

A modular and scalable computational framework for interactive immersion into imaging data with a holographic augmented reality interface

Jose D. Velazco-Garcia^{a,*}, Dipan J. Shah^b, Ernst L. Leiss^a, Nikolaos V. Tsekos^a

^a MRI Lab, Dept. of CS, University of Houston, 4800 Calhoun Road PGH 501, Houston, TX, USA

^b Cardiovascular MRI Lab, Houston Methodist DeBakey Heart and Vascular Center, 6550 Fannin St., Smith Tower – Suite 1801, Houston, USA

ARTICLE INFO

Keywords:

Medical imaging
Rendering and visualization
Augmented reality
Computational framework
Human-cyber Interface
Interactive Immersion

ABSTRACT

Background and objective: Modern imaging scanners produce an ever-growing body of 3D/4D multimodal data requiring image analytics and visualization of fused images, segmentations, and information. For the latter, augmented reality (AR) with head-mounted displays (HMDs) has shown potential. This work describes a framework (FI3D) for interactive immersion with data, integration of image processing and analytics, and rendering and fusion with an AR interface.

Methods: The FI3D was designed and endowed with modules to communicate with peripherals, including imaging scanners and HMDs, and to provide computational power for data acquisition and processing. The core of FI3D is deployed to a dedicated computational unit that performs the computationally demanding processes in real-time, and the HMD is used as a display output peripheral and an input peripheral through gestures and voice commands. FI3D offers user-made processing and analysis dedicated modules. Users can customize and optimize these for a particular workflow while incorporating current or future libraries.

Results: The FI3D framework was used to develop a workflow for processing, rendering, and visualization of CINE MRI cardiac sets. In this version, the data were loaded from a remote database, and the endocardium and epicardium of the left ventricle (LV) were segmented using a machine learning model and transmitted to a HoloLens HMD to be visualized in 4D. Performance results show that the system is capable of maintaining an image stream of one image per second with a resolution of 512×512 . Also, it can modify visual properties of the holograms at 1 update per 16 milliseconds (62.5 Hz) while providing enough resources for the segmentation and surface reconstruction tasks without hindering the HMD.

Conclusions: We provide a system design and framework to be used as a foundation for medical applications that benefit from AR visualization, removing several technical challenges from the developmental pipeline.

1. Introduction

In diagnostic imaging, as well as in the ever-growing clinical adaptation of image-guided interventions, an important challenge is the effective visualization and use of three dimensional (3D) and/or multi-slice imaging information [1–7]. This challenge is particularly relevant to the practice of interventional medicine that re-

quires the accurate appreciation of spatial relationships of anatomical structures that are important for both appreciating the shape, position, and size of the pathologic loci, as well as for planning access paths. The latter is critically important to correctly reach the target(s) and to prevent unintended damage to healthy or vital tissue. Today, almost all imaging modalities can provide 3D data. Among them, magnetic resonance imaging (MRI) is one of the most valuable modalities [8]. It allows the collection of images with a wide range of contrast mechanisms to investigate different properties of the healthy or diseased tissue. In addition, MRI offers computer-controlled selection of the orientation and position (spatially oblique) of multi-slice or 3D scanning, as well as an inherent coordinate system generated by the magnetic field gradient coils of

* Corresponding author at: MRI Lab, Dept. of Computer Science, University of Houston, 4800 Calhoun Road PGH 501, Houston, TX, USA.

E-mail addresses: jdelazcogarcia@uh.edu (J.D. Velazco-Garcia), DJShah@houstonmethodist.org (D.J. Shah), coscel@handy.cs.uh.edu (E.L. Leiss), nvtsekos@central.uh.edu (N.V. Tsekos).

the scanner. Secondary to the state-of-the-art hardware and software of MRI scanners is the unparalleled ability to alter imaging acquisition protocols on-the-fly, i.e., while data is collected, and the patient resides in the MRI scanner [9,10]. This is a property that may enable paradigm changes toward interactive data collection and visualization with concomitant scanner control to improve informational content. As the potential of such human-to-data and human-to-sensor interactions in the clinical realm is emerging, computational frameworks are needed to seamlessly integrate the multitude of reconstruction, processing, analytics, and human-data interaction facilities. Such a framework is the Gadgetron [11], offering a pipeline and plethora of tools for image reconstruction from raw data and customization of data flow. Another platform with a similar objective integrates image acquisition, visualization, and robot control in MRI guided interventions [12]. Also, a recent work introduced an extensible software platform for deploying, investigating, and evaluating cardiovascular imaging research [13]. These applications provide significant tools for medical imaging research but do not offer visualization mechanisms that benefit from true 3D augmented reality (AR) visualization.

Despite this multimodal, multi-contrast 3D/4D information, the cornerstone of image-based diagnosis and planning is two-dimensional (2D) visualization. In practice, the vast majority of clinicians use conventional 2D displays to view 3D or even 4D information and reconstruct internal structures, performing mental extraction of features and their spatial relationships by viewing numerous 2D MRI slices from 3D or multi-slice sets [14]. In practice, understanding complex 3D tissue from multi-contrast or multimodal 3D imaging data sets is challenging and time-consuming. In response, many pioneering and groundbreaking works in rendering techniques have enabled 3D visualization, such as maximum intensity projections in angiography and virtual colonoscopy, e.g., [2,4,5]. However, 2D visualization remains the standard practice. AR visualization has been hailed as a potential solution to the above challenges, and numerous works have contributed in establishing AR visualization in different medical domains [15–17]. Indeed, by fusing and co-registering images, segmented structures, and patient models into an AR scene, information is contextualized. Such enhanced operator immersion into 3D information was recently demonstrated with AR capable head-mounted displays (HMDs) [2,4,5,18,19]. Furthermore, a growing number of studies demonstrate the potential of AR capable HMDs in different medical domains, including image guided interventions [1–7,20–24]. However, HMDs have limited processing capabilities (RAM, CPU, and storage) that preclude a single-platform implementation of advanced medical imaging data processing algorithms and control of imaging scanners. In addition, human-in-the-loop, i.e., direct involvement of the operator to the decision process or control loop, or on-the-loop, i.e., operator supervision of the process or control loop, interaction with data processing or analytics and visualization is rapidly expanding with established or new algorithms and libraries. As a result, to bring those tools into the clinical realm, seamless integration and customization are necessary, as discussed in [7,22].

A solution to this challenge is to implement at the least a two-processor approach: the main computer, running the computational core, and an HMD, acting as an I/O interfacing device. In this work, we describe a computational framework that runs on a dedicated CPU that (i) communicates with the HMDs and controls holographic scenes, (ii) enables operator-controlled data acquisition (i.e., control of the MRI scanner on-the-fly during data collection), (iii) provides computational power for image data processing and analytics, and (iv) enables the user into an interactive immersion with the data. Within this context, we have investigated integrating image analysis and visualization of medical data that included early methodologies for MRI-guided intracardiac

interventions [25,26] and MRI-guided robot interventions [12], as well as recent works that assessed modern immersive technologies with holograms using the Microsoft HoloLens Generation 1 HMD [27] for prostate cancer interventions [22,28,29], neurosurgery [21], and use of artificial intelligence for enabled automatic tissue segmentation [30].

