Received 15 November 2022, accepted 12 December 2022, date of publication 26 December 2022, date of current version 3 January 2023.

Digital Object Identifier 10.1109/ACCESS.2022.3232275

RESEARCH ARTICLE

A Web-Based Synchronized Architecture for Collaborative Dynamic Diagnosis and Therapy Planning

QI ZHANG
School of Information Technology, Illinois State University, Normal, IL 61761, USA
e-mail: qzhan10@ilstu.edu

This work was supported by the Illinois State University, including the Research Grant, the Seed Grant, the Travel Funding of the School of Information Technology, the University Research Grant, the Publication Incentive Program of the College of Arts and Sciences, and the University Publication Support Program.

ABSTRACT Clinical treatment delivery often involves multiple medical professionals, where collaborative teamwork is crucial to ensure the quality of diagnosis and therapeutic decision making. With the development of Internet of Medical Things (IoMT) and its applications in mobile health (mHealth), healthcare services can be delivered to remote users. However, a challenging situation arises when the data are volumetric and dynamic, it will be difficult to achieve real-time performance in information streaming and data dynamic rendering and synchronization over Internet, as is the case with cardiac procedures, where both the anatomy and dynamic function characteristics of the organ must be considered. To address this issue, we have developed new algorithms and a web-based collaborative software architecture to display dynamic volumetric data in a web browser to enable dynamic data sharing with remote healthcare professionals. We illustrate our system using the diagnosis and treatment planning of an atrial septal defect (ASD) repair as an example to evaluate the capabilities of our platform. In this example, a series of three-dimensional (3D) ultrasound images are registered and fused with the high-quality magnetic resonance (MR) cardiac data to provide complementary information; virtual septal occluder patches are modeled and integrated into the beating heart views to facilitate the planning of the procedure to occlude the ASD, and feedback is streamed amongst all participating professionals. Our algorithm runs on a Node.js server with WebSocket protocol to synchronize dynamic heart rendering, so all the connected users can observe the same four-dimensional (4D) display of the dual-modality heart data during the process of examining cardiac anatomy, performing functional analysis, and sharing treatment strategies across distributed geographic locations. The presented computational models and software architecture create a new vehicle for collaborative exploration and manipulation of dynamic volumetric medical datasets.

INDEX TERMS Cardiac image display, web-based architecture, beating heart, raymarching, bidirectional visual synchronization, information streaming.

I. INTRODUCTION

Complex medical procedure planning often requires input from multiple clinicians [1], who may not be physically co-located – a situation that has recently been exacerbated during COVID-19. Under these conditions, to optimize medical resources while making full use of the clinical experience of multiple medical practitioners and clinical doctors, a web-based collaborative teleradiology system is critically important for data exploration and feedback streaming among team members [2], [3].

Internet of Medical Things (IoMT) connects various physical spatial-restricted medical equipments and personals through Internet and can access to distributed medical

The associate editor coordinating the review of this manuscript and approving it for publication was Lei Wei.

services and resources. The rapid progress of IoMT makes personalized mobile health (mHealth) and distributed medical treatment available to remote patients [4]. Moreover, the widely accessible medical devices and senors can collect patients' data to generate insights about their health information through IoMT. The ubiquitous data collecting and computing environment constructs a smart space for medical diagnosis and information sharing [5]. These Internet-based technologies create a cost-effective way to connect the remote patients with a personalized continuous monitoring system for semantic analysis of the acquired dynamic dataset and delivery of smart space based approach for emergency situations in personalized mobile healthcare [6], [7]. However, when the medical data are volumetric and dynamic, such as the case of cardiac procedures, it is still a challenging task to provide physicians who are in multiple geographical locations with real-time synchronized visual feedback and information streaming using the algorithms currently available in the traditional IoMT environment and smart space.

Furthermore, due to the complex anatomical structure of the human heart and the limitations of network and visualization hardware and software, rendering large and dynamically varying datasets (such as multi-modal cardiac data) and streaming diagnostic information via the Internet remains a topic of on-going research [8]. Current web-based visualization algorithms are mainly restricted to single modality data rendering on an HTML5 canvas using WebGL [9], [10], [11], [12], where the dynamic cardiac functions and detailed structures of the atrium, ventricle, aorta, and myocardium cannot be clearly revealed [13], [14], [15]. No research has been reported on integrating virtual tools into complimentary images in conventional cardiac image rendering methods, making it difficult to deliver satisfactory visual feedback related to the exploration of internal cardiac structures [16], [17], [18], [19]. To address this need, we developed new algorithms and a software platform to synchronously display fused multi-modality dynamic medical images over the Internet [20], [21]. A particularly challenging situation arises when rendering dual-modality four-dimensional (4D) heart data with virtual tool integration, where both the anatomy and dynamic function characteristics of the organ must be considered. To the best of our knowledge, this problem has not so far been addressed in the literature.

This paper is an extension of our research to address the above described limitations and challenges, which focuses on developing new medical data visualization algorithms with feature of interest enhancement in web browsers, information sharing and streaming strategies over Internet, graphics processing unit (GPU)-accelerated computational methods for parallel computation in data processing and volume rendering, mathematical models, and a unique web-based dual-modality medical data visualization and synchronization architecture for collaborative atrial septal defect diagnosis and therapeutic planning. Our platform can display dynamic cardiac images which have been pre-processed to identify their various anatomical sub-structures, along with registered 3D ultrasound images and a modeled virtual septal occlusion patch tool. All such data can be synchronized across all connected visualization workstations and mobile devices. The virtual septal occlusion tool is modeled using a ray-marching algorithm and merged into the volumetric data rendering pipeline. The presented software platform employs a Node.js framework and a WebSocket protocol to create bidirectional connections and visual streaming of cardiac data display among the server and all connected client computers. Moreover, a data management algorithm runs on the Node.js server to provide users with real-time data access and information sharing without geographic restriction. The developed algorithms and models can be considered as an extension to the existing computational methods available in the Internet of Medical Things (IoMT).

The organization of this paper is as follows. Section I introduces the project background and its connections with the current medical applications as well as the novelty and importance of the presented research. Section II illustrates the related work and Section III introduces the approaches related to data acquisition and processing, as well as texture generation. In this section, the methodologies employed to generate implicit surfaces and model the virtual septal occlusion tool are described, dynamic rendering with region of interest enhancement and virtual patch integration are illustrated, and the techniques employed for visual synchronization and information sharing are introduced in detail. Section IV discusses experimental results using the presented system for Internet-based collaborative atrial septal defect diagnosis and therapy planning, and concluding remarks are outlined in Section V.

II. RELATED WORK

Traditional teleradiology systems have been developed to address the problem of transferring radiological images between medical image workstations to facilitate diagnosis, treatment and patient follow-up, training, and consultation [22]. Abrardo and Casini [23] developed a web-based solution for remote access to radiologic data and diagnostic services, while Gomez et al. [24] presented a telemedicine platform that was based on an asynchronous transfer mode (ATM) for cooperative video conferencing, image digitizing, as well as local and remote transparent database access. A similar web-based environment was presented by Lasierra et al. [25] for teledermatology support of a dynamic evaluation process with clinical personnel. Moreover, a web-based teleradiology management system for automatically transferring images and radiologists' reports was reported by Caffery and Manthey [26].

To facilitate rapid development of a non-proprietary, low-cost teleradiology solution, Puech [27] introduced a software framework for reading and working on DICOM (Digital Imaging and Communications in Medicine) images, while Shen et al. [28] developed a medical image access and presentation system (MIAPS) for remotely accessing and presenting DICOM images. In addition, a multi-functional telemedicine system that supported both telediagnosis and teleconsultation services was proposed by Lin et al. [29]. For the purpose of integrating DICOM and common Internet services into an operational domain that allowed access to all levels of the DICOM information hierarchy, Koutelakis et al. [30] implemented a modular multiprotocol teleradiology architecture, while Mitchell et al. [31] introduced a client-server teleradiology system for the diagnosis of acute stroke, which could interactively visualize 2D and 3D brain images on a smartphone device.

None these teleradiology systems however took advantage of modern Internet protocols and were limited to processing and displaying a single image modality, static volumetric data or dynamic 2D DICOM images. They all lacked the capacity to provide synchronous real-time 4D visual feedback, which restricted their application in dynamic 3D cardiac image-based applications such as the planning of mitral, aortic, and tricuspid valve repair, ablation procedures for the treatment of atrial fibrillation, and the repair of septal defects (ASD).

Using 4D cardiac data sets, medical image computing and visualization methods can be developed to display different aspects of cardiac motion and extract quantitative motion parameters [32]. For example, Kim et al. [33] implemented a software platform for visualizing the myocardium and quantitatively evaluating cardiac MR data for ASD therapy, and Linguraru et al. [34] used real-time 3D ultrasound to guide atrial septal defect closure. Meanwhile real-time 3D echocardiographic images were employed by Suematsu and his colleagues [35] to guide beating heart atrial septal defect closure. Zhang et al. [36] designed a stereoscopic cardiac image dynamic visualization platform to provide cardiologists with real-time feedback regarding heart anatomical information, Kidoh et al. [37] reported a rendering method for motion analysis of cardiac computed tomography, and the impact of visualizing 4D flow on diagnosis and surgical planning was demonstrated by Christopher et al. [38].

However, all these described software platforms were limited to local computers without Internet-based synchronization or medical information streaming, so were unsuitable for collaborative analysis and treatment planning [39]. Moreover, while the self-gated 4D whole heart imaging technique can potentially allow cardiac anatomy and function to be assessed from a single measurement, the published literature lacks the ability to enhance various relevant sections of the anatomy (cardiac chambers for example), making it difficult to strike a balance between rendering internal structures and displaying their dynamic behavior during a network-based collaborative assessment [14].

Internet technologies can create advanced and rich web-based applications that allow radiologists to readily access teleradiology systems and view medical images remotely [8]. Through high-speed Internet connections, medical data can be communicated between distinct clinical departments, making it is possible to connect remote health care professionals. The concept of Cyber-Medicine System (CMS) is introduced and applied to research and development of Internet-based medical information systems, which can connect medical devices and healthcare services with patients and medical professionals, and provide support for assisting patients in the case of emergency and first aid [7]. In particular, the visualization of multi-modality cardiac data over the Internet can convey a significant amount of image information from various sources into a single and meaningful display [16], [40], [41], and the complementary image can facilitate timely diagnosis and identification of patients in need of surgical management [17], [18]. For instance, Qiao et al. [10] proposed an HTML5-based solution to access large-scale 3D cardiac volume, and Arumugham's research group [42] built a user interface to Local Area Network for storing and accessing medical records, while Min et al. [12] reported several applications for viewing radiological images remotely. Korzun [5] presented a detailed overview of the applications of the Internet of Things (IoT) in mobile health (mHealth) system for providing patients with customized smart services though Internet.

The concept model of a semantic layer and smart space based approach were introduced by Korzun and his colleagues for connecting patients with their surrounding equipments and for providing digital assistance for emergency situations in personalized mobile healthcare, and the introduced architectural model was used in developing the first aid assistance system for helping people with chronic cardiovascular deceases [6]. Moreover, Zavyalova and her colleagues [7] added the derived knowledge from analyzing the end-user activities to the CMS for supporting medical professionals and for delivering intelligent services to remote patients. Furthermore, an online visualization framework for cardiac data was implemented by Li et al. [15], and a web-based diagnosis system for improving the quality of cardiac surgery was introduced by Yano's team [11].

III. MATERIALS AND METHODS

In the description of our platform, we make extensive reference to the planning for an atrial septal defect (ASD) procedure as an example case of its application but emphasize that this platform is not specific to this procedure and can be readily adapted to any 3D dynamic image-based diagnostic/therapy planning procedure. ASD occurs when there is a failure to close the communication between the right and left atria, which is one of the most common type of heart defect around the world [43]. Surgical closure is a major treatment solution and can prevent functional cardiac deterioration, and is often accomplished using interventional cardiac techniques [44]. Due to the complexity and morphologic variations of the heart between patients, ASD is usually suspected clinically, so a web-based collaborative treatment platform is a valuable architecture in the confirmative diagnostic procedure for surgical repair.

