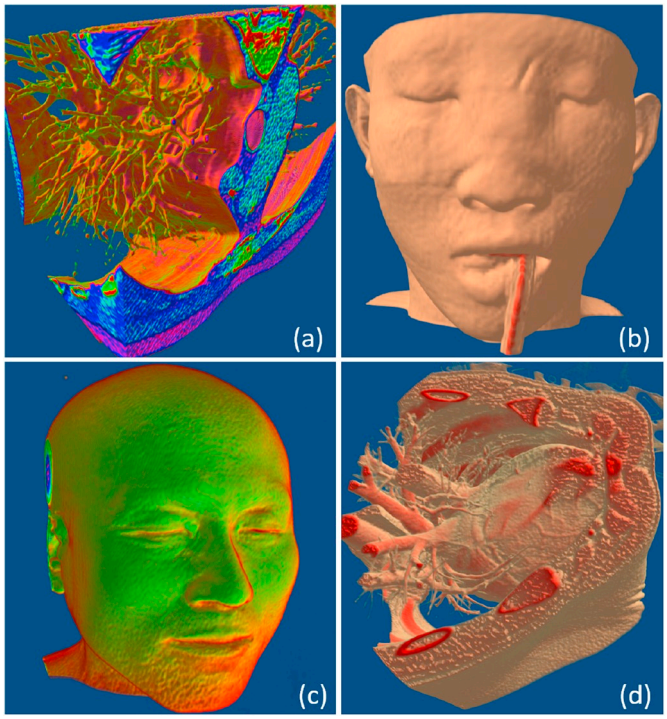


Fig. 8. The rendered 3D medical images using our new raycasting algorithm running graphics hardware's fragment shader and vertex shader. Our post color attenuated voxel classification algorithm and lighting effect were used to deliver realistic high image quality and interactive classification speed. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

volumes without artifacts and in a manner which allows the data to be understood correctly are paramount considerations in medical fields. Fig. 8 illustrates the rendered 3D medical images using our new raycasting algorithm running graphics hardware's fragment shader and vertex shader using GLSL programming language. Our unique post color attenuated voxel classification algorithm was used to build the transfer functions, which can seamlessly work with raycasting algorithm and deliver high image quality and interactive voxel classification speed on web browsers.

Fig. 8(a) is a clipped heart CT image with bones, and the vessels and heart are enhanced with red and green colors, while (d) is MR data of heart inside the chest, we can see that the heart with vessels and the chest with bones are extracted and emphasized with skin and red colors. The image (b) and (c) are brain images, (b) is rendered CT data with relatively low image quality displayed with skin color, while (c) is high quality MR data, which is visualized with green color with red boundaries and enhancement. The overall appearance of the volumes can satisfy the requirement of medical research and normal clinical applications such as distributed diagnosis on internet using 3D medical data exploration.

When using the post color-attenuated classification algorithm with raycasting method running on GPU shaders of WebGL2, the medical data features of interest can be effectively captured and enhanced. The four images in Fig. 9 demonstrate the experimental results of visualizing a dataset of heart in chest. Through adjusting the transfer function, our post color-attenuated classification algorithm can interactively update color and opacity of a volume rendered heart data with surrounding tissues and bone. Different organ structures can be visualized and enhanced through using different opacity settings. Fig. 9(a) is the whole heart with surrounding tissues, and (b) displayed part of bones inside the muscles. Image (c) extracted heart organ and bone very well, and only most of the bone is displayed (d).

Our algorithm can keep the image quality and data information well, as shown in Fig. 10, this system can effectively extract tissue

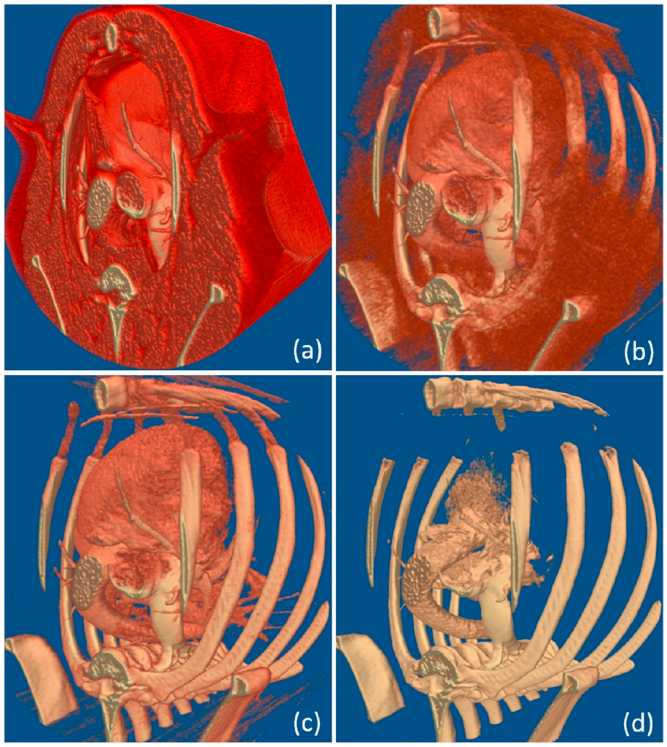


Fig. 9. Different organ structures can be effectively visualized and enhanced using our post color-attenuated classification algorithm: from the whole heart with surrounding tissues (a), to image with heart extracted from surrounding tissues with different extent described in the images (b) and (c) respectively, and to only most of the bone being displayed (d). (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

information and keep high image quality when zoom in the rendered image in web browser. The zoom in part (left image of Fig. 10) of the right image of Fig. 5 clearly show the vessels with high quality, while the right image displays the enlarged part of Fig. 9 (d), which keeps the bone structures well with super image quality.