From these studies, it became evident that to effectively integrate interactive control of imaging scanners, streamline in real-time image processing, and immerse the operator into the data a framework was needed that offered common functionalities as well as an easiness incorporating of custom-made software and interfacing. As a result, we embarked in the development of the Framework for Interactive Immersion into Imaging Data (FI3D) for medical data visualization through an interactive and immersive interface using AR capable HMDs. The objective of the presented framework is to offer a tool for processing and visualization of imaging data endowed with interactive AR interfaces to immerse the physician into multimodal and multidimensional data. The framework is the outcome of prior efforts and research experiments, including user input from collaborating clinical sites. An earlier system that demonstrated the potential of linking imaging data collection, the image segmentation, 3D rendering, spatiotemporal fusion of virtual entities, and visualization on a standard display was first presented in [12]. In parallel, we were investigating the use of emerging holographic technologies and performed preliminary developmental and assessment of AR visualization with HMDs [20–22]. Based on those works, we embarked on developing the FI3D framework, and specialized applications of it were used and tested in specific clinical paradigms. These works include simulation of robot-assisted transrectal MR-guided prostate biopsies [28,29]. In addition, the FI3D framework was used for immersive visualization of 4D cardiac renderings generated on-line using a machine learning model for segmenting CINE MRI [30]. These works used a developing version of the framework to describe the use of AR visualization but not the computational framework. Having a completed and stable version, in this paper, we introduce the FI3D framework upon which these prior works were based on and the steppingstone for future integration of imaging scanners and the end-user.

2. System design

The design objectives of the platform, performed with collaborating clinicians, was to: (1) Increase the computational power beyond what is available with commercial HMDs; (2) Be interfaceable with devices that generate data, such as MRI scanners, as well as with applicators, such as robotic surgical manipulators and human-to-information haptic devices; (3) Be modular and adaptable to specific clinical paradigms or needs in order to incorporate medical image and data processing to existing or custom made libraries implemented using C/C++. The solution of running the code on a processor, separate from the HMD, was pursued to achieve all three objectives. The system is deployed on and executed in at least two distinct but interconnected physical entities: the core computational unit, the FI3D, and at least one AR capable HMD, labeled as Framework Interface (FI). Each entity is programmed independently of the other and communicates through a communication protocol using TCP. FI3D executes all data related tasks and keeps all connected FIs up to date and synchronized with any pertinent information. FIs serve two main purposes: (i) with the data received from FI3D, generate the holographic scene and present it via its holographic projector, and (ii) control FI3D through the device's input mechanism, such as hand gestures and voice commands for the Microsoft's HoloLens. Fig. 1 shows an overview of the system's design, illustrating the interface of the operator to the different entities in it.

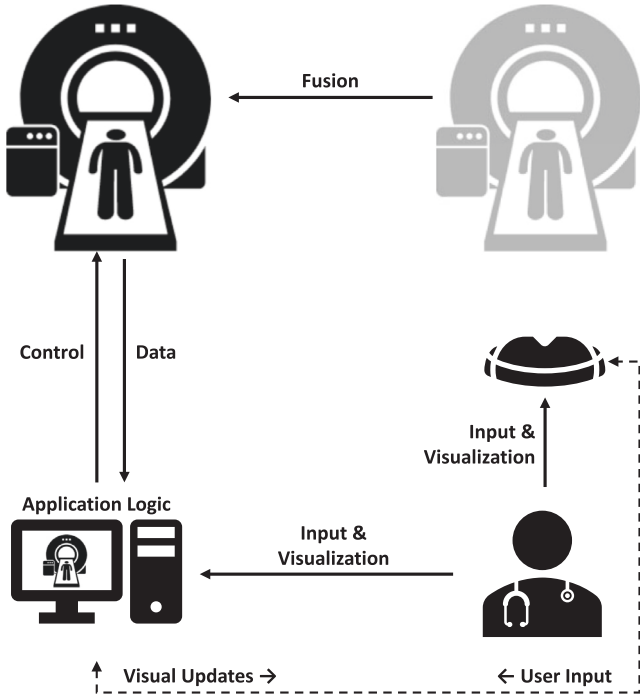


Fig. 1. The four primary entities of the proposed data/command pipeline deployed in situ in an imager suite: imaging scanner, processing apparatus, at least one holographic or 2D visualization display, and the operator(s). The operator interacts with the computational apparatus, as well as the imaging scanner to select data and adjust imaging parameters. If a holographic interface is used, this can be registered to the patient (a mixed reality version) or placed anywhere the operator prefers.

In addition to the objectives, our vision is for FI3D to be an enhancement to current clinical applications. Hence, the modular architecture of the framework we present could be used as a method to embed existing tools as part of FI3D. Because we understand that many of these tools are applicable and used by physicians in their current state, the system aims to provide these tools without the necessity of using an AR capable HMD. Furthermore, we think of FIs as visualization and user interaction aids that can be toggled by the physician as the need arises.

2.1. FI3D architecture

FI3D is a cross-platform framework, written in C++ and designed using object-oriented programming (OOP). It uses two main libraries: Qt [31] and the Visualization Toolkit (VTK) [32]. Qt is used for its GUI support and standard classes, such as strings, smart pointers, lists, etc. Additionally, Qt's signals and slots mechanism is used for event handling. VTK is used primarily for rendering and data processing algorithms, and the VTK package *vtkDICOM* is used for reading DICOM files into the system [33].

Fig. 2 illustrates the connectivity between FI3D, FIs, and external peripherals. Peripherals, such as data sources (on-line sensors and databases), processors (local or distributed), and controllers (sensors, interventional manipulators, agent injectors, etc.), are connected to the FI3D to be controlled and provide data. In FI3D, there are two self-contained pieces of code: specific applications called modules and module services called components. Modules use components to interface with peripherals and other generic uses, such as logging. The FI3DController oversees the execution of the entire framework, initiates the components, and instantiates the modules. The FIs connect to FI3D and module instances using the underlying framework to exchange data and visual information using a messaging scheme through a TCP/IP connection. FI3D has a registry used by components and modules to

register themselves, making themselves visible for the entire system. The self-registration design is important because it enables the developer to customize the software for the end-user. FI3D can be configured in the build process to enable/disable components and/or modules depending on the needs of the end-user. Furthermore, FI3D does not have any previous information on their existence, with the exception of certain default components used internally, such as the *DataManager* to manage data and the *Server* to communicate with FIs.

FI3D provides the user with a GUI as shown in Fig. 3. The developer of the component or module has complete flexibility of its GUI. The holographic scene visualized in the FIs can also be rendered in the module's GUI, serving three purposes: (i) the operator only has to use the FIs when the need arises (such as the need of true 3D visualization), (ii) one operator may adjust the visuals in FI3D while a second operator views the holographic scene in an FI, (iii) multiple FIs are connected and use the module's scene as the synchronization point.

2.2. Component design

Since FI3D does not know about components until run time, the component must register itself. To streamline the creation of components, we provide the FI3D_COMPONENT macro which declares and defines the required functions for its self-registration. FI3D automatically initiates and cleans the component's resources when the application is executed and closed, respectively. Furthermore, the component design ensures that only one component instance runs in the entire system, following the singleton pattern. This is enforced through the FI3D_COMPONENT macro.

By default, two components are provided in FI3D. The *Server* is a component used to communicate with the FIs. On a new connection, the *Server* generates and assigns it a unique ID which must be included in every future request. The connected client is required to provide a password set by the user. This ensures that only authorized FIs can manipulate the system. The *Server* uses the TCP protocol, and every message sent and received is encoded using JSON [34] and must follow the format illustrated in Fig. 4.