A. DATA ACQUISITION

A set of T1-weighted 3D MR cardiac images of a healthy subject was acquired using a 1.5 T GE CVi scanner. The

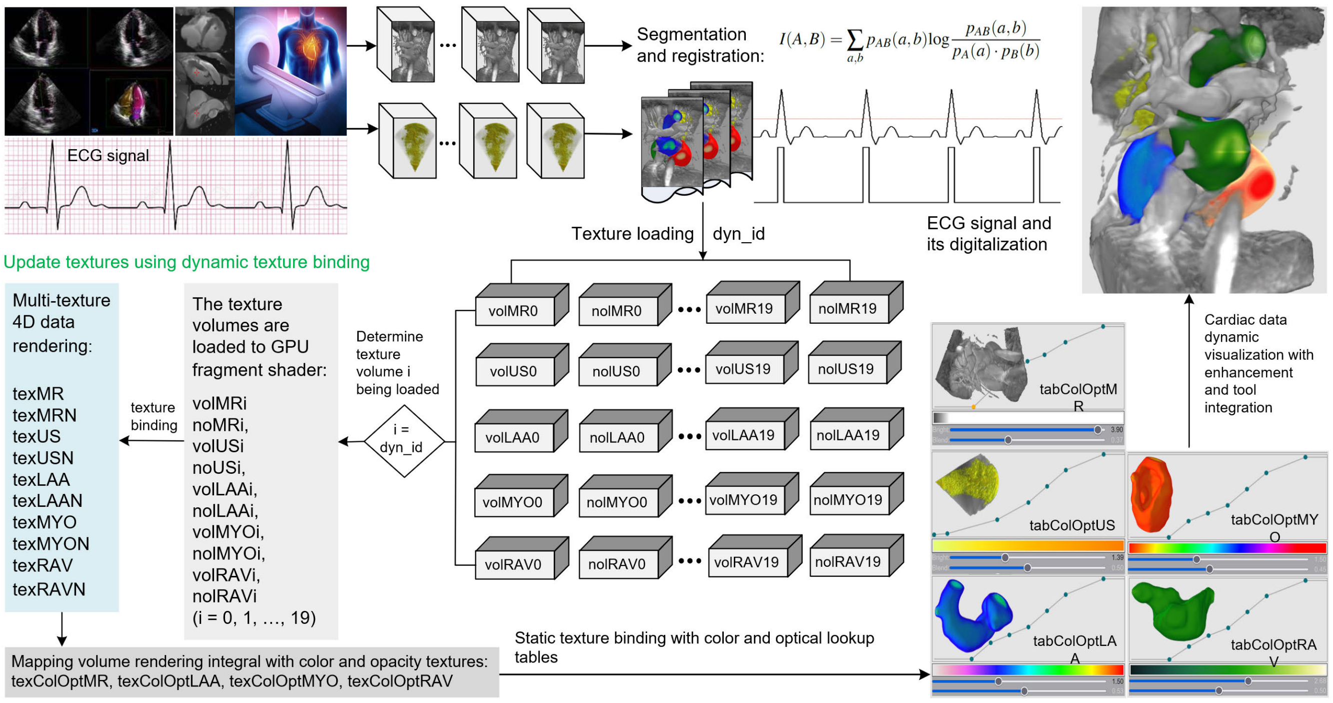

ECG signal
Segmentation and registration:
$I(A,B)=\sum_{a,b} p_{AB}(a,b)\log\frac{p_{AB}(a,b)}{p_A(a)\cdot p_B(b)}$
Texture loading
dyn_id
ECG signal and its digitalization
Update textures using dynamic texture binding
Multi-texture 4D data rendering:
texMR
texMRN
texUS
texUSN
texLAA
texLAAN
texMYO
texMYON
texRAV
texRAVN
texture binding
The texture volumes are loaded to GPU fragment shader:
volMRi
noMRi,
volUSi
noUSi,
volLAAi,
nolLAAi,
volMYOi,
nolMYOi,
volRAVi,
nolRAVi
(i = 0, 1, …, 19)
Determine texture volume i being loaded
i = dyn_id
volMR0
nolMR0
volMR19
nolMR19
volUS0
nolUS0
volUS19
nolUS19
volLAA0
nolLAA0
volLAA19
nolLAA19
volMYO0
nolMYO0
volMYO19
nolMYO19
volRAV0
nolRAV0
volRAV19
nolRAV19
Cardiac data dynamic visualization with enhancement and tool integration
tabColOptMR
tabColOptUS
tabColOptMYO
tabColOptLAA
tabColOptRAV
Mapping volume rendering integral with color and opacity textures: texColOptMR, texColOptLAA, texColOptMYO, texColOptRAV
Static texture binding with color and optical lookup tables

FIGURE 1. Flowchart describing the scanning of the patient and reconstructing 4D MR and ultrasound cardiac images gated with ECG signals, data storage scheme, dual-modality image registration and processing, cardiac chamber enhancement, texture construction, and dynamic texture binding.

imaging protocol employed an ECG-gated gradient echo pulse sequence and a 115 × 170 image matrix. As shown in Fig. 1, the dataset consists of twenty 3D images equally spaced throughout a cardiac cycle, each identified numerically from 0 to 19. For each image, the in-plane resolution was 1.48×1.48 mm and the slice thickness 1.5 mm.

The end-diastolic (ED) image from the 4D dataset was chosen as a reference for segmentation. This image was manually segmented to outline the morphological structures of interest: left atrium and aorta (LAA), myocardium (myo) of left ventricle, as well as right atrium and ventricle (RAV). The voxels of interest are labeled by setting the mask value to unity, while zero is used to mark the background voxels, ensuring an unambiguous differentiation during visualization. To reduce the intermixing artifacts along the edges of the segmented sub-volume, binary mask labels resulting from the explicit segmentation were filtered with the spherical Gaussian kernel to achieve a compromise between image quality and accuracy. Next, a nonlinear registration algorithm was used to register the ED image to each of the remaining images in the 4D dataset [45]. The acquired 19 transformation matrices were employed to assign the three segmented subvolumes of the ED image, corresponding to LAA, myo, and RAV, respectively, to the each of the remaining 19 heart volumes throughout a cardiac cycle.

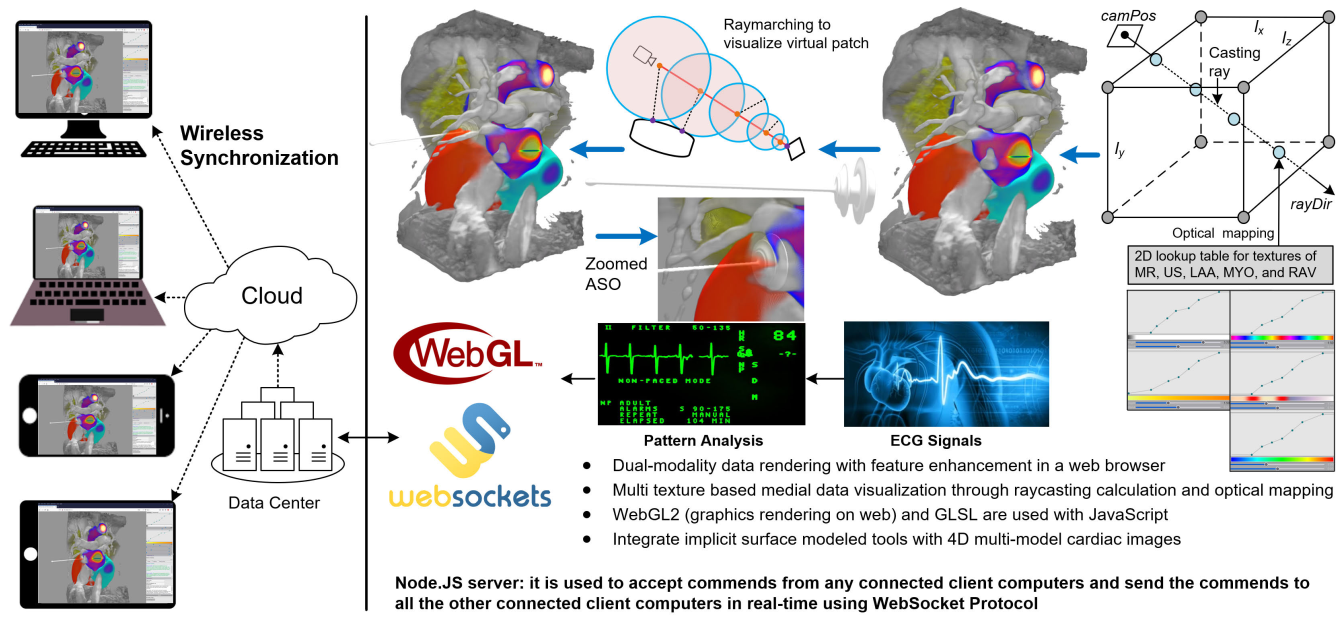
The next step was the acquisition of the ultrasound image, which is demonstrated in the top left of Fig. 1. Using a Polaris optical tracking system to continuously track an US probe while ECG signals are simultaneously acquired from the patient, fourteen 3D ultrasound images of the same subject were acquired on a Philips SONOS 7500 US machine. The pose of the ultrasound probe is fixed during acquisition of the US images, so we can assume that each tracking transformation is represented by the identity matrix. The dimension of each acquired ultrasound volume is $160\times208\times144$ with a voxel size $1.24\times1.26\times0.8$ mm^3.

Even though these data were acquired some time ago, using imaging equipment that is definitely not state of the art today, both systems nevertheless represent the kind of dynamic images that are acquired using current MR and ultrasound technology.

B. IMAGE REGISTRATION AND PROCESSING

As described in Fig. 1, for the 20 images in a cardiac cycle, the first one and its corresponding ultrasound image are used to initialize the registration, and the remaining images are employed for intra-procedural registration. A mutual information (MI) algorithm is applied to align the two images [46], which is illustrated by the formula in Fig. 1, where A and B are two discrete data volumes with individual probability density distributions $p_A(a)$ and $p_B(b)$.

Based on maximization of the mutual information metric, the MI algorithm iteratively transforms the floating template image until it is optimally aligned with the reference image. Due to artifacts, clutter and low signal-to-noise ratio (SNR) in the ultrasound images, when using the MI method to register 3D ultrasound to dynamic 3D MR images, the MI metric considers both the heart and background information in its



- Dual-modality data rendering with feature enhancement in a web browser
- Multi texture based medial data visualization through raycasting calculation and optical mapping
- WebGL2 (graphics rendering on web) and GLSL are used with JavaScript
- Integrate implicit surface modeled tools with 4D multi-model cardiac images

Node.JS server: it is used to accept commends from any connected client computers and send the commends to all the other connected client computers in real-time using WebSocket Protocol

FIGURE 2. Workflow and system architecture of modeling virtual septal occlusion tool using implicit surfaces and displaying them with 4D cardiac images in web browsers, during which graphics hardware accelerated algorithms including raymarching, raycasting, and optical mapping running on a Node.js server. The beating heart and fused virtual patch operations are streamed amongst all the connected computers using WebSocket protocol over ethernet or wireless network for maximum collaborative interaction.

calculation, which often results in a mismatch. Moreover, due to the large amount of information that must be processed, the convergence of MI algorithm is usually too slow for dynamic beating heart applications.

To address the issue of incorrect registration associated with the MI algorithm for registering ultrasound and MR images, and to accelerate the registration speed, a median filter is first applied to the ultrasound images to reduce the image speckles. In the ultrasound images used in this project, because most clutter and artifacts generally have a lower intensity compared with dominant features of interest, i.e., the boundaries between the heart muscle and chamber blood pools, a threshold is applied to the ultrasound images to obtain a binary mask image, which is employed to extract the filtered image, so only the most apparent anatomical features of interest are depicted and most artifacts are removed. The threshold value of 100 units within the 256 level scale is used in our system. Moreover, the field of view in the ultrasound images is restricted to highlight the cardiac structures to reduce the amount of information being processed during the registration procedure, therefore, the registration speed can be improved.

For the purpose of evaluating the registration accuracy, five anatomical landmarks were identified in both ultrasound and MR images, and their misalignment in the two registered images was calculated. The registration transformations were then applied to these landmarks to calculate the target registration error (TRE) of $\sim$1.8$\pm$0.9 mm. The intraprocedural transformation matrices were also applied to the segmented cardiac chambers of interest acquired in Subsection III-A, i.e., LAA, myo, and RAV for image alignment. The registered 3D MR images with segmented cardiac chambers and ultrasound images were processed by algorithms described in [21] to generate 2D montage images and then loaded to the vertex and fragment shaders in graphics memory, to render the registered multi-modality cardiac image volume in a web browser.

C. TEXTURE GENERATION

For segmented dataset volume rendering, the 5 groups of 20 source volumes $V[i, j]$ are stored as 3D arrays, where $i = 0, 1, \cdots, 4$, $j = 0, 1, \cdots, 19$, and $V[0, j]$, $V[1, j]$, $\cdots$, $V[4, j]$ holds volMR$_j$, volUS$_j$, volLAA$_j$, volMYO$_j$ and volRAV$_j$ respectively in Fig. 1.

For each voxel in the volume $V[i, j]$, we calculate its normal using trilinear interpolation, obtaining their corresponding normal volumes $N[i, j]$, which are also stored as 3D arrays with names nolMR$_j$, nolUS$_j$, nolLAA$_j$, nolMYO$_j$ and nolRAV$_j$, respectively. The intensity value of each voxel in the volumes $V[i, j]$ and $N[i, j]$ is normalized to a range [0, 255], and post-color attenuated classification algorithm [47] is employed to create a 2D lookup table, which maps each value between 0 and 255 to R, G, B color component and opacity α, where $\phi \in [0, 255]$ and $\alpha \in [0, 1]$, $\phi = R$, G, or B. These lookup tables are represented as 2D arrays $T[i]$ for optical mapping of MR$_j$, US$_j$ LAA$_j$, myo$_j$, and RAV$_j$ respectively with $i = 0, 1, \cdots, 4$.

In our JavaScript based software platform, WebGL functions *createTexture* and *bindTexture* are used to create and bind five 3D textures using *TEXTURE_3D* with initial data $V[i, 0]$, and create the corresponding five 2D textures using *TEXTURE_2D* with data $T[i]$ $(i = 0, 1, \cdots, 4)$. All these textures are loaded into the GPU fragment shader in a web browser using *gl.getUniformLocation* to upload data values to 3D textures vol_i and nol_i as well as 2D textures col_i respectively. Therefore, we have 5 texture sets Ω_i, where $i = 0, 1, 2, 3$, and 4 that are used to represent the 3D volume data textures and corresponding normal textures of MR, US, LAA, MYO and RAV respectively, as shown in Fig. 1.