6.4. Rendering speed evaluation

This section will describe the experimental results of running the software system on four hardware systems listed in Table 1 and five datasets illustrated in Table 2. We use a Dell UltraSharp U3014 monitor

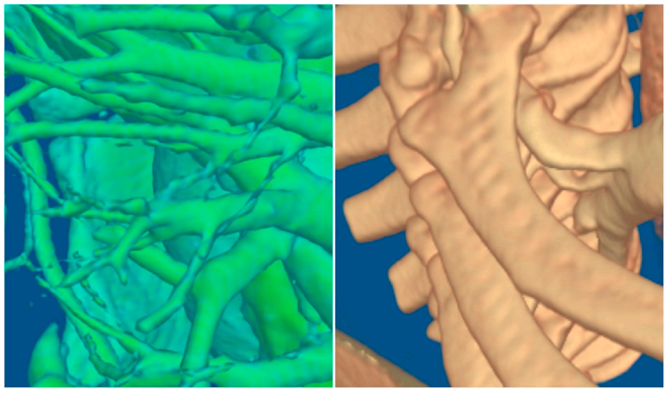


Fig. 10. Volume part captured from the zoom in rendered images: the left one is a zoom in part of the right image of Fig. 5, while the right one is the zoom in part of Fig. 9 (d). The zoomed in images can keep in the same high data rendering quality as the original image and keep all the data details.

Table 2
Datasets used in the rendering experiment, including data number, visualized image in the figures of subsections 6.1 and 6.3, number of slices, original data size of raw data format in megabyte (MB), and the extracted data size in MB.

Data number	Visualized figure	Number of slices	Original data size	Extracted data size
1	Fig. 9	84	187	1.78
2	Fig. 5 (right)	280	21.4	1.52
3	Fig. 8 (a)	100	64	3.43
4	Fig. 8 (c)	176	22	1.71
5	Fig. 8 (d)	122	72.5	2.80

with resolution 2560 × 1600 to display all these datasets in full screen. Mozilla Firefox Quantum 61.01 64-bit and Google Chrome Version 67.0 (Official Build) (64-bit) are employed to run the 3D datasets on host and client computers. To calculate the rendering speed, for each hardware system, we use the average rendering speed of every 5 seconds as one test for each dataset. To get stable test results, we conduct more than 40 tests for every dataset and hardware system in our experiment, and use the tests from 21 to 40 to analyze the system's performance.

The original raw datasets are pressed using the method described in subsection 3.4.1, extracting the 2D images from the 3D raw data and then aligning the 2D slices in a big 2D montage image with one or multiple columns. The distance between the 2D slices are set based on the information of the raw data or meta data. The 2D montage images are then loaded into the web browser to create 3D textures for ray-casting computation on GPU's fragment and vertex shaders, so 3D textures can be generated and used during the volume rendering process.

Table 3 shows the system's performance of rendering the five datasets listed in Table 1 using the four hardware systems described in Table 2. From this table, we can see that using the different central processing unit (CPU) and main memory have less impact on the rendering performance than using different graphics processing unit (GPU), i.e., different generation of graphics cards.

Table 3
System rendering performance of five medical datasets described in subsection 6.1 and 6.3. Four different hardware configurations and Firefox and Chrome web browsers are used in the experiment. Twenty average speeds are selected, and the mean speed of these 20 recorded tests with standard deviation (SD) are also calculated.