The *DataManager* component is available for modules to register and retrieve data. Furthermore, it handles all data requests from FIs. Modules may register data in the *DataManager* by providing the data object and a unique ID. There are three major types of data objects: *Model*, *Imaging*, and *Study*. *Model* instances represent polygon meshes whereas *Imaging* instances represent 3D arrays, where each 2D sub-array defines an imaging slice. Lastly, a *Study* is a set of *Imaging* instances. *Imaging* instances that are part of a *Study* are called *Series*. For example, in Fig. 5, the data set illustrated consists of 25 frames with 10 slices of MRI each, i.e., 25 *Series* for one *Study* with a total of 250 image slices. Generally, *Study* instances are always used, but *Imaging* instances are used if one wishes to render a simple image, such as a JPG or PNG. The *DataManager* is also able to load DICOM data from disk and create *Study* instances based on the read data.

2.3. Module design

Just like components, modules are registered in the system at run time. Each module has a unique name and acronym used across the system to identify the module. Similarly to components, the FI3D_MODULE macro is provided to streamline the registration of a module. Unlike components, however, modules may have several instances running at once, so each running instance is given an ID used to identify the instance in the system.

A module is comprised of several objects, each with its own responsibility and purpose, as shown in Fig. 6. The FI3DController controls the module using the *ModuleHandler* as it is the entry

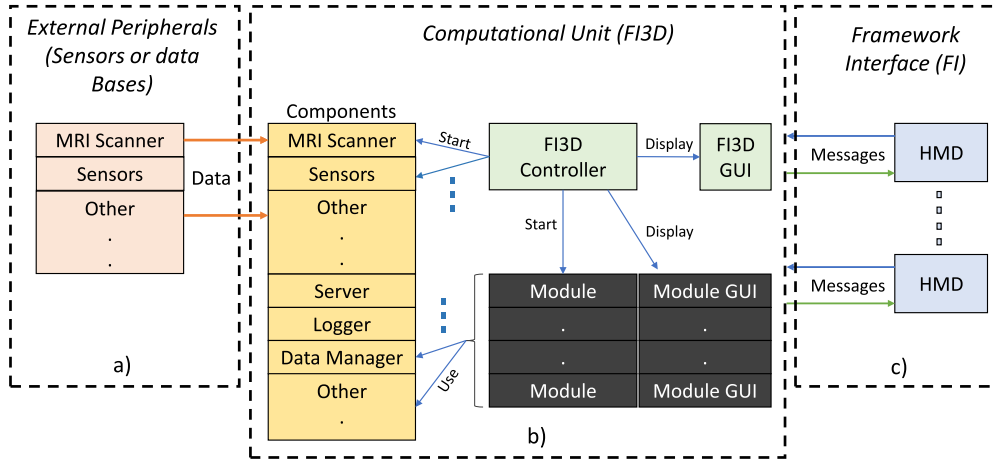


Fig. 2. The three foundational entities of the system: (a) the pool of peripherals, i.e., external to the framework, (b) the FI3D framework and its self-contained pieces of code, modules and components, used to customize the functionalities, and (c) the FIs connected to the FI3D.

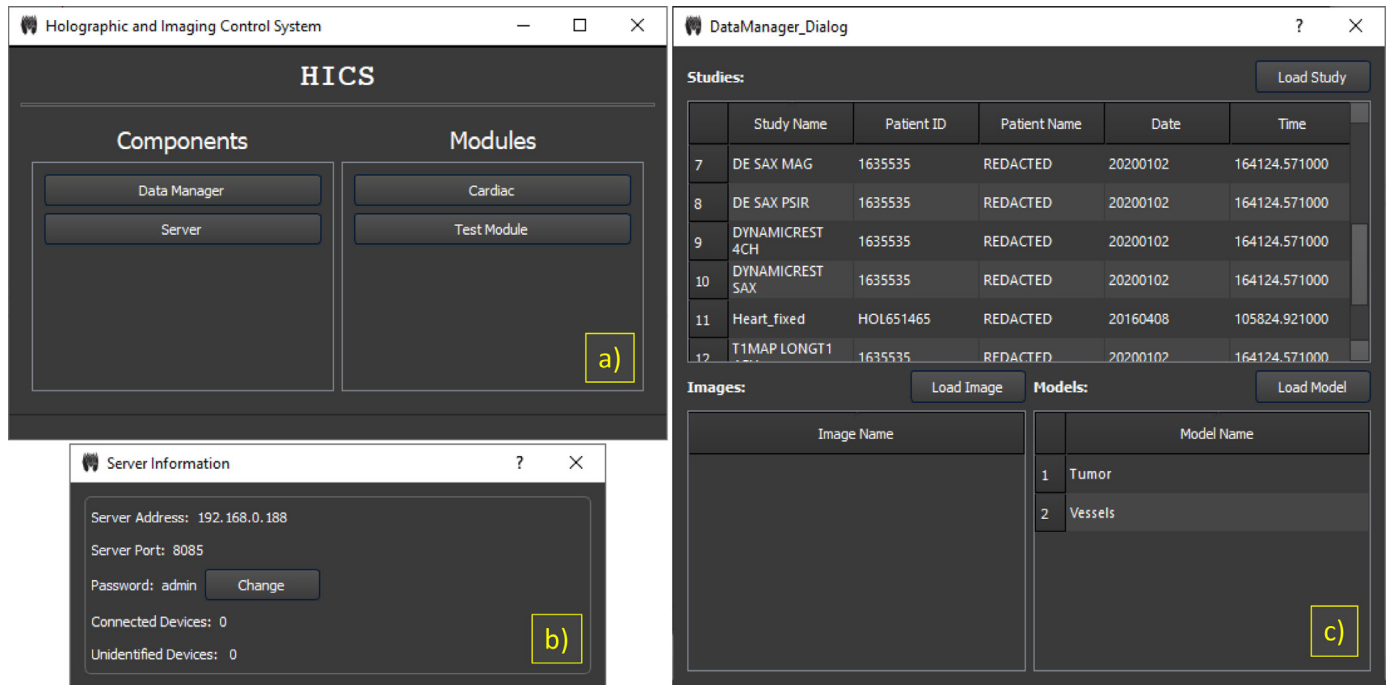


Fig. 3. The GUI of the platform and its two default components. The main GUI, shown in (a), lists the components that are configurable through a GUI and the registered modules. The GUI in (b) is the GUI of the Server component which shows the server information and gives the user the ability to change the password of the Server. In (c), the GUI of the DataManager is shown; it lists all the data objects available in the system. The top list shows the studies, along with study information. The two bottom lists show the 3D imaging sets and model sets from left to right, respectively. The user can add data using the GUI by clicking on the corresponding "load" buttons.