D. IMPLICIT SURFACE GENERATION

For the purpose of modeling virtual septal occlusion tools used to repair the septal defect, a raymarching algorithm was designed to generate the implicit surface function $f(x)$ ($x \in \mathbb{R}^3$) in 3D space. To create an image on the virtual display window between the implicit surface and the camera, a ray $r(t) = p_c + \vec{r}_d \times t$ ($t \in \mathbb{R}^3$, $\vec{r}_d$ is the ray direction) is cast from the camera $p_c = (0, 0, c)$ through every pixel (*fragCoord.xy*) on the image plane into the scene to find the intersection point between $r(t)$ and the implicit surface $f(x)$, which is demonstrated in Fig. 2. If the search is successful, the color and shading of the corresponding pixel is computed, otherwise, the pixel is set to the background color.

To define an implicit surface in the scene, we assume the function $f(x)$ ($x \in \mathbb{R}^3$) is a continuous mapping from 3D to 1D space $f(x)$, i.e., $\mathbb{R}^3 \to \mathbb{R}$. The function $f(x)$ implicitly describes a point set $P \subset \mathbb{R}^3$ and is strictly negative over the interior points P, therefore, it returns zero on the boundary ∂P, which forms the implicit surface of $f(x)$ and can be represented as $f^{-1}(0)$. The entire 3D scene is defined as a signed distance function (SDF), which takes in a point $\tilde{p}_i$ ($i = 0, 1, \cdots, N-1$) as input and returns the shortest distance $\tilde{d}_i$ from that point $\tilde{p}_i$ to the surface $f^{-1}(0)$ of any object in the scene. For a point $\tilde{p}_i$ ($i = 0, 1, \cdots, N-1$) inside the rendering scene, the surface $f^{-1}(0)$ is given by a distance estimator, which returns the distance $\tilde{d}_i$ to it from the point $\tilde{p}_i$. By this means, for every point $\tilde{p}_i$, we can find the distance to the closest surface in the scene.

The SDF is employed to find the intersection point $\hat{p}$ of the casting ray $r(t)$ and implicit surface $f^{-1}(0)$. At each ray marching step k ($k \leq M-1$, where k starts from 0 and M is the preset maximum steps), if the return distance $\tilde{d}_k \leq \varepsilon$ ($k \leq M-1$), the casting ray $r(t)$ hits the implicit surface $f^{-1}(0)$, then the calculated color with shading information is added to the pixel on the image plane from which the ray is cast. This pixel is then used to generate the final pixel color of the rendered image on the display window. If $k > M-1$, (i.e. there are no implicit surfaces with the range along the marching ray), then the distance $\tilde{d}_k$ from camera to the interest point is set to -1, and the corresponding color of the pixel from which the ray is cast is set the background color. Otherwise, if $\tilde{d}_k > \varepsilon$ and $k < M-1$, march to the next step $k+1$ along the ray $r(t)$ until $\tilde{d}_k \leq \varepsilon$ and $k < M-1$ or $k \geq M-1$, and the corresponding calculated or background color is set to the pixel on the image plane from which the ray is cast.

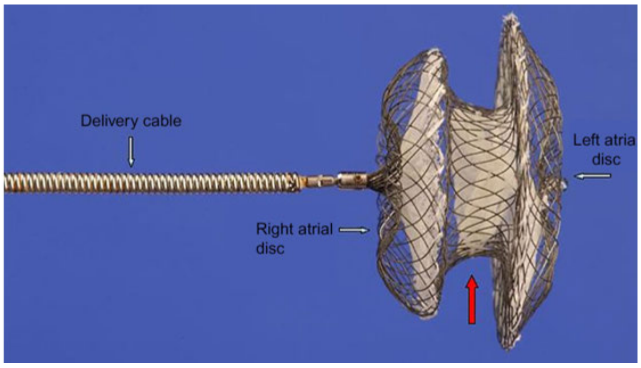


FIGURE 3. Amplatzer septal occluder (ASO): Lateral aspect. Left and right discs (blue arrows), central core (red arrow) [48].

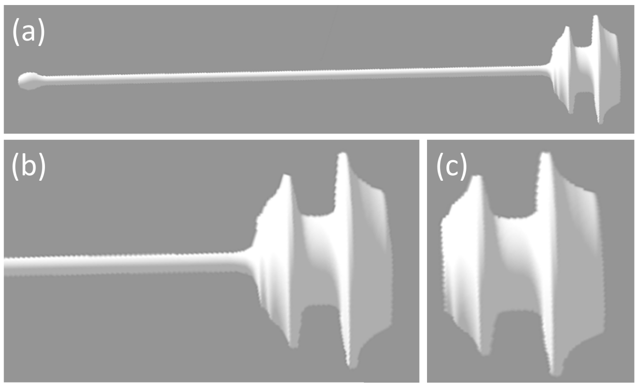


FIGURE 4. Modeled Amplatzer septal occluder (ASO) patch using raymarching implicit surface modeling algorithm. (a) ASO with delivery cable / bar; (b) enlarged ASO model; (c) ASO model without the delivery cable / bar.

E. SEPTAL OCCLUDER PATCH MODELING

The Amplatzer Septal Occluder (ASO) is the standard of care tool for minimally invasive atrial septal defect (ASD) closure. As illustrated in Fig. 3, the ASO is a double-disc occluder comprised of nitinol mesh and polyester material. To model a virtual ASO, the signed distance function (SDF) is employed to create implicit surfaces, which returns the minimum possible distance from point $\hat{p}$ to the surface it describes. When the sample point $\hat{p}$ is inside the implicit surface, a negative value is returned by SDF. The absolute distance from the sample point $\hat{p}$ to the surface $f^{-1}(0)$ is used in the process of implicit surface searching and calculation. Basic mathematical implicit surface functions are first defined, including torus, cone, round box, capped and rounded cylinders, and prism. Then, various implicit surface blending operations such as smooth union, subtraction and intersection are designed to merge multiple simple implicit surfaces to model comprehensive geometric surfaces such as the ASO tool.

Fig. 4 is the virtual model of the real ASO displayed in Fig. 3, which is created with a smooth union of seven capped cylinders and one round cone. The smooth factors are set

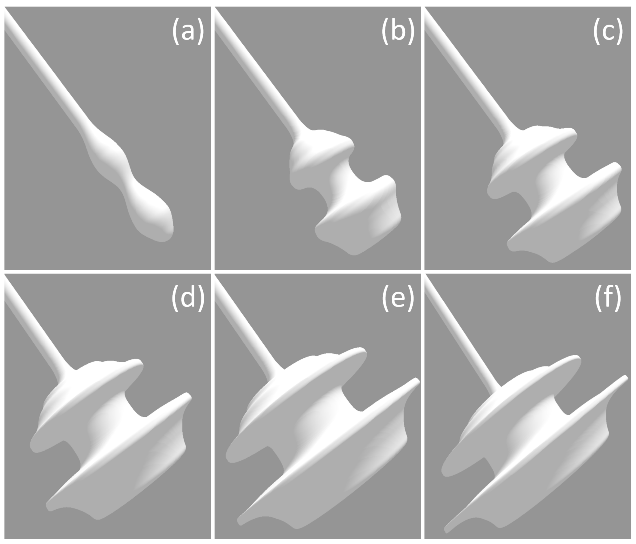


FIGURE 5. Modeled Amplatzer septal occluder (ASO) using a raymarching implicit surface modeling algorithm and different parameter ϕ value to control the patch size, where $\phi = 0.0$, 0.32, 0.5, 0.65, 0.86, and 1.08 corresponding to (a) - (f) respectively. The system patch size interactive adjustment can be used to fit various sizes and shapes of the defect in the intra-atrial septum.

between 0.01 and 0.04. The modeled ASO can detach the delivery cable under user control, which is demonstrated in Fig. 4 (c). The ASO is designed to securely appose the septal wall between left and right atria and create a platform for tissue in-growth after implantation. To accommodate different aperture sizes in the septum, our system permits the users to interactively control the patch size, as described in Fig. 5. The parameter $\phi \in [0.0, 1.5]$ is interactively controlled in the user interface, where $\phi = 0.0$, 0.32, 0.5, 0.65, 0.86, and 1.08 are used to create the corresponding virtual ASO patches in (a) - (f) respectively.

F. DYNAMIC CARDIAC DATA RENDERING AND FUSION

Inside a web browser, vertex and fragment shaders supported by WebGL2 are used for raycasting calculation and data rendering inside the graphics memory. The variables listed in Table 1 are used in the description of Algorithm 1. The texture set $tex\Pi_i$, $i = 0, 1, \cdots, 4$, represents 3D texture generated from dataset of magnetic resonance (MR), ultrasound (US), left atrium and aorta (LAA), myocardium of left ventricle (MYO), and right atrium and ventricle (RVA) respectively.

We assume $\vec{r} = \vec{r}(t) = ray(x(t), y(t), z(t))$ is a viewing ray cast from the screen into texture space, where $s(t_k) = s(\vec{r}(t_k))$ and $\vec{n}(t_k) = n(\vec{r}_n(t_k))$ are the intensity value and normal at the sampling point $\vec{r}(t_k)$ on the ray $\vec{r}(t)$, which is acquired by sampling texture vol_i and nol_i using texture coordinate at $\vec{r}(t_k)$. At each point $\vec{r}(t_k)$, the 2D texture generated from lookup table $colMap\Pi_i$ maps the intensity value $s(\vec{r}(t_k))$ to color $C_i(t_k)$ and opacity $\alpha_i(t_k)$, which is acquired from sampling texture col_i using the texture coordinate at $\vec{r}(t_k)$, where

TABLE 1. The meaning of variables used in the pseudo code of Algorithm 1.

Variable	Meaning explanation
Π_i	$i = 0, 1, \cdots, 4$, represents dataset MR, US, LAA, MYO, and RVA
$tex\Pi_i$	3D texture of dataset Π_i
$tex\Pi N_i$	3D normal texture of dataset Π_i
$colMap\Pi_i$	2D texture of color and opacity mapping for dataset Π
$bl\Pi_i$	Blending factor for rendered image of dataset Π_i
$br\Pi_i$	Brightness adjustment factor for rendered image of dataset Π_i
Num	The maximum sampling step in raycasting calculation
img	Data position after transformation in texture space
d_0, d_1	Two distances from the two intersection points of casting ray with texture cube to the camera
$\Omega[j]$	Array $\Omega = val, col, col, enhan$ for $j = 0, 1, \cdots 4$, corresponding to data set MR, US, LAA, MYO, and RVA
$\vec{r}_d$	Casting ray direction from camera to data texture
$\vec{r}$	Casting ray for volume rendering calculation
C_f, α_f	Calculated final pixel color and opacity on virtual screen
C_s, α_s	Calculated color and opacity from raymarching
θ	Preset opacity threshold used to stop raycasting calculation
μ	Adjustment factor for normal calcuation
p_c	Camera position
d_s	The calculated distance from camera to implicit surface
$ambLt$	Ambient light for the objects in the scene
$difLt$	Diffusion light for the objects in the scene
$ltDir$	Light direction
$ltVec$	Light reflection direction
p_i	Texture coordinate at sampling step i, $0 \leq i < Num - 1$
$val[i]$	Calculated color and opacity at current sampling step for i
$nol[i]$	Calculated normal at current sampling step for i
$col[i]$	Extracted color and opacity from transfer function texture at current sampling step for i
$enhan[i]$	Enhancement factor for diffuse color calculation of i
$brSur$	Implicit surface brightness adjustment factor
$brTran$	Volume data transparency adjustment factor
$volFlg$	Flag used to control volume rendering: true for rendering volume, false for not rendering volume.
$toolFlg$	Flag used to control tool rendering: true for rendering tools, false for not rendering tool.

$i = 1, 2, \cdots, 4$ represents 2D texture generated from dataset MR, US, LAA, MYO, and RVA respectively.

Eq. (1) represents the alpha blending used in computing the accumulated color and opacity, which is based on volume rendering integral.

$$I_i = \sum_{k=0}^{n-1} \alpha_i(t_k) C_i(t_k) \prod_{m=0}^{k-1} \left[T_i(t_m) \times bl\Pi_i \right] \tag{1}$$

The pre-multiplied $\alpha_i(t_k)C_i(t_k)$ represents the total light emitted from the point, and $\vec{r}(t_k)$, $T_i(t_k) = 1 - \alpha(t_k)$ describes the transparency of the point, where $\alpha_i(t_k) \in [0, 1]$ is its opacity, and $bl\psi_i \in [0, 1]$ is a blending factor, which can be interactively adjusted. Here $i = 0, 1, \cdots, 4$ represents the ray-casting calculation for MR, US, LAA, MYO and RAV respectively. During this process, for every ray $\vec{r}(t)$ cast, the computation based on Eq. (1) must be conducted at each sampling point $\vec{r}(t_k)$. To add sharing in our rendered cardiac volume, we set the light direction to $ltDir$, the ambient light color to $ambLt$, and the diffuse light color to $difLt$. Then, at each sampling step $\vec{r}(t_k)$, the improved Phong optical model is used to update the calculated color $C_i(t_k)$.