Hardware	Browser	Data	Data rendering performance tests: frames per second (fps)																			Static analysis
system		number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Mean ± SD
System 1	Firefox	1	60	32	59	54	26	49	59	55	58	20	56	57	54	54	56	41	39	56	55	58	49.9 ± 11.8
		2	28	9	35	31	34	34	35	29	34	34	35	31	34	34	34	30	34	33	35	32	31.8 ± 5.7
		3	47	18	44	19	46	53	43	17	41	53	42	42	47	51	41	39	41	53	42	42	41.1 ± 10.9
		4	40	47	49	40	9	40	42	49	39	37	43	53	39	40	43	5	40	39	6	40	37.0 ± 13.7
		5	43	44	38	59	43	44	43	59	43	45	43	59	42	43	43	59	42	45	44	59	47.0 ± 7.2
	Chrome	1	40	57	55	59	60	57	57	59	48	57	57	47	40	57	45	59	60	53	57	59	54.1 ± 6.5
		2	34	35	32	39	36	36	35	39	32	36	37	40	36	35	33	40	36	36	37	40	36.2 ± 2.5
		3	52	42	46	52	46	42	46	52	53	42	42	52	52	42	46	52	39	41	46	52	46.9 ± 4.8
		4	43	42	47	51	43	39	51	51	43	42	51	51	43	42	47	51	42	42	45	43	45.5 ± 4.1
		5	38	33	49	22	42	45	54	47	42	45	54	51	43	44	54	50	42	45	54	49	45.1 ± 7.9
System 2	Firefox	1	45	55	30	52	43	46	55	51	52	31	54	46	57	50	58	46	57	50	42	21	47.1 ± 9.9
		2	34	30	33	31	34	29	34	21	33	32	33	32	34	31	33	32	33	30	34	31	31.7 ± 2.9
		3	38	43	38	46	40	47	37	37	39	34	36	47	39	46	38	18	28	38	13	39	37.1 ± 8.7
		4	42	32	42	46	42	45	43	30	21	19	49	41	23	41	42	41	48	38	44	19	37.4 ± 9.8
		5	51	57	16	47	53	34	41	46	53	57	41	47	53	31	47	46	20	39	45	60	44.2 ± 11.7
	Chrome	1	45	49	49	45	49	49	49	44	49	49	46	42	42	49	29	49	49	45	49	49	46.3 ± 4.8
		2	42	42	43	43	43	37	43	42	38	40	43	38	38	43	42	40	40	42	39	42	41.0 ± 2.1
		3	42	46	39	46	44	44	44	46	46	45	46	40	43	45	39	46	45	45	44	43	43.9 ± 2.3
		4	44	41	43	46	44	42	44	48	40	53	41	53	37	53	44	42	44	50	44	43	44.8 ± 4.5
		5	17	46	55	51	33	38	48	52	43	46	54	49	44	46	55	52	44	46	40	52	45.6 ± 8.9
System 3	Firefox	1	60	60	60	60	54	60	60	60	60	60	48	53	60	60	60	60	50	53	60	60	57.9 ± 3.9
		2	60	60	60	59	57	60	60	60	60	58	60	59	60	60	60	60	59	35	60	60	58.4 ± 5.6
		3	57	60	60	60	46	60	60	60	60	60	40	60	60	60	60	49	55	60	60	60	57.6 ± 5.7
		4	52	60	60	60	60	59	57	60	60	60	60	60	60	60	60	59	46	60	60	60	58.7 ± 3.5
		5	60	19	49	60	55	60	51	60	60	60	37	59	51	59	60	60	54	59	60	58	54.6 ± 10.2
	Chrome	1	60	60	58	60	60	50	57	60	60	60	60	46	59	60	60	60	60	60	60	60	58.5 ± 3.7
		2	60	57	56	60	60	60	60	54	57	48	45	60	60	57	60	55	60	60	60	60	57.5 ± 4.2
		3	60	60	52	59	60	60	60	54	60	60	60	59	57	60	58	60	60	58	60	60	58.9 ± 2.2
		4	60	60	52	52	60	60	60	60	52	53	60	60	60	60	58	60	60	60	60	54	58.1 ± 3.3
		5	60	60	55	56	57	60	60	60	56	58	55	45	60	60	60	60	60	49	60	60	57.6 ± 4.1
System 4	Firefox	1	60	60	60	60	53	60	60	60	60	60	60	60	59	57	60	60	60	60	57	60	59.3 ± 1.8
		2	58	60	57	60	60	60	60	58	58	57	60	60	60	60	60	59	60	35	47	60	57.5 ± 6.1
		3	60	54	60	55	60	60	58	60	60	60	60	49	59	60	59	59	60	60	60	60	58.7 ± 2.8
		4	46	48	60	60	60	55	60	60	60	58	57	60	60	24	48	60	59	52	60	60	55.4 ± 8.7
		5	56	60	56	58	60	60	60	36	60	60	58	60	60	60	24	50	60	51	60	60	55.5 ± 9.4
	Chrome	1	60	60	50	60	60	60	60	60	60	60	55	35	60	60	60	60	59	60	60	45	57.2 ± 6.6
		2	60	60	60	60	59	60	57	57	60	60	60	60	55	60	57	60	60	60	59	58	59.1 ± 1.5
		3	60	60	58	60	60	59	60	60	60	60	58	60	60	58	57	56	60	60	60	60	59.3 ± 1.2
		4	60	60	60	58	58	60	60	60	58	60	60	60	60	55	60	60	60	53	60	60	59.1 ± 1.9
		5	57	60	60	60	60	57	60	49	60	60	60	60	53	51	56	28	45	60	60	60	55.8 ± 7.9