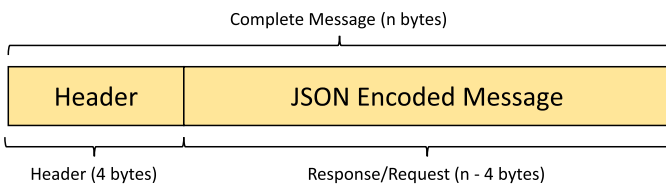


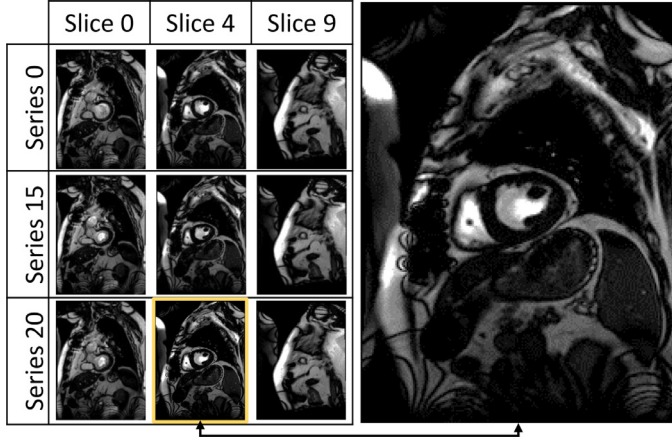
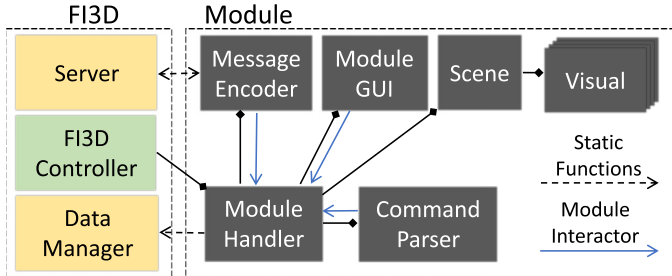
Fig. 4. The common message format used in all requests and responses exchanged between FI3D and FIs. Every n-bytes long message starts with a 4-byte integer header which defines the length of the complete message. The (n-4) bytes long body of the message may include either authentication information, FI3D system information, module information, or visualization data.

point of the module and contains the module's main logic. Furthermore, the *ModuleHandler* has no restrictions and can do any-

thing the module's designer needs it to do. All the inputs a module can receive are called module interactions; they are defined by the *IModuleInteractor* interface. These interactions are structured as events and are available to every object that implements the interface. The *MessageEncoder*, *ModuleGUI*, and *CommandParser* are module interactors because they can interact with the module by modifying visuals and changing module parameters. The *ModuleGUI* maps UI events to module interactions, while the *CommandParser* maps text-based commands to them. The *MessageEncoder* maps requests from FIs to the module interactions. The *ModuleHandler* links the interactors' events to functions that handle the event's input. As a result, the event, independently of the object that triggered it, produces the same result. By default, the *IModuleInteractor* interface provides every module with interactions that manipulate the *Scene*, and new modules may derive their own *IModuleInteractor* to add module-specific interactions.

Table 1
3D visual properties.

Property name	Data type	Description
Holographic	Boolean	Visualize in FI
Interactable	Boolean	Fixed position and orientation
Visible	Boolean	Show/Hide in scene
Opacity	Double	Visual's opacity
Transform	Double Array	Defines visual's position, orientation, and scale
Parent Visual	Visual3D	Optional <i>Visual3D</i> used as origin for this <i>Visual3D</i>

**Fig. 5.** Sample slices of MRI collected data from a beating heart. The data set consists of 10 sequential slice locations, each with 25 frames to represent the entire cardiac cycle.**Fig. 6.** The base elements that constitute a module and their relationship.

Every module should at least have one *Scene* instance. Each *Scene* manages all its rendered visuals. All *Visual* instances in a *Scene* have a unique ID. *Visual* instances are added to the *Scene*, and each *Visual* represents a single rendered object. There are two main *Visual* classes available: *ImageSliceVisual* and *ModelVisual*. Both classes derive from the *Visual3D* class which defines the properties shown in Table 1.

The *Transform* is an array of 16 double values, i.e., a 4×4 matrix, and represents the position, orientation, and scale of the *Visual3D* with respect to the world space, or the *VisualParent* if not null. An *ImageSliceVisual* is used to render one single slice of imaging data. The rendered *ImageSliceVisual* can be any 2D-subarray from the *Image* or *Series* in any of the three orientations: transverse, sagittal, or coronal. The third-dimension index corresponding to the 2D-subarray defines the slice index. Thus, a slice is identified by providing the *Image* or *Series*, its slice index, and orientation. In addition, if the slice to be rendered belongs to a *Study*, the *Series* index in the *Study* is also used to identify the slice. A *ModelVisual* is used to render one polygon mesh.

The *Scene* triggers an event every time a *Visual* is modified in any way. The *MessageEncoder* listens for these events and prepares a new message that is sent to connected FIs after the event is trig-

Table 2
Module interaction types for FIs.

Interaction data type	Description
Valueless	Triggered only, no input
Boolean	True or false input
Integer	Numeric integer input
Float	Numeric float input
String	Character array input

Table 3
Constraints for module interactions.

Interaction constraint	Description
Min	Has a minimum limit
Max	Has a maximum limit
Range	Has a minimum and maximum limit
Select	Must select from a given list

gered. Because FI3D does not have any information on modules until runtime, the module must provide the type of interactions that an FI may have to manipulate the module, i.e., it must inform the FI of the available module interactions. As a result, the module's designer must also define the module interactions in the *MessageEncoder*. Each module interaction has a unique ID, a value data type, a value constraint, and a default value. Table 2 lists the interaction data types available, and Table 3 lists the corresponding interaction constraints.

By combining an interaction data type and constraint, the programmer can define a variety of inputs for the holographic interface. Ideally, each module interaction defined in the *MessageEncoder* should map to a module interaction defined in the *IModuleInteractor* interface.

2.4. FI communication and control

FIs are generally programmed, i.e., they do not have any pre-determined information or programming about available modules and their module-specific logic. Hence, an FI's access to the modules and their information is granted by FI3D through the messaging protocol. There are four different types of messages: authentication, application, module, and data. Authentication messages are used by the FI3D's *Server* component to handle the authentication process of new connections. Once an FI is authenticated, FI3D automatically sends an application message that includes two lists: registered modules and running module instances. FIs can use this information to either start or stop a module. There are many types of module requests and responses. Table 4 describes the requests available to all FIs.

An FI must first subscribe to the module it intends to manipulate, i.e., it must send a *SubscribeToModule* request. Subsequently, the *MessageEncoder* begins to send module information to the subscribed FI. Once subscribed, the FI may send requests to manipulate the module. Table 5 describes the type of responses FI3D sends as a result of changes to the *Scene*.

Table 4
FI requests.

Module request	Description
Module Interaction	The request maps to a module interaction
Subscribe to Module	Begins to receive information about the specified module
Unsubscribe from Module	Stops receiving information about the specified module
Get Scene	Gets all scene information about the subscribed module
Get Object	Gets all information about a single object
Hide Object	Requests to hide an object
Translate Object	Requests to translate an object by x, y, z
Rotate Object	Requests to rotate an object by x, y, z
Select Slice	Requests to change the slice of an <i>ImageSliceVisual</i>
Select Orientation	Requests to change the orientation of an <i>ImageSliceVisual</i>
Select Series	Requests to change the series of an <i>ImageSliceVisual</i>

Table 5
FI3D responses.