For each casting ray $\vec{r}(t)$ described by Eq. 1, the calculated value I_i includes both RGB color I_i^c and opacity I_i^α. We set a threshold $\mu = 0.95$, when $\Pi_{i=0}^{4} I_i^\alpha \geq \mu$, the calculation is stopped on the casting ray, and brightness adjustment factors $br\Pi_i$ are added to color, i.e., $I_i^c = I_i^c \times br\Pi_i$. The final color of the fused heart volume generated by the casting ray $\vec{r}(t)$ is $\Pi_{i=0}^{4} I_i$. The raycasting computations are executed in parallel on the GPU, so for each pixel in the display window, a ray is cast into texture space, and the raycasting calculation is conducted simultaneously on all the casting rays, enabling the final image to be generated in real time.

During the data visualization process, the patient's ECG signal is digitized and used to adjust heart rate through updating the 3D textures in WebGL function *texImage3D* from $V[i, 0]$ to $V[i, 19]$ in one cardiac cycle, and from $V[i, 19]$ to $V[i, 0]$ in the following cardiac cycle. This process is then repeated. Users can interactively manipulate the lookup table editor to adjust color and opacity mapping through updating $T[i]$ in WebGL's *texImage2D* function. Therefore, during the rendering process, the texture values can be dynamically updated in a GPU fragment shader and thus the rendered heart beats synchronously with the acquired ECG signals.

For each casting ray $\vec{r}(t)$ described by Eq. 1, the calculated value I_i includes both RGB color I_i^c and opacity I_i^α. We set a threshold $\mu = 0.95$, when $\Pi_{i=0}^{4} I_i^\alpha \geq \mu$, the calculation is stopped on the casting ray, and brightness adjustment factors $br\Pi_i$ are added to color, i.e., $I_i^c = I_i^c \times br\Pi_i$. The final color of the fused heart volume generated by the casting ray $\vec{r}(t)$ is $\Pi_{i=0}^{4} I_i$. The raycasting computations are executed in parallel on the GPU, so for each pixel in the display window, a ray is cast into texture space, and the raycasting calculation is conducted simultaneously on all the casting rays, enabling the final image to be generated in real time.

During the data visualization process, the patient's ECG signal is digitized and used to adjust heart rate through updating the 3D textures in WebGL function *texImage3D* from $V[i, 0]$ to $V[i, 19]$ in one cardiac cycle, and from $V[i, 19]$ to $V[i, 0]$ in the following cardiac cycle. This process is then repeated. Users can interactively manipulate the lookup table editor to adjust color and opacity mapping through updating $T[i]$ in WebGL's *texImage2D* function. Therefore, during the rendering process, the texture values can be dynamically updated in a GPU fragment shader and thus the rendered heart beats synchronously with the acquired ECG signals.

G. MULTI-MODALITY DATA VISUALIZATION WITH PATCH INTEGRATION

The pseudocode in Algorithm 1 (written in OpenGL Shading Language (GLSL) format) describes the procedure for rendering multi-modality data along with the enhanced volumes of interest and tools modeled with implicit surface models in the same web browser window. The variables employed are explained in Table 1.

Algorithm 1 Dual-modality data visualization and enhancement with integration of a virtual instrument

```
Require: Textures and transformation matrices are loaded into the shader
 1: 3D textures and voxel normals: $tex\Omega$, $tex\Omega N$, $\Omega$ = $MR$, $US$, $MYO$, $LAA$, $RAV$
 2: 2D textures of color and opacity mapping: $colMap\Omega$
 3: Transform matrix $trans$

Ensure: Pixel color and opacity $C_f, \alpha_f$
 4: Initialize $p_c, img, ambLt, difLt, ltVec, ltDir, Num, C_f, \alpha_f, N, M, \theta, \mu$
 5: Initialize $enhan[i], val[i], col[i], nol[i], i \in \{0, 1, \cdots, N-1\}$

 6: if $toolFlg$ then
 7:     $initRot()$
 8: end if
 9: $\vec{r}_d \leftarrow normalize(p_c - img)$, $\vec{r} \leftarrow makeRay(p_c, \vec{r}_d)$
10: $d_0, d_1 \leftarrow calIntersect(\vec{r}, \Delta)$

11: if $toolFlg$ then
12:     $C_s, \alpha_s \leftarrow rayMarch(p_c, \vec{r}_d)$, $d_s \leftarrow findDist(p_c, \vec{r}_d)$
13: end if

14: if $d_0 > d_1$ then
15:     if $!toolFlg$ then
16:         $C_f \leftarrow val[0].rgb$, $\alpha_f \leftarrow val[0].a$
17:     else
18:         $C_f \leftarrow C_s \times brSur$
19:     end if
20: else
21:     if $volFlg$ then
22:         $p_s \leftarrow p_c + d_0 \times r_d$, $p_e \leftarrow p_c + d_1 \times r_d$
23:         if $toolFlg$ && $d_s < (p_c, p_s)$ then
24:             $C_f \leftarrow C_f \times \alpha_f + C_s \times (1.0 - \alpha_f) \times brSur$
25:         end if
26:         for $i \in \{0, 1, \cdots, Num-1\}$ do
27:             $p_i \leftarrow mix(d_0, d_1, i/Num)$
28:             if $toolFlg$ && $d_s < distance(p_c, p_i)$ then
29:                 $C_f \leftarrow C_f \times \alpha_f + C_s \times (1.0 - \alpha_f) \times b_s$
30:                 break
31:             end if

32:             $k \in \{0, \cdots, 4\}$
33:             $col[k] \leftarrow texture(colMap\Omega, vec2(texture(tex\Omega, p_i).a, 0))$
34:             $nol[k] \leftarrow normalize(texture(tex\Omega N, p_i).xyz - \mu)$

35:             for $j \in \{0, 1, \cdots, N-1\}$ do
36:                 if $j < M$ then
37:                     $col[j].rgb \leftarrow ambLt \times col[j].rgb$
38:                 else
39:                     $enhan[j] \leftarrow clamp(dot(nol[j], ltDir), 0.0, 1.0)$
40:                     $col[j].rgb \leftarrow (ambLt + difLt \times enhan[j]) \times col[j].rgb$
41:                 end if
42:             end for

43:             $val[k] \leftarrow val[k] + col[k] \times (1.0 - val[k].a) \times bl\Omega$
44:             $C_f \leftarrow (\sum_{i=0}^{4} val[i]).rgb$, $\alpha_f \leftarrow (\sum_{i=0}^{4} val[i]).a$
45:             if $\alpha_f \geq \theta$ then
46:                 break
47:             end if
48:         end for
49:         if $toolFlg$ && $d_s \geq distance(p_c, p_e)$ then
```

```
50:             $C_f \leftarrow C_f \times \alpha_f + C_s \times (1.0 - \alpha_f) \times brSur$
51:          end if
52:       end if
53:       if $!volFlg$ && $toolFlg$ then
54:          $C_f \leftarrow C_s \times brSur$
55:       end if
56:    end if

57:    $C_f \leftarrow C_f + val[0].rgb \times (brMR - 1.0) + val[1].rgb \times (brUS - 1.0)$
58:    $C_f \leftarrow C_f + val[2].rgb \times (brLAA - 1.0) + val[3].rgb \times (brMYO - 1.0)$
59:    $C_f \leftarrow C_f + val[4].rgb \times (brRAV - 1.0)$

60:    if $brTran > 1.0$ then
61:       $C_f \leftarrow C_f + C_s \times (brTran - 1.0)$
62:    end if
```

The items (a) – (i) explain Algorithm 1, which uses variables *volFlg* and *toolFlg* to control the display of the image data, virtual tools, or both interactively.

(a) The algorithm inputs include 3D image textures and voxel normal created from MR, US, LAA, MYO, and RVA data sets, the corresponding 2D color mapping textures generated by individual transfer functions, and the matrix *trans* for data and implicit surface transforms. The outputs are the calculated color C_f and opacity α_f for each pixel in the GPU fragment buffer, which is displayed in the web browser directly.

(b) As shown in lines 4 – 5, first initialize camera and image position, lighting directions, ambient and diffuse colors, sampling step, threshold, as well as output color and opacity. Then, set the initial value for the color and opacity $val[i]$, voxel normal $nol[i]$ at the current sampling step, and the enhancement factor $enhan[i]$ for diffuse lighting calculation.

(c) When *toolFlg* is true, i.e., visualize the virtual patch, initialize the transformation matrices in function $initRot()$: lines 6 – 8. Transform and scale camera and 3D texture positions to get the casting ray direction $\vec{r}_d$, which is used to make the casting ray $\vec{r}$ and the function $callntersect(\vec{r}, \Delta)$ is used to find the intersection points d_0 and d_1 of the casting ray $\vec{r}$ with the texture unit cube Δ, which is illustrated in lines 9 – 10. If *toolFlg* is true, use camera position p_c and casting ray direction $\vec{r}_d$ to calculate the ray-marching color and opacity C_s and α_s, then find the distance d_s from the camera to the implicit surface in lines 11 – 13.

(d) As demonstrated by lines 14 – 19, when $d_0 > d1$, i.e., there is no intersection between casting ray and texture volume, if *toolFlg* is false, i.e., no implicit surface rendered, set C_f and α_f to the initial values and stop sampling the current ray. Otherwise, use the product of ray-marching color C_s and brightness adjustment factor *brSur* as the final output color C_f.

(e) When $d_0 \le d1$, there are two cases to consider for data and implicit surface rendering.

– The first case (lines 20 – 52) is when the rendering volume, i.e., *volFlg* is true. Calculate the start and end points p_s and p_e of the casting ray. If *toolFlg* is true and the distance from camera p_c to the implicit surface is less than the distance from camera to the casting ray start point p_s, calculate the final color C_f: lines 23 – 25. As shown by lines 26 – 48, at each sampling point p_i (i = 0, 1, $\cdots$, *Num* - 1), if *toolFlg* is true and the distance from camera p_c to the implicit surface is less than the distance from camera to point p_i, calculate the final color C_f through blending C_f and C_s using α_f and stop the current casting ray. Next, calculate the mapped color $col[j]$ using texture sampling of the 2D color mapping texture $colMap\Pi_j$, where j = 0, 1, $\cdots$, 4 corresponding to MR, US, LAA, MYO, and RVA respectively, and then acquire the corresponding normals $nol[j]$ using 3D texture sampling. As explained by code in lines 49 – 51, calculate the lighting value at point p_i for each of the five data modalities, and compute the accumulated color and opacity of each data modality and final color and opacity. If α_f is greater than the preset threshold θ, stop the current casting ray, otherwise, continue to loop to the next sampling point. When the casting ray is stopped, check whether *toolFlg* is true and d_s is greater than the distance from camera to the end of the casting point p_e. If it is true, blend the implicit color C_s with the final color C_f on the current casting ray.

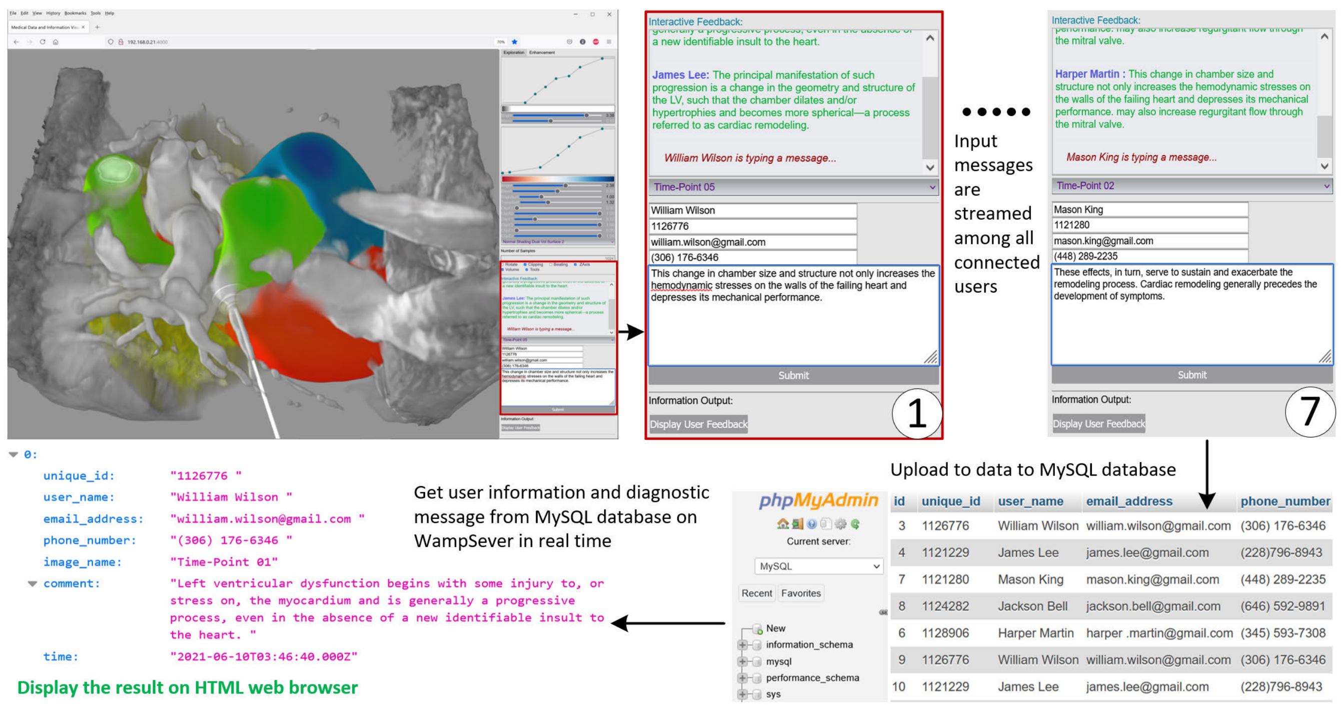
– The second case (lines 53 – 55) only renders the implicit surface, i.e., *volFlg* is false and *toolFlg* is true, and sets the final color C_f with the product of the raymarching color C_s and brightness adjustment factor *buSur*.