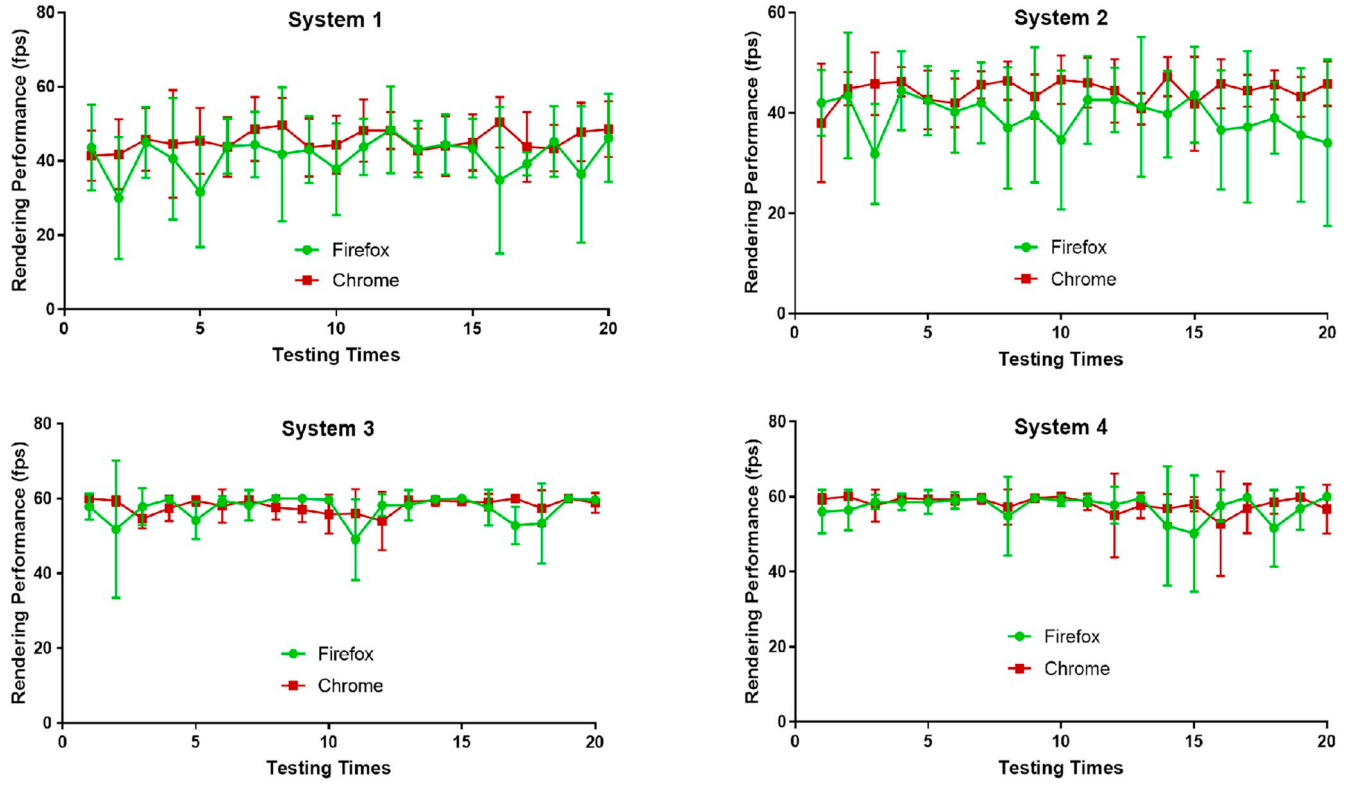


Fig. 11. Rendering performance calculated with frame per second (fps) using the five datasets listed in Tables 2 and 3. Web browsers Firefox and Chrome are used in the experiment. Four hardware configurations are employed in the test as demonstrated in the four figures titled with System i (i = 1, 2, 3, 4) respectively.

When using GTX 480 graphics card and Firefox web browser, the system 1, i.e., dual Intel Xeon X5690 with 96GB memory can deliver mean rendering speed with standard deviation (SD) 41.4 ± 9.9 frames per second (fps), while the system 2, i.e., Intel i7-3770K with 32GB memory can achieve 39.5 ± 8.6. When using the same hardware configuration of systems 1 and 2, the speeds of Chrome are 45.6 ± 5.2 fps and 44.3 ± 4.5 fps respectively. For both web browsers, the dual Intel Xeon can only increase ~1 fps average speed with similar standard deviations. However, Chrome browser can deliver ~11% better performance when using these two systems than Firefox, and the speed standard deviation when using Chrome is ~9% smaller than that of using Firefox, which means that when using the same hardware configurations, Chrome can deliver better unified rendering performance than that of Firefox.

At the same time, when using GTX 980 graphics card with the same above hardware configure, i.e., systems 3 and 4 listed in Table 1, there is great rendering speed enhancement, i.e., Firefox web browser can deliver rendering speeds of 57.4 ± 5.8 fps and 57.3 ± 5.8 fps respectively, which are 38.65% and 45.06% performance enhancement, and the average speed standard deviation deceases 37.3%. The web browser Chrome's corresponding performance are 58.1 ± 3.5 fps and 58.1 ± 3.8 fps, which are 27.41% and 31.15% speed increase, and the average performance standard deviation deceases 31.96%. We can see that both browsers have the similar performance patten, i.e, delivering the similar performance when using the same GPU and different CPU and main memory, and we can also notice that Chrome can achieve a little bit better performance of ~1 fps with less variation of ~2 fps. The performance advantage of Chrome is much less when using newer graphics card than that of employing older graphics hardware. The comparison results are visualized in the left figure in Fig. 12.