Module response	Description
Module Information	Sends all module information: scene and module interactions
Refresh Scene	Sends all scene information
Update Module Interaction	Updates the state of a module interaction
Add Visual	Adds a visual to the holographic scene
Update Visual	Updates all properties of a visual, including data
Refresh Visual	Updates all properties of a visual, excluding data
Remove Visual	Removes a visual from the holographic scene
Hide Visual	Hides/Shows a visual
Transform Visual	Translates, rotates, and/or scales visual
Parent Change	Changes the visual's parent
Set Visual Opacity	Changes the opacity of the visual
Set Slice	Changes the rendered slice
Set Model Color	Changes the color of the model

Because the interactions applied to a module occur through the *IModuleInteractor* interface, any modification to the visuals, independently of where the input comes from, will automatically update the subscribed FIs. Furthermore, the connection between FI3D and FIs is stateless, i.e., FI3D does not keep up with what data and visuals are currently being visualized in the subscribed FIs. Instead, FI3D generates and delivers a message every time a visual's property has changed. Module messages do not include the visuals' data. Rather, the module messages include the ID of the data object being rendered by the *Visual*. The FI can use this information to request the data using a data request. Visual information and data are decoupled because it is possible for multiple *Visual* instances to visualize the same data object, preventing the delivery of data multiple times. Data messages are simple but vary in size depending on the data's resolution. When the *DataManager* receives a data request, it sends it to the requesting FI. Afterwards, the FI decodes the response and uses it in conjunction with the *Visual* information previously received to render the hologram. In the event that the FI loses connection with the server, the FI will repeatedly try to reconnect with the server. Once the connection has reestablished, the FI automatically subscribes to the module again, resulting in a *Scene* update from FI3D. An example of the communication between the two devices can be observed in Fig. 7. This design is tuned for scalability so that multiple HMDs can be used. Subsequent devices may connect to the FI3D by providing the IP, port, and password of the system. Because the FIs are synchronized with the module, automatically all FIs will receive and display the same information in/at their corresponding environment/location.

3. Microsoft's HoloLens Gen 1 as an FI

To evaluate the system's messaging protocol, we implemented an application, as illustrated in Fig. 8, with Unity [35] using Microsoft's Mixed Reality Toolkit (MRTK) [36] and deployed it to the Microsoft's HoloLens Gen 1. The design itself is similar to that of FI3D, with certain differences due to its nature. These differ-

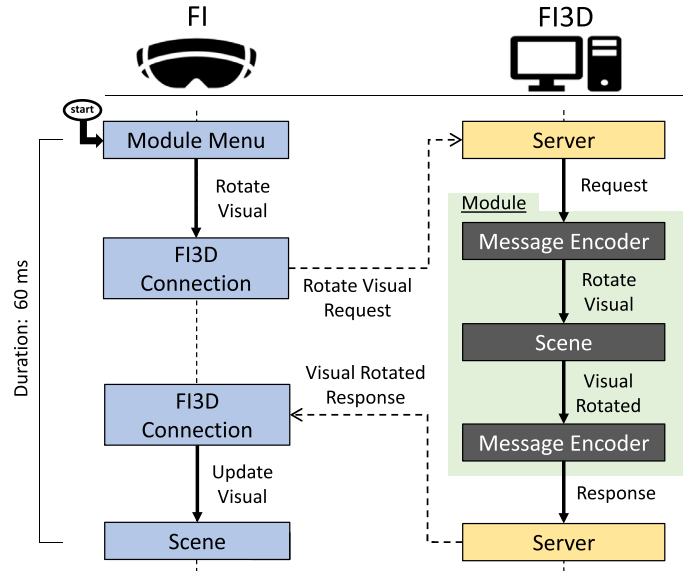


Fig. 7. Chart showing the synchronization and communication flow between FI3D and one FI. In this figure, the FI rotates a visual in the holographic scene, resulting in a message delivered to FI3D. Subsequently, the module for which this message belongs to executes the message and updates the FI with the new visual information.

ences ensure that instead of using a GUI, we implemented holographic menus and no module-specific logic exists. Rather, Unity's *GameObject* instances are used to set up the connection with FI3D, handle messages, and dynamically create visuals based on the received module information. The user interacts with the application through a set of holographic menus, as shown in Fig. 9. Upon subscription to a module instance, a holographic scene based on the visual information received from FI3D and a holographic mod-

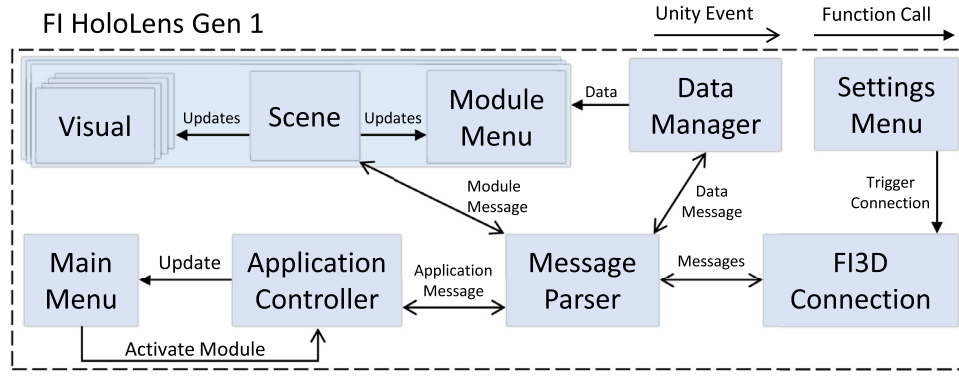


Fig. 8. Design of the FI implementation using the Microsoft's HoloLens Gen 1 HMD.