(f) Blend the final color with the calculated final color of MR, US, LAA, MYO, and RVA and the corresponding brightness adjustment factors, which is demonstrated by code in lines 57 – 59. As outlined in lines 60 – 62, if the data transparency factor *brTran* is greater than 1.0, update the final color C_f by adding it to the product of the implicit surface color C_s and *brTran* - 1.0.

H. VISUAL SYNCHRONIZATION AND INFORMATION SHARING

This software is developed on a web-based framework, which is based on a client/server architecture and bidirectional connections are established between server and all connected client computers. The Node.js server and WebSocket protocol ensures persistent connection and information sharing. Fig. 2 outlines the system architecture of the server side programming and the two-way interactive communications using WiFi gateway wireless access points or ethernet for wired networks. The server IP address and port 192.168.0.21:4000 is the IP and port of our server in the study used for client computer connections.

Using an event-driven scheme, our software platform can synchronize web-based data visualization, virtual instruments, and message transformation, allowing users to share the same dynamically updated streamed data. Algorithm 2 describes server side programming and operations, where



a new identifiable insult to the heart.

James Lee: The principal manifestation of such progression is a change in the geometry and structure of the LV, such that the chamber dilates and/or hypertrophies and becomes more spherical—a process referred to as cardiac remodeling.

William Wilson is typing a message...

William Wilson
1126776
william.wilson@gmail.com
(306) 176-6346
This change in chamber size and structure not only increases the hemodynamic stresses on the walls of the failing heart and depresses its mechanical performance.

the mitral valve.

Harper Martin : This change in chamber size and structure not only increases the hemodynamic stresses on the walls of the failing heart and depresses its mechanical performance. may also increase regurgitant flow through the mitral valve.

Mason King is typing a message...

Mason King
1121280
mason.king@gmail.com
(448) 289-2235
These effects, in turn, serve to sustain and exacerbate the remodeling process. Cardiac remodeling generally precedes the development of symptoms.

```
▼ 0:
    unique_id:        "1126776 "
    user_name:        "William Wilson "
    email_address:    "william.wilson@gmail.com "
    phone_number:     "(306) 176-6346 "
    image_name:       "Time-Point 01"
  ▼ comment:          "Left ventricular dysfunction begins with some injury to, or
                      stress on, the myocardium and is generally a progressive
                      process, even in the absence of a new identifiable insult to
                      the heart. "
    time:             "2021-06-10T03:46:40.000Z"
```

id	unique_id	user_name	email_address	phone_number
3	1126776	William Wilson	william.wilson@gmail.com	(306) 176-6346
4	1121229	James Lee	james.lee@gmail.com	(228)796-8943
7	1121280	Mason King	mason.king@gmail.com	(448) 289-2235
8	1124282	Jackson Bell	jackson.bell@gmail.com	(646) 592-9891
6	1128906	Harper Martin	harper .martin@gmail.com	(345) 593-7308
9	1126776	William Wilson	william.wilson@gmail.com	(306) 176-6346
10	1121229	James Lee	james.lee@gmail.com	(228)796-8943

FIGURE 6. Demonstration of the software platform's information streaming and data storage, in which users can input their comments regarding atrial septal defect diagnosis and treatment planning that can be shared among all the connected users, i.e., seven users' platforms are displayed. All the messages are stored in a MySQL database and can be extracted and displayed in web browsers.

line 1 shows the Node.js web application framework *express*, used to provide a set of features for web and mobile applications to monitor all the events at the designated server IP and port address. The event listeners are added to all the interactive panels, buttons, menus, and data rendering windows of our software platform, and when any of the listeners receive events or messages from user input or from a connected server, the software updates the data rendering, patch manipulations, message display, and other operations according to the commends contained in the events or messages. As shown in lines 2 – 3, the Socket.IO library runs on a Node.js server to build real-time, bidirectional communications between the server and connected clients. The created server object *io* is employed to listen to all events and receive all messages. At the same time, the function *io.sockets.on* broadcasts the received events and messages to connected clients in real time using the function *socket.broadcast.emit*, which controls all the events and message data from 1 to N as demonstrated by in lines 5 – 11, where N is the total number of events and messages.

We have also developed a message streaming scheme for data processing and information sharing, which is outlined in Fig. 6. All the input information and messages from the graphics user interface are automatically streamed into a database and synchronized amongst all the connected clients. Seven user platforms are listed ①, ②, $\cdots$, ⑦, user input messages such as name, id, email address, phone number, and diagnostic comments are shared among all the connected client computers in real time, and stored in a MySQL database running on an Apache HTTP server, which shares the same hardware platform with the Node.js server. All the saved data can be extracted and displayed in a web browser when user clicks the "Display User Feedback" button from any connected computers.

The algorithm for data storage and streaming described in Fig. 6 is explained inthe Algorithm 2 panel above. As demonstrated by line 4, *mysql* library is employed to create an object, which is then used with *express* object *app* to conduct data post and get operations. As shown by lines 16 – 19, *app.post* is used to acquire all user input messages from the client computer's user interface and save them to a database table. The *mysql* library creates a connection between the Node.js server and all the connected clients based on host and user: lines 20 – 22. At the same time, as described in lines 23 – 26, all the saved data can be extracted through the *mysql.createConnection* function and displayed in a web browser using *jason* on client computers, which connect via the Apache web server using the WebSocket protocol. The Apache web server operates on a Node.js framework that is supported by the Socket.IO library.

As described in Algorithm 3, event listeners are added to all $widget_i$ ($i = 1, 2, \cdots, N$), which monitors all operation events from both client itself or the Node.js server synchronized by the WebSocket protocol. Here $widget_i$ represents the platform's documents, bars, menus, rendering screens, and message display panels. Then, the platform running on

Algorithm 2 Server side event and message processing and emission

```
1: $const\ server \leftarrow require(`express').listen(process.env.PORT \parallel 4000, listen)$
2: $const\ io \leftarrow require(`socket.io')(server)$
3: $io.sockets.on(`connection', newConnection)$
4: $const\ mysql \leftarrow require(`mysql')$

5: function $newConnection(socket)$
6:     $i \in \{1, 2, \cdots, N\}$
7:     $socket.on(`eventi\_name', eventi\_msg)$
8:     function $eventi\_msg(eventi\_msg\_data)$
9:         $socket.broadcast.emit(`eventi\_name', eventi\_msg\_data)$
10:    end function
11: end function

12: function $listen()$
13:     $var\ host \leftarrow server.address().address$
14:     $var\ port \leftarrow server.address().port$
15: end function

16: function $app.post(`/user\_create', (req, res))$
17:     $const\ queryString \leftarrow$ "$INSERTINTOuser(message\_items)$"
18:     $getConnection().query(queryString, [message\_items])$
19: end function

20: function $getConnection()$
21:     Return $mysql.createConnection(host, user, database)$
22: end function

23: function $app.get(`/users', (req, res))$
24:     $const\ connection \leftarrow mysql.createConnection(host, user, database)$
25:     $const\ queryString \leftarrow SELECTmessage\_itmes$
26: end function
```

Algorithm 3 Client side event and message processing and emission

```
1: procedure $widget_i.addEventListener(`opt\_name', function(e))$
2:     $sendOptEvent_i(empty/opt\_data)$
3: end procedure

4: function $sendOptEvent_i(empty/opt\_data)$
5:     $socket.emit(`event_i\_name', opt\_data)$
6: end function

7: function $socket.on(`event\_name', opt\_data)$
8:     $`event\_name'$ and $opt\_data$
9: end function
```

the client computers updates system operation, messages, data rendering, tool manipulation according to the received events, and sends the operation event with operation data to the event function *sendOptEvent*, which is described in lines 1 – 3. As outlined by lines 4 – 6, the event and operation data is collected and transmitted through the *socket.emit* function. Simultaneously, the Socket.IO library running on the client side receives all events and operation data from the Node.js server, and then the client system updates the platform and conducts operations based on the received events and operation data, as illustrated in lines 7 – 9.

The received events and operation are forwarded to the Node.js server through function *socket.emit* using bidirectional WebSocket protocol in real time, and the server broadcasts the received messages to all the connected client computers using the Socket.IO library running on the server side, as presented in lines 4 – 6. Therefore, all the system and data rendering operations are synchronized by the Socket.IO library running on the Node.js server as well as all the client computers connected with WebSocket protocol, in real time.

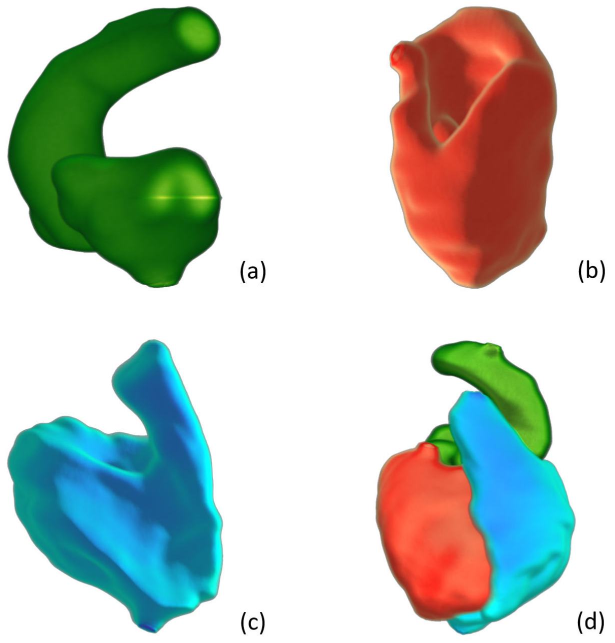


FIGURE 7. The end-diastolic (ED) image from 4D dataset is chosen as a reference for segmentation to outline the chambers of interest of (a) left atrium and aorta (LAA), (b) myocardium (myo) of left ventricle, (c) right atrium and ventricle (RAV), and (d) the combination of these three heart chambers. All these cardiac structures are rendered with of web-based raycasting algorithm.

IV. RESULTS AND DISCUSSION

An Intel 9900KF CPU, 64GB DDR4 memory, and equipped with four Nvidia graphics cards was used in the study to evaluate the system performance. As an illustrative example, we tested the dual-modality 3D rendered beating heart with chamber enhancement and virtual ASD repair instrument integration using the Chrome, Edge, and SeaMonkey in web browsers. The Node.js server was hosted on a HP Spectre x360 Laptop connected to the client computers using a wireless network that has ∼100 MB download and ∼20 MB upload speed. Using this platform, users can interactively manipulate the 4D image brightness, blending effect, virtual tool position and display transparency, and volume clipping.

A. DISPLAY RESULT WITH VIRTUAL OCCLUDER PATCH

As demonstrated in Fig. 7, to outline the structures of interest, the end-diastolic (ED) image from THE 4D dataset is chosen as a reference for segmentation. Three chambers (a) LAA, (b)myo, (c)RAV along with (d) the combination of all three, are chosen for segmentation and visualization.

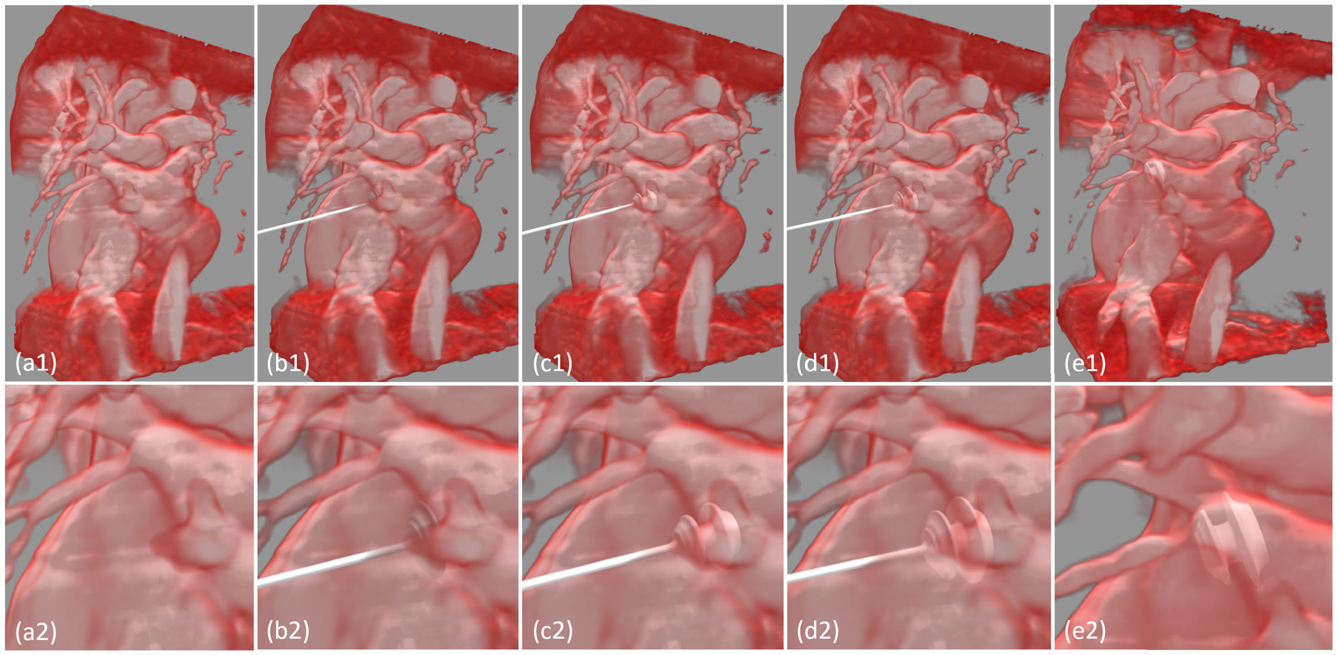


FIGURE 8. Visual results of MR cardiac data rendering with virtual septal occluder patch integration. The bottom row images (a2) - (e2) are enlarged part of the corresponding images of the top row in the same column (a1) - (e1). (ai) MR heart rendering; (bi) MR heart with virtual patch integration; (ci) enhanced transparency of (bi); (di) enlarged patch size of (ci); (ei) the delivery cable is detached, where i = 1, 2.