As shown in Table 3 and Fig. 11, when using the older graphics card, i.e., GTX 480, the browser Chrome can deliver better performance than Firefox in the 20 average speed tests and their speed variations are almost the same. However, when using the newer graphics card, i.e., GTX 980, the average speed performance of the two web browsers is similar, but the speed average deviation of Chrome is smaller than that of Firefox. We also notice that in the 20 average speed tests, some tests have large speed changes, i.e., during the rendering process, at some time points, the browser's rendering speed is much slower than the rest. Based on the fact that the speed variance when using GTX 980 is much smaller than that of using GTX 480, we believe that the reason of the big speed variances maybe the issues of graphics driver and GPU's WebGL2 support and compatibility. We also believe that new graphics drivers and next generation graphics hardware may address this issue effectively. For both web browsers, when using GTX 980, their performance difference is smaller than that of using GTX 480 graphics card, but the speed variance of Chrome is smaller than that of Firefox when using the first two system as shown in Fig. 11. This speed variance trend can also be demonstrated in the right figure of Fig. 12.

As demonstrated in Fig. 11, when using newer graphics hardware, the average rendering speed is higher than 55 fps constantly in all of the 20 tests of our experiment for both web browsers and CPU hardware configurations, i.e., systems 3 and 4 of Table 1. For Firefox, the speed variance range is 1.8 ~ 10.2 fps and the average speed variance is ~5.8 fps, while the speed variance range is 1.2 ~ 7.9 fps for the Chrome web browser and the average speed variance is ~3.7 fps, which is ~56.8% better than that of using Firefox. Therefore, we can see that the performance of Chrome is more uniform than that of Firefox. However, both browsers can achieve real time performance when using the newer graphics hardware, i.e., GTX 980. We can conclude that when using current generation graphics card and commonly available web browsers, our system can deliver real time rendering speed of average size medical datasets. We can also conclude that the system's performance more depends on GPU than CPU and system's memory. Also, the Chrome browser has better WebGL2 based graphics rendering performance than that of Firefox.

In the performance testing experiment, we use a network with average 30 Mbps download speed and 2.5 Mbps download speed, we find that the data download from the Apache server to our client system

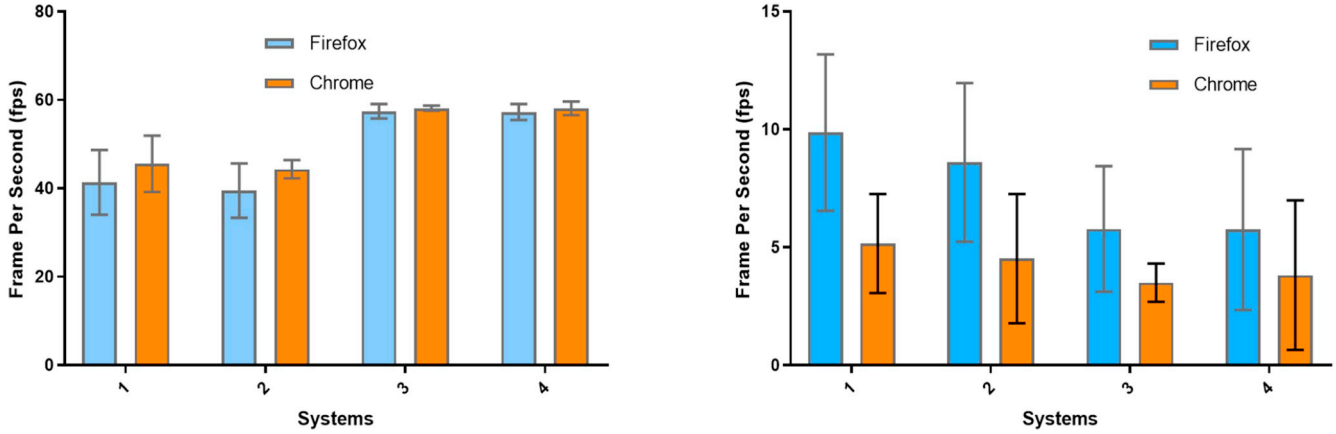


Fig. 12. Mean performance of the 20 average rendering speed tests (left figure) and standard deviation (right figure) using the 5 datasets listed in Tables 2 and 3. Web browsers Firefox and Chrome, and four hardware systems are employed in the experiment.

is usually less than 2 seconds. All the messages of user input can be uploaded to the Apache server and shared with other users in the network in real time. The Apache sever is used to store our software system and datasets, any users in the network can launch the system and use the medical data for feature detection and data exploration. Also, users can input their feedbacks to the Apache server and all the comments can be visualized in a separate web page for information sharing. Furthermore, the data rendering speed relies on the client computer rather than the host server that runs Apache. We have tested different hardware system as a web server, such as the described four hardware systems listed in Table 1 and a Dell Laptop XPS 15 9550, and find that the difference of the system's performance is very similar.

7. Results and discussion

In this paper, we presented a network based collaborative software platform for exploring medical data and sharing information. In this system, clinical users can visualize medical data in high quality and real time on modern web browsers through internet and explore medical data interactively. The users can also analyze medical data on web server dynamically and extract targeted clinical information for distributed disease diagnosis and treatment.