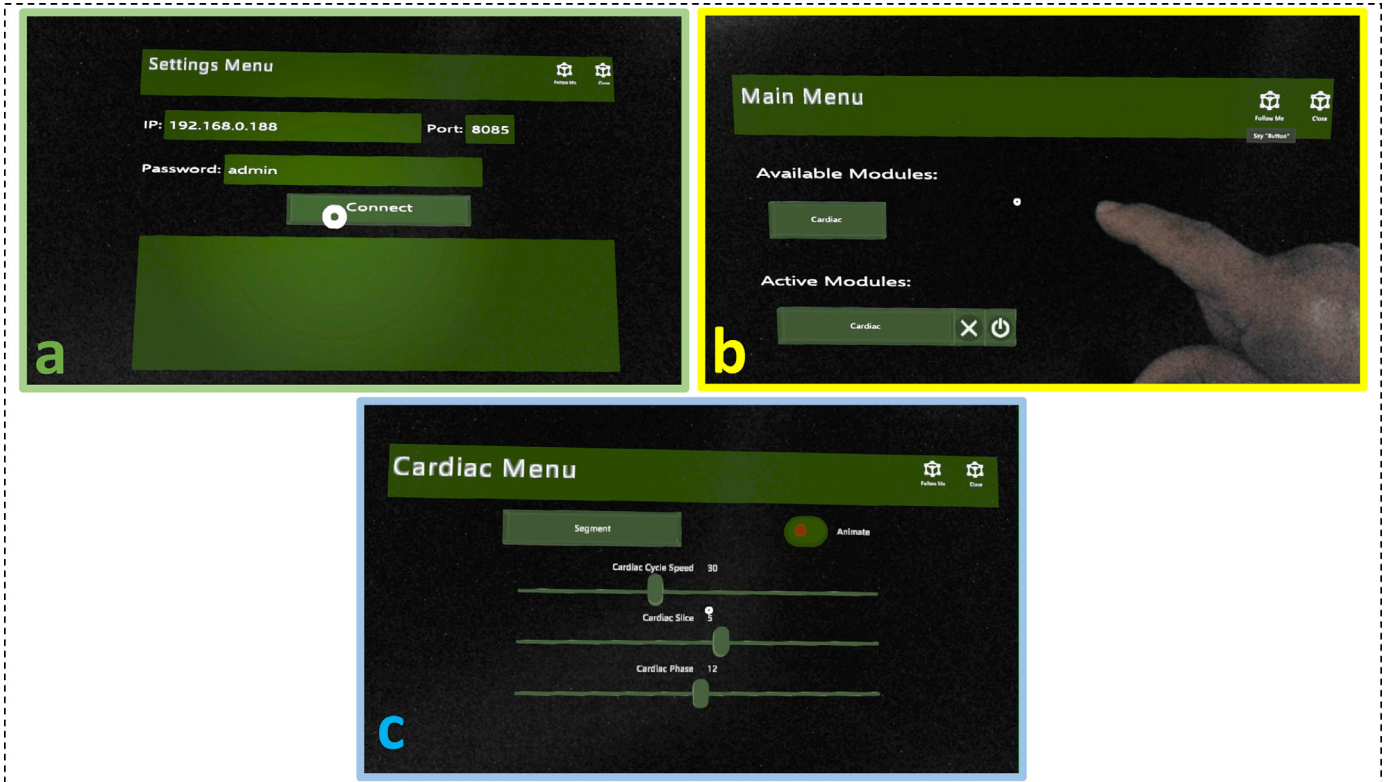


Fig. 9. Illustration of all three holographic menus. In (a), the settings menu to enter the IP, port, and password of the system and establish a connection. The module menu in (b) lists the available modules and the active module instances which the user can subscribe to. A module menu in (c) is populated with the holographic UI elements necessary for the module interactions. All three images were captured using the HoloLens Gen 1 built-in capture mechanism. The visual quality of the menus is much better than what is shown in these captures.

Table 6
Module interaction to holographic GUI map.

Interaction Type	Interaction Constraint	GUI Element
Valueless	N/A	Button
Boolean	N/A	Toggle
Integer or Float	Min or Max	Spin Box
Integer or Float	Range	Slider
String	Min, Max, or Range	Input Box
Integer, Float, or String	Select	Drop-down Box

ule menu are instantiated. The module interactions and their constraints are mapped to different holographic GUI elements in the holographic module menu, as described in Table 6.

Once the holographic scene is set up, the data required for the Visual instances is requested from FI3D. Once FI3D responds with

the data, the application creates the polygon mesh or texture, for models and slices, respectively, and assigns it to the corresponding Visual. Every time a new module message is received, the holographic scene automatically applies the change to the corresponding Visual.

4. Heart segmentation module

In the cardiovascular clinical room, a procedure often conducted by physicians for diagnosis involves the manual segmentation of the left ventricle, a constitute of the heart in charge of pumping blood. Currently, physicians perform the segmentation by manually marking in MRI scans the inner wall, known as the endocardium, and the outer wall, known as the epicardium, of the left ventricle. To assess the system and the implementation described in Section 3, we implemented the “Cardiac” module tasked

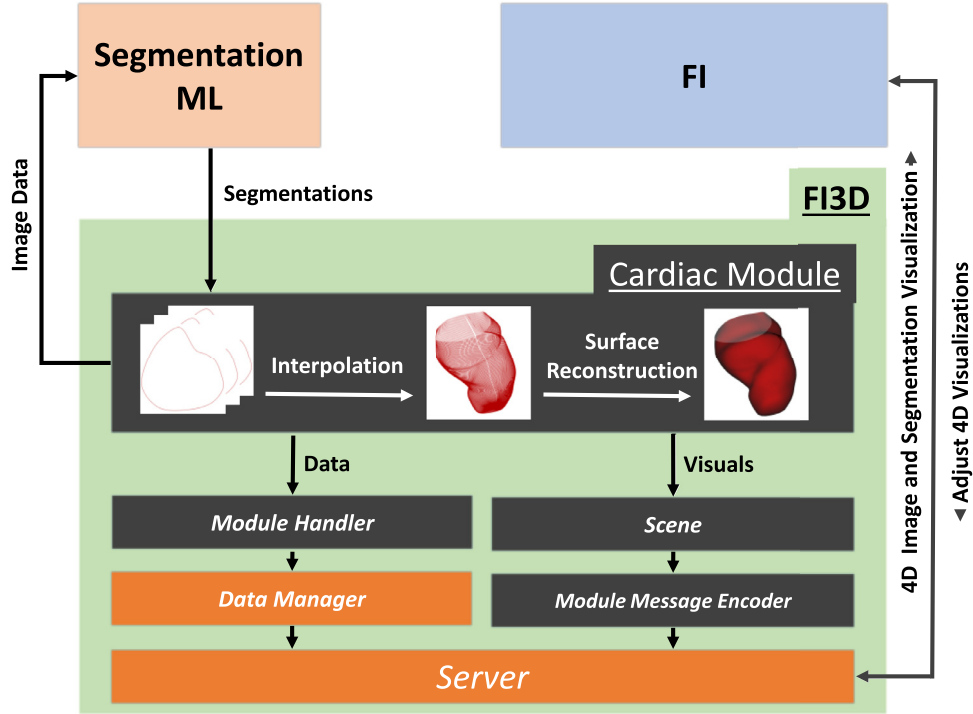


Fig. 10. The cardiac module sends cardiac imaging data to a machine learning segmentation model which process the data and returns segmentation masks of each slice. With the segmentation masks, the surface of the endocardium and epicardium is reconstructed by first interpolating using splines with VTK's `vtkParametricSpline` algorithm and reconstructing the surface using VTK's `vtkSurfaceReconstructionFilter` algorithm. This is repeated for each cardiac phase to create a 4D animation of the left ventricle. In parallel, the module performs standard hemodynamic calculations, such as end-diastolic volume (EDV), end-systolic volume (ESV), stroke volume (SV), ejection fraction (EF), and endocardium's volume vs time. The animated visualization is sent to the module subscribers which use the information to replicate it as a holographic scene.

to segment and visualize the left ventricle's endocardium and epicardium, as illustrated in Fig. 10. The machine learning (ML) model described in [30] was implemented as a Python script. The communication between the module and the ML model is established through a TCP connection, giving us the flexibility to run the model either locally or in a remote machine. This gave us the option to either run the ML model locally or on a separate Linux-based server configured to use the GPU to segment the images. The ability to switch between these two modes was a consideration primarily for the implementation of the Heart Segmentation module and not the overall FI3D system itself. Therefore, we did not evaluate the performance between the module and the ML model but rather the interfacing of the module and the underlying FI3D framework. Then, we set all the *Visual* instances required in the scene: segmentation masks, interpolated segmentations, reconstructed surfaces, and image slice visuals. Using Qt, we designed the GUI as illustrated in Fig. 11. Lastly, we set the module interactions in the module's *MessageEncoder* to give the same interactions given in the GUI to the holographic interface, as shown in Fig. 9c.

Using this module, we used the MR-collected images from Fig. 5 for the segmentation task. Once the learner completed the segmentation, it sent back the segmentation boundaries of the 10 slices for all 25 frames. For each cardiac phase, the surfaces of the endocardium and epicardium are reconstructed. Once this process is complete, the module visualizes one frame of data, i.e., the frame's image slice, segmentation boundaries, interpolated segmentations, and reconstructed surfaces. Additionally, the module updates the ventricle's volume over time graph. Simultaneously, the HoloLens application subscribes to the "Cardiac" module and receives the visuals. The scene in FI3D and holographic scene in the HoloLens FI are shown in Fig. 12. The module's functionalities along with the FI visualization of the holographic scene can be observed in the attached Video 1.