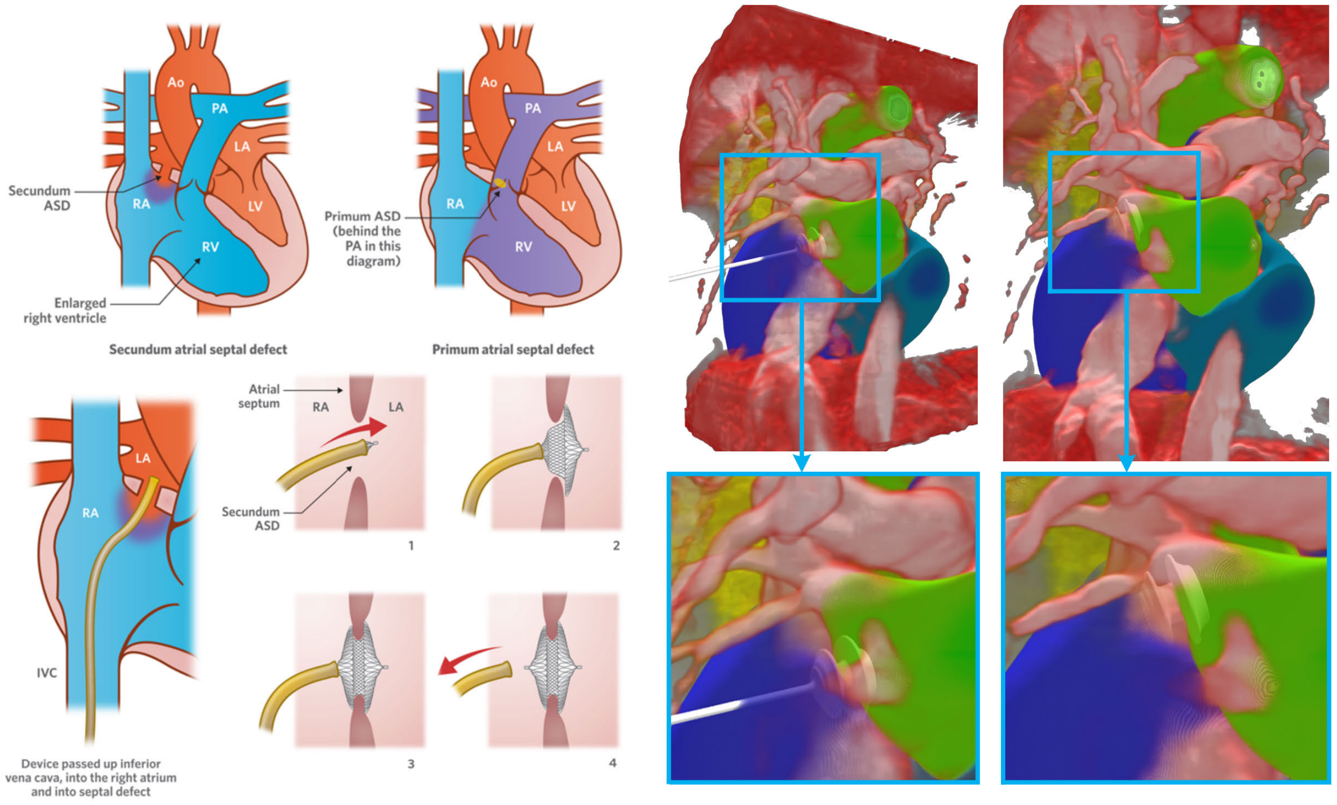

Ao
PA
LA
RA
LV
RV
Secundum ASD
Enlarged right ventricle
Secundum atrial septal defect
Primum ASD (behind the PA in this diagram)
Primum atrial septal defect
LA
RA
IVC
Device passed up inferior vena cava, into the right atrium and into septal defect
Atrial septum
Secundum ASD
1
2
3
4

FIGURE 9. Description of atrial septal defect (ASD) and treatment using septal occluder patch. Top left: Different types (primum and secundum) ASDs ASD [49]. Bottom left: 2D image of ASD treatment using the septal occluder patch to close the orifice between right and left atria [49]. Right: ASD treatment simulation and planning using dual-modality cardiac rendering with LAA, myo of left ventricle and RAV enhancement, as well as the operation of the virtual septal occluder patch.

Each is transformed into the 19 cardiac volumes using the matrices acquired from nonlinear registration as described in Subsection III-B. All these outlined chambers are merged with the entire cardiac data rendering pipeline.

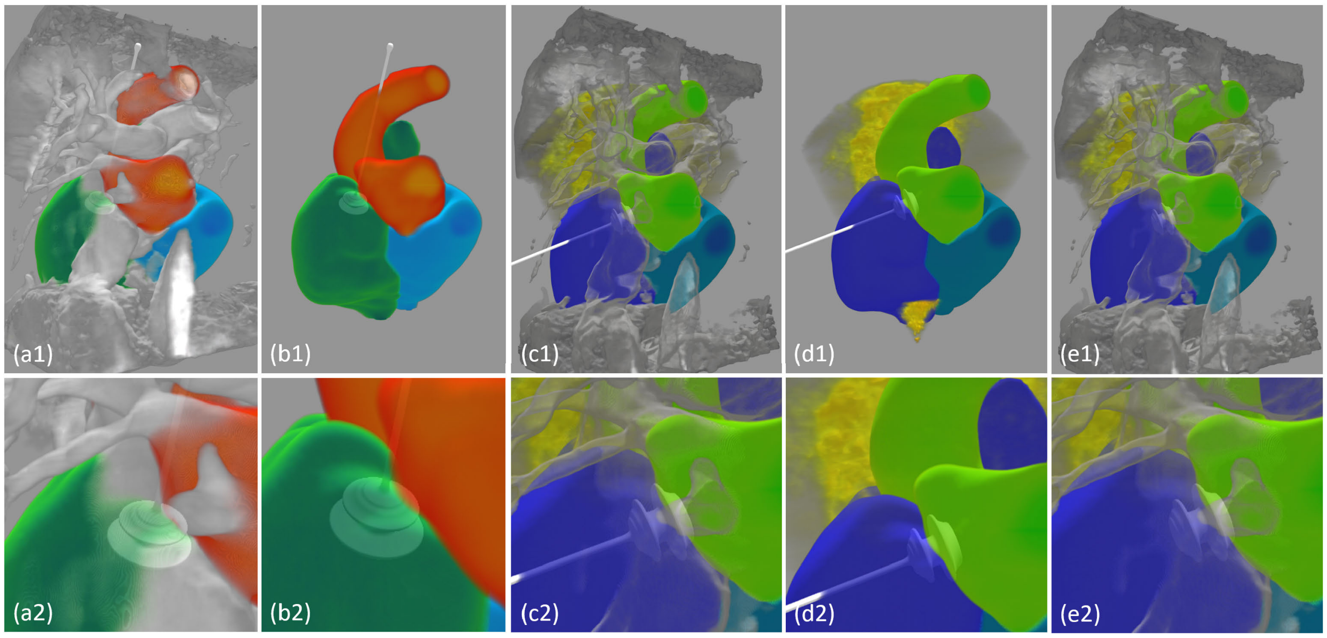


FIGURE 10. **Cardiac data display with virtual SOP integration. The bottom row images (a2) - (e2) are enlargements of the corresponding images of the top row in the same column (a1) - (e1). Image (ai) shows the cardiac MR frame with enhanced LAA, myo, and RAV enhancement, along with the virtual SOP placed to occlude the defect between the left and right atria. (bi) shows only the enhanced chambers LAA and myo integrated with the SOP. (ci)depicts the registered 3D ultrasound merged with the enhanced MR data and virtual SOP. In (di) only the MR chambers and ultrasound data are displayed with the SOP, while in (ei) the SOP delivery cable in (d1, d2) has been detached.**

Fig. 8 shows the result of rendering the cardiac MR image along with the integration the virtual SOP for ASD treatment planning and repair simulation. These images are derived from a single frame of the 20 cardiac volumes representing one cardiac cycle. The bottom row images (a2) - (e2) are enlargements of the corresponding images of the top row in the same column (a1) - (e1). (ai) shows the cardiac MR display, (bi) the virtual SOP merged with the MR image for planning the location of the patch to effect the repair, while in (ci), the transparency of the displayed heart is increased by adjusting the virtual tool factor *TransSurf* from 0.98 to 1.35, and keeping the factor *TransSurf* value constant in the following images, so the virtual SOP can be seen more clearly than (bi). In (di), the occluder patch size is enlarged to fit a larger defect, and the user can adjust the patch size interactively. The delivery cable can be detached to simulate the final repair result of closing the upper chamber septal defect, as shown in (ei).

Due to the single color used in Fig. 8, the chamber structures cannot be outlined clearly, so it is difficult to correctly place the virtual patch during planning the treatment. Therefore, as demonstrated in the right side of Fig. 9, the segmented chambers are registered and merged with the 4D rendered images, and the real-time 3D ultrasound is also included to provide additional imaging information. The left side of Fig. 9 describes the primum and secundum ASD configurations as well as the procedure using the occluder patch to effect the repair, along with the removal of the delivery cable to show the final treatment result. The right side this figure shows the result of using the virtual occluder patch to repair the defect, with the chambers of interest being enhanced with unique colors to show the boundaries, and 3D ultrasound is merged to provide real-time information. The SOP is placed between the left and right atria, and the cardiologists can interactively adjust virtual occluder location and the patch size to fit the heart wall hole. When the patch position inside the heart is satisfied, the delivery cable or bar can be removed to simulate the final surgical result, as show in the right image of (ei).

Various image exploration and septal patch operation results are illustrated in Fig. 10, in which the virtual patch tool factor *TransSurf* is interactively adjusted between 1.25 and 1.5 to enhance patch visibility. The bottom row images (a2) - (e2) are enlargements of the corresponding images of the top row in the same column (a1) - (e1). Fig. 10 (a) is a cardiac image rendered in grey with full opacity, and (b) shows only the chambers, both of which have a virtual SOP placed in an orientation that is different from those demonstrated in Fig. 8 and Fig. 9. Semi-transparency is used in the displayed MR heart of (ci) and (ei), allowing the user to visualize enhanced chambers,complementary ultrasound image data and the SOP clearly with the MR heart as a reference. Moreover, (ei) shows the final ASD treatment planning result by removing the delivery cable from the virtual patch, so the cardiologists can detect the placed virtual occluder patch as well as enhanced heart chambers in each of the 20 volumes of the cardiac data set.

The developed software platform can be used in various environment as long as suitable graphics hardware

TABLE 2. The System rendering environment comprises an Intel CPU 9900k and Nvidia graphics hardware: graphics processing unit (GPU), Firefox and Chrome web browsers; and software platform performance analysis. This latter analysis includes measuring web-based static and dynamic rendering speed in frames per second (FPS), time used for data loading in minutes from Node.js server to client computer, as well as time used for system synchronization between server and connected computers in milliseconds. We employ a wireless network that has ∼100 MB download and ∼20 MB upload speed.