Taking advantage the 3D texture support of WebGL2, we developed a new raycasting algorithm on GPU vertex and fragment shaders, in which an efficient voxel classification method was integrated in data sampling and color mapping process for high-quality medical data visualization and manipulation on web. The high image quality of data rendering results are demonstrated in Figs. 8 and 9, and our visualized can keep high image quality when being zoomed in to see image details 10. In this HTML5 and WebGL based medical data rendering framework, volume data was first downloaded to the local client computer from the server, where the data was then transformed into textures representing the volume and rendered on screen. As shown in Table 3, the normal size medical data can be rendered in around 60 frame per second using commonly available hardware and internet speed, and the performance standard deviation of our system is very good (illustrated in Fig. 11). In addition, as demonstrated in Fig. 12, this software platform can maintain high performance on various modern browsers, such as Firefox and Chrome. Therefore, we can conclude that the developed software system can satisfy the requirements of common medical data exploration and visualization in clinical research and applications such as distributed disease diagnosis.

As illustrated in Fig. 5, our network based web interface allowed remote access, visualization and manipulation of medical data and sharing information through a dynamic web page, where Apache HTTP Web Server, MySQL database, HTML5, and the server scripting language PHP were utilized to achieve these functions. The users can interactively update server database and the system administer can grant users with different right for database access, manipulation and data extraction. The users' feedback and input can be analyzed and visualized in a web browser interface of phpMyAdmin, and an example result is illustrated in Fig. 7. As demonstrated in Fig. 6, our system enables users at arbitrary locations to explore and analyze medical data, leave comments, update sever's database, as well as extract data from database and display users' information and feedback.

The online nature of our software platform makes it feasible for users spread across the globe work together to explore medical data in real time and interactively share information. This collaborative application will assist users by providing them a “common ground” in the shared workspace. In our platform, users do not have to download and install additional software packages or plug-ins to start data exploration and analysis. Furthermore, real-time high-quality data rendering can be linked to server's database, providing users with intelligent feedback and capabilities for data management and information sharing. For exploring large medical data, we can use software tools such as Microsoft OneDrive, Google Drive or Dropbox to synchronize the data between client and server computers, and then load data from the client computer directly into the software platform, which will decrease the burden of data transmission on network and improve the performance of the system.

In the next step of this project, some advanced features such as integrating doctors’ marks, notes, data mining, machine learning and data segmentation results into this software framework, and display medical data in multiple windows. The synchronization of data navigation will also be developed and included into the system. Currently, due to the data storage limitation of WebGL2, our system cannot load very large medical data for rendering. We believe that this limitation will be addressed successfully in the next version of WebGL and HTML5. We plan to collaborate with local hospitals in the near future and will recruit additional medical users to test and evaluate the system in a real world clinical environment. In our future work, we also will add quantitative rendering quality evaluation when we get more clinical data sets through collaborating with hospitals and medical centers. This network based system can be used as an efficient platform for medical data exploration and information sharing, providing medical users with a solid basis for collaborative disease diagnosis and clinical research.

Conflicts of interest statement

The author declares no conflicts of interest.

Acknowledgments

The author would like to thank the Faculty Startup Funds and Grant for Professional Development of the School of Information Technology

at the Illinois State University. The author also thanks the Publication Incentive Program and Publication Open Access Grant of the Illinois State University as well as the support and collaborations of his colleagues. As an adjunct faculty, the author would appreciate the support of the Department of Medical Biophysics at the Western University for providing digital library access and research collaborations. Finally, the author would like to thank Dr. Peters' lab in the Robarts Research Institutle, Western University for supplying part of the medical data sets used in this research project.

References

[1] de Dumast P, Mirabel C, Cevidanes LHS, Ruellas AC, Yatabe M, Ioshida M, Ribera NT, Michoud L, Gomes LR, Huang C, Zhu H, Muniz L, Shoukri B, Paniagua B, Styner M, Pieper S, Budin F, Vimort J, Prieto J. A web-based system for neural network based classification in temporomandibular joint osteoarthritis. Comput Med Imag Graph 2018;67:45–54. https://doi.org/10.1016/j.compmedimag.2018.04.009https://doi.org/10.1016/j.compmedimag.2018.04.009.

[2] Congote J, Moreno A, Segura A, Beristain A, Posada J, Kabongo L, Ruiz O. Volume ray casting in WebGL. INTECH Open Access Publisher; 2012https://books.google.com/books?id=VWjktAEACAAJ.

[3] Noguera-Rozúa JM, Jiménez-Pérez JR, Ogayar-Anguita CJ, Segura-Sánchez RJ. Volume rendering strategies on mobile devices. 7th international conference on computer graphics theory and applications (GRAPP), rome (Italy), 24-26 February. 2012. p. 447–52.

[4] Pienaar R, Rannou N, Bernal J, HÃ¤hn D, Grant PE. Chris- a web-based neuroimaging and informatics system for collecting, organizing, processing, visualizing and sharing of medical data. 2015 37th annual international conference of the IEEE engineering in medicine and biology society EMBC; 2015. p. 206–9. https://doi.org/10.1109/EMBC.2015.7318336.