5. Performance analysis

Based on the implementation of the HoloLens FI and the "Cardiac" module, we can highlight various performance details of the system. FI3D enabled us to use machine learning, data interpolation, and surface reconstruction algorithms that were already in pre-existing C++ libraries and provided enough resources to perform these in a timely manner for holographic visualization, otherwise unachievable directly in a single HMD implementation. However, by offloading the work onto a separate unit, a communication task between the two entities must now be taken into consideration as part of the system's performance. To assess this, we tracked the number of messages queued in an FI as the speed of the left ventricle's animation changed. The system is stable and synchronized if the messages received from FI3D in the FI are processed immediately, i.e., the number of queued messages should not increase over time. System evaluations were conducted with the FI3D executed on a PC running Windows 10 with 16 Gb RAM and Intel i7-6700k Skylake Quad-core 4.0 GHz processor. The HoloLens communicated with the FI3D via a 2.4 GHz WiFi connection using a Tp Link Archer C5400 gigabit router. With these settings, the FI implementation in the HoloLens Gen 1 could execute 1 visual update every 16 milliseconds or 62 visual updates per second, excluding data requests. Additionally, the time required for the FI to send a module interaction request to the FI3D, execute the request by the FI3D module, generate and send a response to the FI, and apply the response to the affected visual takes 60 ms, as shown in Fig. 7.

There were no delays observed in the FI3D implementation of the "Cardiac" module, aside from the interpolation and surface reconstruction of the segmentations. The total time taken to reconstruct these surfaces was 7 s, but this can be further improved by parallelizing each frame and using GPU acceleration for the in-

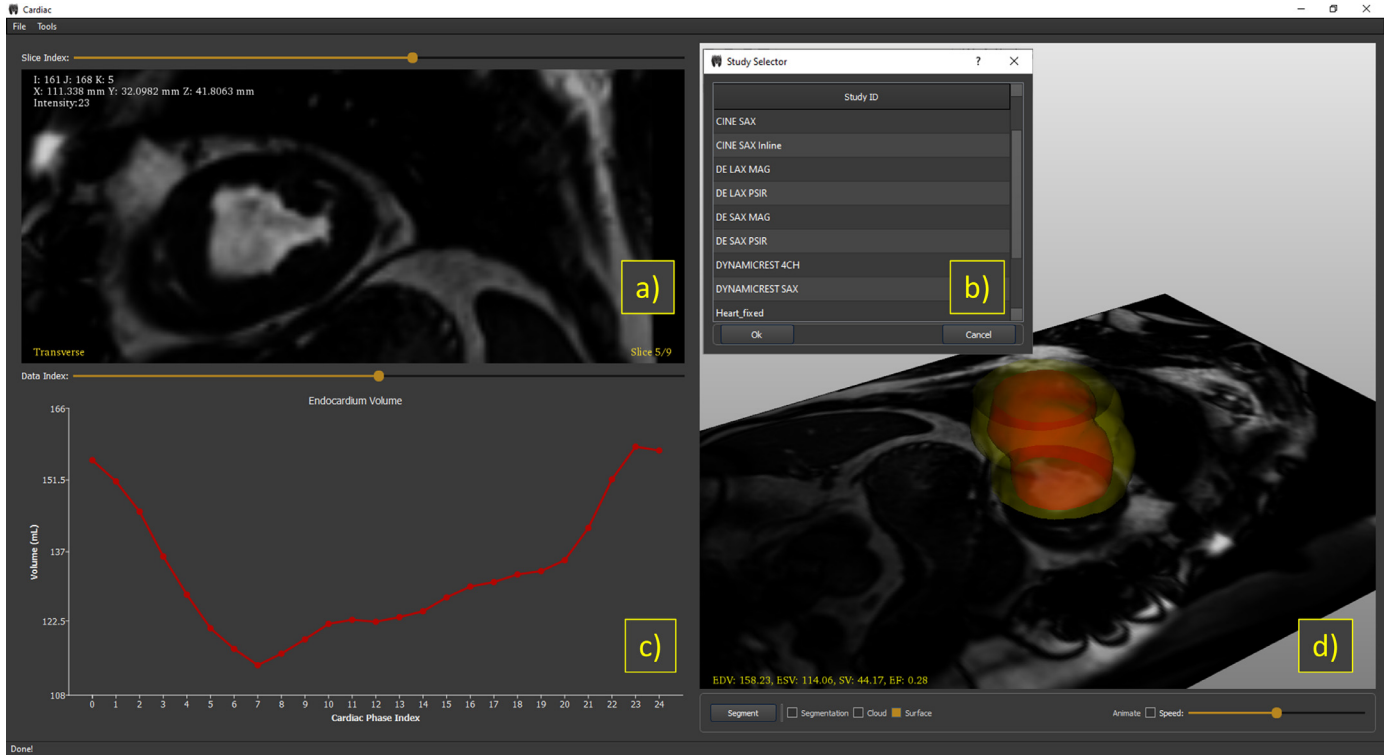


Fig. 11. The GUI of the “Cardiac” module. (a), A representative 2D Scene used to render 2D imaging data; the user can select the series and slice indices with the provided sliders and drag the mouse over the image to get the intensity value and coordinates of the cursor relative to the MRI scanner coordinate system. (b), The “Study Selector” window is shown which is activated in the “File” menu on the toolbar and shows the available studies that can be loaded into the module. (c), A graph illustrating the calculated volume of the endocardium at each cardiac phase. (d), The main Scene is rendered and visualizes the reconstructed segmentations, the cardiac image, and a label with the hemodynamic calculations. This is the Scene that is replicated by HMDs as a holographic scene. Under the Scene, the module interactions that start the segmentation routines, enable/disable visuals, toggle animation, and change animation speed are available.

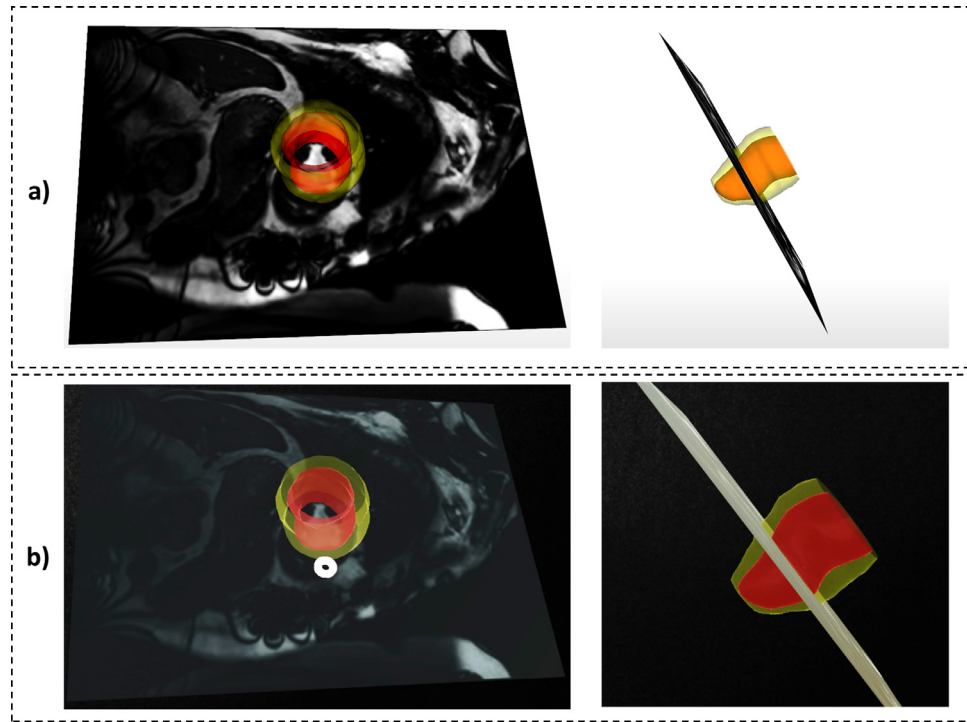


Fig. 12. Representative captions of the holographic scene, from different perspectives, as rendered in (a) FI3D and (b) FI's implementation in the Microsoft's HoloLens Gen 1. Both scenes belong to the same “Cardiac” module and are synchronized using FI3D's base classes.