System	System Environment		Performance Analysis					
Number	GPU	Web	Static Rendering ± STD (FPS)		Dynamic Rendering ± STD (FPS)		Loading Time	Synchronization
	Hardware	Browser	Volume	Volume + Tool	Volume	Volume + Tool	(Second)	(Millisecond)
1	GTX 980	Chrome	55.16 ± 3.84	14.09 ± 0.39	6.87 ± 0.21	6.84 ± 0.14	$42 \sim 110$	$0.33 \sim 3.76$
2	GTX 980	Edge	56.30 ± 5.57	19.72 ± 0.67	7.22 ± 0.18	7.28 ± 0.29	$98 \sim 113$	$0.50 \sim 4.13$
3	GTX 980	SeaMonkey	59.49 ± 5.87	14.60 ± 0.31	6.05 ± 0.06	4.60 ± 0.02	$217 \sim 217$	$0.35 \sim 3.69$
4	GTX 1070	Chrome	73.99 ± 5.89	22.73 ± 0.83	7.04 ± 0.29	7.38 ± 0.24	$69 \sim 110$	$0.41 \sim 3.81$
5	GTX 1070	Edge	75.42 ± 6.07	28.90 ± 1.29	7.20 ± 0.15	7.11 ± 0.16	$8 \sim 110$	$0.40 \sim 13.44$
6	RTX 1070	SeaMonkey	86.88 ± 8.80	24.00 ± 0.72	6.33 ± 0.10	5.41 ± 0.04	$220 \sim 220$	$0.42 \sim 3.56$
7	GTX 1080	Chrome	81.85 ± 5.84	27.20 ± 0.73	7.33 ± 0.24	6.88 ± 0.26	$64 \sim 111$	$0.57 \sim 3.38$
8	GTX 1080	Edge	79.77 ± 6.16	35.40 ± 0.75	7.12 ± 0.16	7.11 ± 0.21	$69 \sim 110$	$0.49 \sim 4.74$
9	RTX 1080	SeaMonkey	91.43 ± 11.01	27.61 ± 0.50	6.27 ± 0.08	5.47 ± 0.05	$48 \sim 152$	$0.48 \sim 3.79$
10	RTX 2080	Chrome	117.86 ± 7.48	39.94 ± 0.93	7.33 ± 0.24	7.06 ± 0.23	$4 \sim 110$	$0.32 \sim 4.04$
11	RTX 2080	Edge	117.40 ± 9.27	55.90 ± 1.04	6.98 ± 0.20	6.94 ± 0.23	$68 \sim 111$	$0.32 \sim 3.53$
12	RTX 2080	SeaMonkey	141.36 ± 7.70	42.01 ± 0.57	6.21 ± 0.06	5.88 ± 0.04	$52 \sim 194$	$0.36 \sim 3.44$

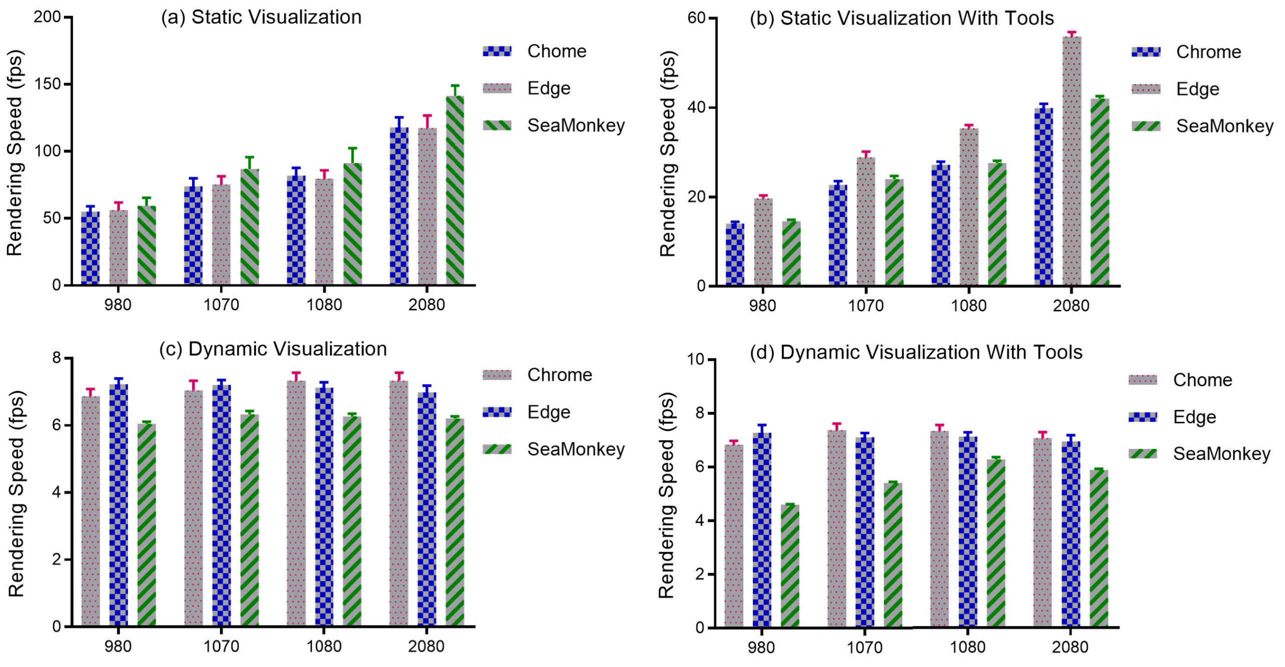


FIGURE 11. Bar chart of the dual-modality cardiac data rendering with chamber enhancement on four different Nvidia graphics cards and three web browsers using the evaluation results listed in Table 2. The visualization methods include cardiac static and dynamic rendering without and with virtual tool integration.

support, high-speed Internet, and web browser are available. When the multi-modality medical data have been loaded to the system, the system can deliver real-time synchronized dynamic data visualization with feature of interest enhancement and virtual instrument integration and medical information can be streamed among all the connected users. The introduced system can process various medical images, including computed tomography (CT), MR, and ultrasound. We plan to add functions for processing and analyzing CT scan heterogeneity, which can be quantified using texture analysis to extract spatial information from medical CT images.

B. PERFORMANCE EVALUATION

Table 2 shows the rendering performance of the system operating on the 4D cardiac MR images registered with complimentary 3D ultrasound images and with chambers of interest enhanced. Moreover, the virtual SOP is modeled and integrated into the image rendering pipeline. We have also tested image exploration synchronization and data the loading speed from a Node.js server to client computers connected with a wireless network and WebSocket protocol.

All the listed performance numbers in Table 2 are mean rendering speeds of 30 tests in frames per second (fps) and corresponding standard deviation (STD) for static and dynamic rendering with chambers of interest enhanced with and without virtual SOP integration. For static rendering, the graphics hardware is the major factor that determines the rendering speed. As shown in Fig. 11 (a), the system performance for static data rendering without virtual patch integration can achieve ∼141 fps when using an RTX 2080 card in the SeaMonkey web browser and ∼117 fps when using Edge and Chrome browsers with an average STD ∼8.2 fps. The lowest rendering speed of the system is greater than ∼55 fps when using the GTX 980 card on a Chrome browser, and all the STDs are less than 10 fps, which means that it can achieve real-time performance with relatively low speed variations without updating the hardware or web browser.

We also demonstrate that web browser can have an impact on the performance, which is demonstrated in Fig. 11 (b). For instance, when compared with Chrome or Edge browsers, SeaMonkey can increase rendering speed by up to ∼24 fps. When the virtual instrument is integrated into the data visualization pipeline, we note that the average performance increases from ∼50 fps for GTX 980 to ∼80 fps for RTX 2080, and the Edge browser gives the best performance when compared with Chrome or SeaMonkey. For virtual instrument integration, when using each of the four testing graphics hardware systems, we can obtain rendering performance that is faster than ∼57 fps on all the three web browsers.

When visualizing the entire 4D dual-modality dataset with the chambers of interest enhanced, we note a performance degradation, which can change from an average ∼125 fps to ∼7 fps. However, the graphics hardware employed has little impact. As illustrated in Fig. 11 (c), the average rendering speeds using the four GPUs are within the range from ∼6.7 fps to ∼6.9 fps with the average STD ∼0.16 fps, which means that the system can maintain a uniform rendering speed. When integrating the virtual instrument into the rendered data, we obtain an average speed ∼6.5 fps with STD ∼0.16 fps for the four graphics systems, a performance degradation of only ∼4.4%.

As demonstrated in Table 2, when using a wireless network connection with ∼100 MB download and ∼20 MB upload speed, for data loading from Node.js server into client computers, the average time needed to load all the 20 by 4 data sets is ∼110 seconds when using Edge or SeaMonkey, with a slower loading speed when using SeaMonkey with time variation less than 3%, which needs up to ∼200 seconds. In the developed system, the ∼2 minutes high loading time is only for the initial medical data loading, when the data have been loaded, they remain in the CPU main memory of the client computers and can be used for the following operations and rendering. The system only sends signals to synchronize the visualization of the heart beating. For the beating heart rendering synchronization between client computers connected with WebSocket protocol, the time delay is usually less than ∼5 milliseconds for all four systems and three web browsers used in the experiment, so users can get real-time visual feedback without perceptible delay. We believe the reason for the relatively high standard deviation of the loading times is the Internet speed fluctuation.

We conclude that the main performance restriction of our system for 4D data rendering, is texture loading from CPU main memory to GPU graphics memory and binding them with the 3D and 2D textures in GPU fragment shader, with the impact of the graphics card type being minimal. In Table 2, we compared the rendering speed using different generation GPUs, from which we can see that when using the newest generation GPU such as Nvidia 2080 in the experiment, the presented system can deliver a rendering speed that is faster than 110 fps, and the speed is ∼7 fps for dynamic rendering (tissue feature enhanced beating heart) with virtual tool integration. We believe the system rendering speed can be greatly improved when using the newest GPU such as Nvidia 4090. Moreover, the synchronization delay is less than ∼5 milliseconds, so all the connected users cannot detect any delay in beating heart synchronization and exploration over the Internet. Therefore, we can conclude that our system can provide sufficient performance to be employed in a collaborative multi-user environment for procedures such as cardiac intervention planning.

Fig. 6 illustrated that users can stream diagnostic messages over Internet, share the same dynamic cardiac images and interactively manipulate the virtual instrument using web browsers.

V. CONCLUSION AND FUTURE WORK

In this paper, we presented a web-based architectural model that enables interactive multi-user visualization and manipulation of dynamic medical image data in real time, and demonstrated its capabilities using the planning of an atrial septal defect repair procedure as an example. New algorithms, computational methods, implicit surface models, and a web-based collaborative software architecture were developed to display a series of MR and registered 3D ultrasound images of a beating heart, in which incorporated color-enhanced regions of interest and an interactive instrument - in this case a septal patch occlusion device. The software system runs on a Node.js framework and enables synchronous visualization across multiple Internet-connected observers without appreciable delay.

Taking advantage of mainstream graphics hardware and modern Internet protocols, our system can display static cardiac data in ∼140 frames per second (fps) with less than 8 fps standard deviation (STD), and render dynamic chamber enhanced cardiac images with virtual tools in a speed that is faster than the heart rate. During the planning procedure, cardiologists can stream diagnostic messages amongst all Internet connected collaborators, share the same high-quality cardiac views and instrument manipulations with less than ∼5 milliseconds delay.

This web-based architecture can stream synchronized images of the beating heart, which provides cardiologists and radiologists with 4D information of heart inner anatomies and its functions in real time, enabling them to collaboratively explore cardiac chamber structures and study its dynamic behavior without geographical limitations, and thus can be used as a valuable platform for collaboratively diagnosing atrial septal defect and planing an optimized treatment strategy over Internet.

The presented algorithms and software platform are for new technology development and medical verification, which have been not integrated with existing Electronic Health Record (EHR) systems. In the future work, when this research project is implemented in a real-world clinical scenario, we will consider the Protected Health Information (PHI) through setting advanced security measures, such as dynamic password and dual layer verification for protecting the access of the sensitive data and information regarding patients' identities, testing results, medication or prescription histories, phone records, and billing information. Role-based security levels will be used to ensure only those with clearance can see PHI. Furthermore, we will use encrypted methods when sending PHI electronically over Internet.

Deep learning based registration algorithms will be designed and used for MR and ultrasound image registration in our future work to improve the efficiency of processing a large amount of medical data. We also plan to intelligently integrate multi-source data generated from the Internet of Medical Things (IoMT) environment for first aid assistance to patients with chronic cardiovascular deceases.

ACKNOWLEDGMENT

The author would like to thank Dr. Terry M. Peters for his strategic feedback, article review, and editing. He acknowledges the medical data and support from the Dr. Peters' Laboratory, Robarts Research Institute, and the valuable technical assistance and collaborations of John Moore, Chris Wedlake, Dr. Gerard Guiraudon, Dr. Xishi Huang, and Dr. Marcin Wierzbicki.

REFERENCES

[1] J. Kiesewetter, F. Fischer, and M. R. Fischer, "Collaborative clinical reasoning—A systematic review of empirical studies," *J. Continuing Educ. Health Professions*, vol. 37, no. 2, pp. 123–128, 2017.

[2] E. Ntasis, T. A. Maniatis, and K. S. Nikita, "Real-time collaborative environment for radiation treatment planning virtual simulation," *IEEE Trans. Biomed. Eng.*, vol. 49, no. 12, pp. 1444–1451, Dec. 2002.

[3] L. Lunde, A. Moen, R. B. Jakobsen, E. O. Rosvold, and A. M. Braend, "Exploring healthcare students' interprofessional teamwork in primary care simulation scenarios: Collaboration to create a shared treatment plan," *BMC Med. Educ.*, vol. 21, no. 1, pp. 1–14, Dec. 2021.

[4] D. Korzun and A. Meigal, "Multi-source data sensing in mobile personalized healthcare systems: Semantic linking and data mining," in *Proc. 24th Conf. Open Innov. Assoc. (FRUCT)*, Apr. 2019, pp. 187–192.

[5] D. G. Korzun, *Internet of Things Meets Mobile Health Systems in Smart Spaces: An Overview*. Cham, Switzerland: Springer, 2017, pp. 111–129.