[5] Levoy M. Display of surfaces from volume data. IEEE Comput Graph Appl Mag 1988;8(3):29–37. https://doi.org/10.1109/38.511.

[6] Lichtenbelt B, Crane R, Naqvi S. Introduction to volume rendering. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.; 1998.

[7] Li L, Peng H, Chen X, Cheng J, Gao D. Visualization of boundaries in volumetric data sets through a what material you pick is what boundary you see approach. Comput Methods Progr Biomed 2016;126:76–88https://doi.org/10.1016/j.cmpb.2015.11.014https://doi.org/10.1016/j.cmpb.2015.11.014.

[8] Virag I, Stoicu-Tivadar L. A survey of web based medical imaging applications. Acta Electrotehnica 2015(3):365–8.

[9] Noguera JM, Jiménez JR. Mobile volume rendering: past, present and future. IEEE Trans Visual Comput Graph 2016;22(2):1164–78.

[10] E. Goceri, Future healthcare: Will digital data lead to better care?, New Trends and Issues Proceedings on Advances in Pure and Applied Sciencesdoi:07.10.18844/gjapas.v0i8.2781. URL https://sproc.org/ojs/index.php/paas/article/view/2781.

[11] Elhoseny M, Ramírez-González G, Abu-Elnasr OM, Shawkat SA, A. N, Farouk A. Secure medical data transmission model for iot-based healthcare systems. IEEE Access 2018;6:20596–608. https://doi.org/10.1109/ACCESS.2018.2817615.

[12] Congote JE, Novo E, Kabongo L, Ginsburg D, Gerhard S, Pienaar R, Ruiz OE. Real-time volume rendering and tractography visualization on the web. J WSCG 2012;20(2):81–8.

[13] Mobeen MM, Feng L. High-performance volume rendering on the ubiquitous webgl platform. High performance computing and communication & IEEE international conference on embedded software and systems, IEEE international conference on (HPCC-ICESS), vol 00. 2012. p. 381–8.

[14] Mahmoudi SE, Akhondi-Asl A, Rahmani R, Faghih-Roohi S, Taimouri V, Sabouri A, Soltanian-Zadeh H. Web-based interactive 2d/3d medical image processing and visualization software. Comput Methods Progr Biomed 2010;98(2):172–82. https://doi.org/10.1016/j.cmpb.2009.11.012https://doi.org/10.1016/j.cmpb.2009.11.012.

[15] Marion C, Jomier J. Real-time collaborative scientific webgl visualization with websocket. Proceedings of the 17th international conference on 3D web technology, Web3D'12. New York, NY, USA: ACM; 2012. p. 47–50.

[16] Kaspar M, Parsad NM, Silverstein JC. An optimized web-based approach for collaborative stereoscopic medical visualization. J Am Med Inf Assoc 2013;20(3):535–43.

[17] Rego N, Koes D. 3Dmol.js: molecular visualization with WebGL. Bioinformatics 2015;31(8):1322–4.

[18] Jaworski N, Iwaniec M, Lobur M. Composite materials microlevel structure models visualization distributed subsystem based on webgl. 2016 XII international conference on perspective technologies and methods in MEMS design MEMSTECH); 2016. p. 22–4. https://doi.org/10.1109/MEMSTECH.2016.7507511.

[19] Mwalongo F, Krone M, Karch G, Becher M, Reina G, Ertl T. Visualization of molecular structures using state-of-the-art techniques in webgl. Proceedings of the 19th international ACM conference on 3D web technologies, Web3D '14. New York, NY, USA: ACM; 2014. p. 133–41.

[20] Yuan S, et al. Implementing webgl and html5 in macromolecular visualization and modern computer-aided drug design. Trends Biotechnol 2017;35(6):559–71.

[21] Kokelj Ž, Bohak C, Marolt M. A web-based virtual reality environment for medical visualization. 2018 41st international convention on information and communication technology Electronics and Microelectronics (MIPRO); 2018. p. 0299–302. https://doi.org/10.23919/MIPRO.2018.8400057.

[22] Sherif T, Kassis N, Rousseau M-t, Adalat R, Evans AC. Brainbrowser: distributed, web-based neurological data visualization. Front Neuroinf 2015;8:89. https://doi.org/10.3389/fninf.2014.00089https://www.frontiersin.org/article/10.3389/fninf.2014.00089.

[23] R. Buels, E. Yao, C. M. Diesh, et al., Jbrowse: a dynamic web platform for genome visualization and analysis genome biology, Genome Biol 17 (1). doi:10.1186/s13059-016-0924-1.

[24] H. Jacinto, R. KÃ©chichian, M. Desvignes, R. Prost, S. Valette, A Web Interface for 3d Visualization and Interactive Segmentation of Medical Images.

[25] Birr S, Mönch J, Sommerfeld D, Preim U, Preim B. The liveranatomyexplorer: a webgl-based surgical teaching tool. IEEE Comput Graph Appl Mag 2013;33(5):48–58. https://doi.org/10.1109/MCG.2013.41.

[26] Zhao Y, Zhao X, Dong F, Clapworthy GJ, Ersotelos N, Liu E. Webgl-based interactive rendering of whole body anatomy for web-oriented visualisation of avatar-centered digital health data. 13th IEEE international conference on BioInformatics and BioEngineering 2013. p. 1–4. https://doi.org/10.1109/BIBE.2013.6701605.

[27] JimÃ©nez J, LÃ³pez A, Cruz J, Esteban F, Navas J, Villoslada P, de Miras JR. A web platform for the interactive visualization and analysis of the 3d fractal dimension of mri data. J Biomed Inf 2014;51:176–90https://doi.org/10.1016/j.jbi.2014.05.011https://doi.org/10.1016/j.jbi.2014.05.011.

[28] Mwalongo F, Krone M, Becher M, Reina G, Ertl T. Remote visualization of dynamic molecular data using webgl. Proceedings of the 20th international conference on 3D web technology, Web3D '15 New York, NY, USA: ACM; 2015. p. 115–22. https://doi.org/10.1145/2775292.2775307http://doi.acm.org.proxy1.lib.uwo.ca/10.1145/2775292.2775307.

[29] Jacinto H, Kéchichian R, Desvignes M, Prost R, Valette S. A web interface for 3d visualization and interactive segmentation of medical images. Proceedings of the 17th international conference on 3D web technology, Web3D '12 New York, NY, USA: ACM; 2012. p. 51–8. https://doi.org/10.1145/2338714.2338722http://doi.acm.org.proxy1.lib.uwo.ca/10.1145/2338714.2338722.

[30] Bernal-Rusiel JL, Rannou N, Gollub RL, Pieper S, Murphy S, Robertson R, Grant PE, Pienaar R. Reusable client-side javascript modules for immersive web-based real-time collaborative neuroimage visualization. Front Neuroinf 2017;11:32. https://doi.org/10.3389/fninf.2017.00032.

[31] Shi M, Gao J, Zhang MQ. Web3dmol: interactive protein structure visualization based on webgl. Nucleic Acids Res 2017;45:W523–7. web Server issue.

[32] Tiwari M, Kumar P, Agrawal A. Web-based volume visualization of 3d medical data using slice streaming method. Proceedings of the sixth international conference on computer and communication technology 2015, ICCCT '15. New York, NY, USA: ACM; 2015. p. 194–8.

[33] Campoalegre L, Brunet P, Navazo I. Interactive visualization of medical volume models in mobile devices. Personal Ubiquitous Comput 2013;17(7):1503–14. https://doi.org/10.1007/s00779-012-0596-0https://doi.org.proxy1.lib.uwo.ca/10.1007/s00779-012-0596-0.

[34] Halic T, Ahn W, De S. Optimization model for web based multimodal interactive simulations. Expert Syst Appl 2015;42(12):5245–55. https://doi.org/10.1016/j.eswa.2015.02.026https://doi.org.proxy1.lib.uwo.ca/10.1016/j.eswa.2015.02.026.

[35] Qiao L, et al. An html5-based pure website solution for rapidly viewing and processing large-scale 3d medical volume reconstruction on mobile internet. International journal of telemedicine and applications. 2017. p. 13https://doi.org/10.1155/2017/4074137.

[36] Pienaar R, Turk A, Bernal-Rusiel JL, Rannou N, Haehn D, Grant PE, Krieger O. CHIPS: a service for collecting, organizing, processing, and sharing medical image data in the cloud. 2017. CoRR abs/1710.00734 1710.00734http://arxiv.org/abs/1710.00734.

[37] Schroeder W, Martin K, Lorensen B. Visualization Toolkit: an object-oriented approach to 3D graphics. fourth ed. 2006. Kitware, Clifton Park, NY https://www.vtk.org/vtk-textbook/.

[38] Group K. WebGL 2.0 specification. Khronos Group; 2017https://www.khronos.org/registry/webgl/specs/latest/2.0/.

[39] A. S. Foundation. Apache http Server Documentation version 2.4. 2017http://httpd.apache.org/docs/2.4/http://httpd.apache.org/docs/2.4/.

[40] Murach J. Murach's MySQL. second ed. Mike Murach & Associates; 2015.

[41] Murach J, Harris R. Murach's PHP and MySQL. third ed. Mike Murach & Associates; 2017.

[42] Ferreira T, Rasband W. Imagej user guide. 2012. [cited IJ 1.46]. https://imagej.nih.gov/ij/docs/guide/.

[43] Kessenich J, Baldwin D, Rost R. The opengl shading language. https://www.khronos.org/registry/OpenGL/specs/gl; 2014.

[44] Zhang Q, Eagleson R, Peters TM. Rapid scalar value classification and volume clipping for interactive 3d medical image visualization. Vis Comput 2011;27(1):3–19.

[45] Bourdon R. Wampserver, a windows web development environment. 2018. [cited 5.04.2018]. http://www.wampserver.com/en/.

[46] Garrett JJ. Ajax: a new approach to web applications. June 2008http://AdaptivePath.com.

[47] Delisle M. Mastering phpMyAdmin 3.4 for effective MySQL management. first ed. Chichester: Packt Publishing; 2012.

[48] Bartholomew D. Getting started with MariaDB. second ed. Packt Publishing - ebooks Account; 2015.