[6] D. G. Korzun, A. V. Borodin, I. V. Paramonov, A. M. Vasilyev, and S. I. Balandin, "Smart spaces enabled mobile healthcare services in Internet of Things environments," *Int. J. Embedded Real-Time Commun. Syst.*, vol. 6, no. 1, pp. 1–27, 2015.

[7] Y. V. Zavyalova, D. G. Korzun, A. Y. Meigal, and A. V. Borodin, "Towards the development of smart spaces-based socio-cyber-medicine systems," *Int. J. Embedded Real-Time Commun. Syst.*, vol. 8, no. 1, pp. 45–63, Jan. 2017.

[8] Z. N. Aghdam, A. M. Rahmani, and M. Hosseinzadeh, "The role of the Internet of Things in Healthcare: Future trends and challenges," *Comput. Methods Programs Biomed.*, vol. 199, Feb. 2021, Art. no. 105903.

[9] L. Liu, L. Wang, Q. Huang, L. Zhou, X. Fu, and L. Liu, "An efficient architecture for medical high-resolution images transmission in mobile telemedicine systems," *Comput. Methods Programs Biomed.*, vol. 187, Apr. 2020, Art. no. 105088.

[10] L. Qiao, X. Chen, Y. Zhang, J. Zhang, Y. Wu, Y. Li, X. Mo, W. Chen, B. Xie, and M. Qiu, "An HTML5-based pure website solution for rapidly viewing and processing large-scale 3D medical volume reconstruction on mobile internet," *Int. J. Telemedicine Appl.*, vol. 2017, pp. 1–13, 2017.

[11] K. Yano, H. Kanda, T. Iida, K. Hayashi, Y. Toyama, and T. Kunisawa, "Internet-based intraoperative real-time transesophageal echocardiography in cardiac surgery," *J. Cardiothoracic Vascular Anesthesia*, vol. 34, no. 4, pp. 1117–1120, Apr. 2020.

[12] Q. Min, X. Wang, B. Huang, and L. Xu, "Web-based technology for remote viewing of radiological images: App validation," *J. Med. Internet Res.*, vol. 22, no. 9, Sep. 2020, Art. no. e16224.

[13] J. Pang, B. Sharif, Z. Fan, X. Bi, R. Arsanjani, D. S. Berman, and D. Li, "ECG and navigator-free four-dimensional whole-heart coronary MRA for simultaneous visualization of cardiac anatomy and function," *Magn. Reson. Med.*, vol. 72, no. 5, pp. 1208–1217, Nov. 2014.

[14] S. P. Rowe, P. T. Johnson, and E. K. Fishman, "Cinematic rendering of cardiac CT volumetric data: Principles and initial observations," *J. Cardiovascular Comput. Tomogr.*, vol. 12, no. 1, pp. 56–59, Jan. 2018.

[15] W. Li, K. Yu, C. Feng, and D. Zhao, "SP-MIOV: A novel framework of shadow proxy based medical image online visualization in computing and storage resource restrained environments," *Future Gener. Comput. Syst.*, vol. 105, pp. 318–330, Apr. 2020.

[16] Q. Zhang, "Dual-modality cardiac data real-time rendering and synchronization in web browsers," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, Aug. 2020, pp. 1–5.

[17] B. Xu and R. Reyaldeen, "Ascending aortic graft infection—An expanding role for multi-modality cardiac imaging," *Int. J. Cardiol.*, vol. 333, pp. 246–248, Jun. 2021.

[18] Y. Zhang, W. Wu, Y. Li, and M. Xie, "Multi-modality echocardiographic imaging in cardiac amyloidosis," *Amer. J. Med. Sci.*, vol. 361, no. 2, pp. e19–e20, Feb. 2021.

[19] K. Lawonn, N. N. Smit, K. Bühler, and B. Preim, "A survey on multimodal medical data visualization," *Comput. Graph. Forum*, vol. 37, no. 1, pp. 413–438, 2017.

[20] Q. Zhang, "Medical data visual synchronization and information interaction using internet-based graphics rendering and message-oriented streaming," *Informat. Med. Unlocked*, vol. 17, Jan. 2019, Art. no. 100253.

[21] Q. Zhang, A. Samani, and T. M. Peters, "MR and ultrasound cardiac image dynamic visualization and synchronization over internet for distributed heart function diagnosis," *Computerized Med. Imag. Graph.*, vol. 88, Mar. 2021, Art. no. 101850.

[22] C. Ruggiero, "Teleradiology: A review," *J. Telemedicine Telecare*, vol. 4, no. 1, pp. 25–35, Mar. 1998.

[23] A. Abrardo and A. L. Casini, "Embedded Java in a web-based teleradiology system," *IEEE Internet Comput.*, vol. 2, no. 3, pp. 60–68, May 1998.

[24] E. J. Gomez, F. Del Pozo, E. J. Ortiz, N. Malpica, and H. Rahms, "A broadband multimedia collaborative system for advanced teleradiology and medical imaging diagnosis," *IEEE Trans. Inf. Technol. Biomed.*, vol. 2, no. 3, pp. 146–155, Sep. 1998.

[25] N. Lasierra, A. Alesanco, Y. Gilaberte, R. Magallón, and J. García, "Lessons learned after a three-year store and forward teledermatology experience using internet: Strengths and limitations," *Int. J. Med. Informat.*, vol. 81, no. 5, pp. 332–343, May 2012.

[26] L. Caffery and K. Manthey, "Implementation of a web-based teleradiology management system," *J. Telemedicine Telecare*, vol. 10, no. 1, pp. 22–25, Nov. 2004.

[27] P. A. Puech, L. Boussel, S. Belfkih, L. Lemaitre, P. Douek, and R. Beuscart, "DicomWorks: Software for reviewing DICOM studies and promoting low-cost teleradiology," *J. Digit. Imag.*, vol. 20, no. 2, pp. 122–130, Jun. 2007.

[28] H. Shen, D. Ma, Y. Zhao, H. Sun, S. Sun, R. Ye, L. Huang, B. Lang, and Y. Sun, ''MIAPS: A web-based system for remotely accessing and presenting medical images,'' *Comput. Methods Programs Biomed.*, vol. 113, no. 1, pp. 266–283, Jan. 2014.

[29] C.-C. Lin, H.-S. Chen, C.-Y. Chen, and S.-M. Hou, ''Implementation and evaluation of a multifunctional telemedicine system in NTUH,'' *Int. J. Med. Informat.*, vol. 61, nos. 2–3, pp. 175–187, May 2001.

[30] G. Koutelakis, G. Anastassopoulos, and D. Lymberopoulos, ''Application of multiprotocol medical imaging communications and an extended DICOM WADO service in a teleradiology architecture,'' *Int. J. Telemed Appl.*, vol. 2012, Jan. 2012, Art. no. 271758.

[31] J. R. Mitchell, P. Sharma, J. Modi, M. Simpson, M. Thomas, M. D. Hill, and M. Goyal, ''A smartphone client-server teleradiology system for primary diagnosis of acute stroke,'' *J. Med. Internet Res.*, vol. 13, no. 2, p. e31, May 2011.

[32] E. G. M. Kanaga, J. Anitha, and D. S. Juliet, ''4D medical image analysis: A systematic study on applications, challenges, and future research directions,'' in *Advanced Machine Vision Paradigms for Medical Image Analysis* (Hybrid Computational Intelligence for Pattern Analysis and Understanding), T. Gandhi, S. Bhattacharyya, S. De, D. Konar, and S. Dey, Eds. New York, NY, USA: Academic, 2021, ch. 4, pp. 97–130.

[33] Y.-C. Kim, K. R. Kim, K. Choi, M. Kim, Y. Chung, and Y. H. Choe, ''EVCMR: A tool for the quantitative evaluation and visualization of cardiac MRI data,'' *Comput. Biol. Med.*, vol. 111, Aug. 2019, Art. no. 103334.

[34] M. Linguraru, A. Kabla, N. Vasilyev, P. Del Nido, and R. Howe, ''Real-time block flow tracking of atrial septal defect motion in 4D cardiac ultrasound,'' in *Proc. 4th IEEE Int. Symp. Biomed. Imag., Nano Macro*, Apr. 2007, pp. 356–359.

[35] Y. Suematsu, J. F. Martinez, B. K. Wolf, G. R. Marx, J. A. Stoll, P. E. DuPont, R. D. Howe, J. K. Triedman, and P. J. Del Nido, ''Three-dimensional echo-guided beating heart surgery without cardiopulmonary bypass: Atrial septal defect closure in a swine model,'' *J. Thoracic Cardiovascular Surgery*, vol. 130, no. 5, pp. 1348–1357, Nov. 2005.

[36] Q. Zhang, R. Eagleson, and T. M. Peters, ''Dynamic real-time 4D cardiac MDCT image display using GPU-accelerated volume rendering,'' *Computerized Med. Imag. Graph.*, vol. 33, no. 6, pp. 461–476, Sep. 2009.

[37] M. Kidoh, D. Utsunomiya, Y. Funama, H. Ashikaga, T. Nakaura, S. Oda, H. Yuki, K. Hirata, Y. Iyama, Y. Nagayama, T. Fukui, Y. Yamashita, and K. Taguchi, ''Vectors through a cross-sectional image (VCI): A visualization method for four-dimensional motion analysis for cardiac computed tomography,'' *J. Cardiovascular Comput. Tomogr.*, vol. 11, no. 6, pp. 468–473, Nov. 2017.

[38] A. Christopher, L. Olivieri, R. Cross, K. Ramakrishnan, and Y.-H. Loke, ''4-dimensional flow by cardiac magnetic resonance informs surgical planning in partial anomalous pulmonary venous return,'' *JACC, Case Rep.*, vol. 2, no. 4, pp. 672–677, Apr. 2020.

[39] J. M. Noguera and J. Roberto Jiménez, ''Mobile volume rendering: Past, present and future,'' *IEEE Trans. Vis. Comput. Graphics*, vol. 22, no. 2, pp. 1164–1178, Feb. 2016.

[40] A. Gimelli, S. Ernst, and R. Liga, ''Multi-modality imaging for the identification of arrhythmogenic substrates prior to electrophysiology studies,'' *Frontiers Cardiovascular Med.*, vol. 8, Apr. 2021, Art. no. 640087.

[41] M. A. Azam, K. B. Khan, S. Salahuddin, E. Rehman, S. A. Khan, M. A. Khan, S. Kadry, and A. H. Gandomi, ''A review on multimodal medical image fusion: Compendious analysis of medical modalities, multimodal databases, fusion techniques and quality metrics,'' *Comput. Biol. Med.*, vol. 144, May 2022, Art. no. 105253.

[42] S. Arumugham, S. Rajagopalan, J. B. B. Rayappan, and R. Amirtharajan, ''Networked medical data sharing on secure medium—A web publishing mode for DICOM viewer with three layer authentication,'' *J. Biomed. Informat.*, vol. 86, pp. 90–105, Oct. 2018.

[43] A. M. Menillo, L. S. Lee, and A. L. Pearson-Shaver, *Atrial Septal Defect*. Treasure Island, FL, USA: StatPearls Publishing, 2021.

[44] F. Attie, M. Rosas, N. Granados, C. Zabal, A. Buendía, and J. Calderón, ''Surgical treatment for secundum atrial septal defects in patients >40 years old,'' *J. Amer. College Cardiol.*, vol. 38, no. 7, pp. 2035–2042, 2001.

[45] M. Wierzbicki, ''Validation of dynamic heart models obtained using non-linear registration for virtual reality training, planning, and guidance of minimally invasive cardiac surgeries,'' *Med. Image Anal.*, vol. 8, no. 3, pp. 387–401, Sep. 2004.

[46] X. Huang, J. Ren, G. Guiraudon, D. Boughner, and T. M. Peters, ''Rapid dynamic image registration of the beating heart for diagnosis and surgical navigation,'' *IEEE Trans. Med. Imag.*, vol. 28, no. 11, pp. 1802–1814, Nov. 2009.

[47] Q. Zhang, R. Eagleson, and T. M. Peters, ''Rapid scalar value classification and volume clipping for interactive 3D medical image visualization,'' *Vis. Comput.*, vol. 27, no. 1, pp. 3–19, Jan. 2011.

[48] J. Thomson and S. Qureshi, ''Device closure of secundum atrial septal defect's and the risk of cardiac erosion,'' *Echo Res. Pract.*, vol. 2, pp. R73–R78, Apr. 2015.

[49] (2022). *Atrial Septal Defect*. [Online]. Available: https://www.rch.org.au/cardiology/heart_defects/Atrial_Septal_Defect/

QI ZHANG received the B.S. degree in computational mathematics and applied software and the M.Sc. degree in computational mathematics from Jilin University, the M.Sc. degree in computer science from the University of Waterloo, and the Ph.D. degree in biomedical computer engineering science from Western University. He was an Assistant Professor at the State University of New York at Canton, a Researcher at the National Research Laboratory, Canada, and a software engineer at high-tech industry. He is an Associate Professor of computer science with the School of Information Technology, Illinois State University (ISU), and an Adjunct Faculty Member with Western University. He has published quality research papers in peer-reviewed leading journals and conferences and was awarded various research grants. His research interests include computer graphics, visualization, virtual reality, medical imaging, mobile computing, networking, and machine learning.