Table 7
Image transfer rates from FI3D to FI.

Resolution –100 images (px)	Total size of 100 images (MB)	Average time (s)	Std. dev. (s)
8 × 8	0.08	5.83	0.13
16 × 16	0.23	6.02	0.15
32 × 32	0.85	6.52	0.06
64 × 64	3.30	7.64	0.06
128 × 128	13.14	12.03	0.17
256 × 256	52.46	31.39	0.89
512 × 512	209.75	102.87	2.05

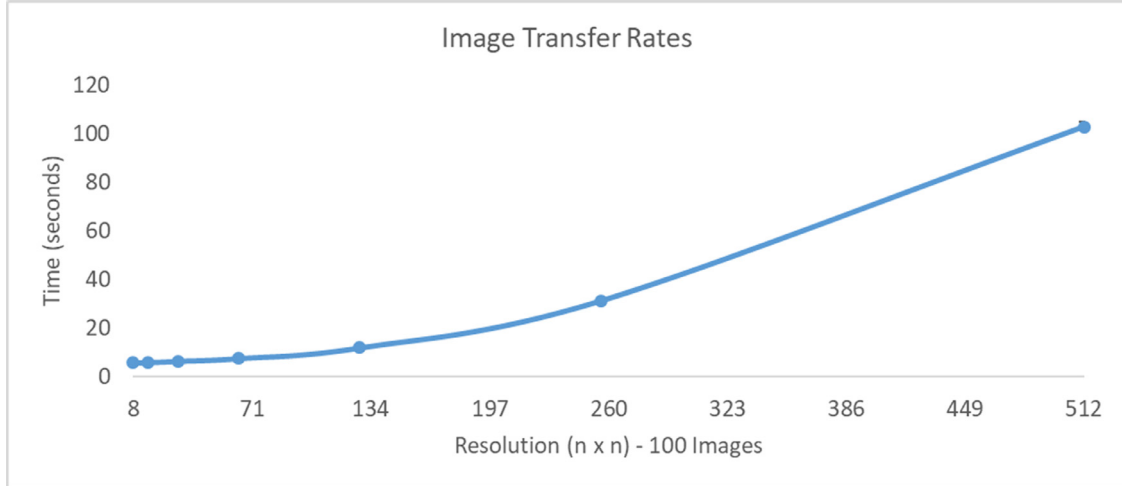


Fig. 13. Time required to transfer MRI sets from the FI3D to an FI and render them using the Microsoft's HoloLens FI implementation for images with a resolution of $n \times n$ (each data set consists of 100 images with $n = 8, 16, 32, 64, 128, 256$, and 512). Each data point is the average time of five trials. Five trials were selected to measure the variability of the network, which was minimal. The measured process included: (i) encoding the image, (ii) transferring the image from FI3D to FI, (iii) decoding the image, (iv) creating the texture, and (v) rendering the texture in 3D space as a hologram, repeatedly 100 times continuously one image after the other.

terpolation and surface reconstruction computations. Furthermore, using this method, the HoloLens visualization was not affected by these background computations, resulting in less battery consumption and overall better user experience.

The visualization of medical imaging is an important aspect of the system as well. Therefore, we also analyzed the amount of time needed to visualize medical images as holograms in the FI HoloLens implementation. To do so, we performed various trials that involved sending images of various resolutions to the FI using a TCP connection. The results are summarized in Table 7 and illustrated in Fig. 13. From the 512×512 resolution results for 100 images, one can see that roughly one image with this resolution can be transferred and visualized per second.

6. Discussion and conclusions

This work presents the first step towards implementing a scalable and modular computational system for generating and manipulating holographic AR scenes. Central to this work is the goal to support high performance computing, integration of current and future libraries used in medical imaging and surgeries, HMD independence, and open I/O for directly linking to the MRI scanner. The objective of this platform is to become a human-centered interface to immerse the clinician in the imaging data as well as enable control of sensors (that generate the data) and software (that process and analyze data). The approaches and architecture were adopted to offer a distributable system and address computational limitations of HMDs. The distributive property of the system sets the direction for the FI3D to run in remote locations, including having several instances of the system running at different workstations. This enables the user to dedicate the computational power required for a particular task without affecting other processes. Al-

though this is not yet in the current version of the system, the goal is to have the option to run the FI3D as a cloud service. We designed the system to have certain features, such as but not limited to: interactive real-time manipulation of data acquisition from medical devices such as MRI scanners, interactive segmentation of anatomical structures from imaging data, simulation of surgical tools, assistive diagnosis and segmentation with machine learning, and registration of patient, imaging data, and segmented structures using an AR interface. Such seamless and interactive integration of the physician with the data and sensors (i.e., MRI scanner) may enable us to move towards a new modality where AR through HMDs is used to fully visualize patient data in 3D and interactively change parameters to collect and analyze new relevant data. Additionally, the FI3D is tuned for scalability so that multiple FIs can connect at once to receive the same information. Because changes are first applied to the module, all connected FIs will receive the most up-to-date information. Therefore, it is possible for multiple FIs to visualize the same scene in their corresponding environment. However, we are yet to test these scenarios as there are other factors and case studies to consider, such as co-registration of holograms across devices.

The presented work includes preliminary demonstrations of using the FI3D platform in immersive and interactive visualization of 3D and 4D medical imaging datasets. While functioning as a proof-of-concept, these early studies were of focused purpose and, thus, of limited extent regarding two aspects. First, only five clinical personnel contributed in setting the technical specifications of the FI3D platform for three clinical paradigms: brain [20,21], prostate [22,28,29], and cardiac [30]. In addition, subjective assessment of ergonomics and functionalities from end-users (14 medical personnel and 18 engineers/scientists) was used for minor improvements on the front-end and interfacing. Second, this work did not in-

clude a quantitative assessment of functionality and ergonomics of the platform. These focused-purpose works served in (i) assessing technical limitations and (ii) as a valuable roadmap for setting current developmental thrusts. Consequently, we are currently focused on three areas: (a) enhance human-in-the-loop data analytics, (b) speed up processing, and (c) fusion of holographic visuals onto the real objects. To achieve these, we will further expand image processing and analytics tools with the operator interfacing via the HMD, incorporate multi-threaded and GPU acceleration, and experiment with sensors to co-register patient data with the real environment. Upon the completion and successful bench-top testing of the next version of the framework, we plan to pursue pre-clinical multi-site quantitative studies to systematically investigate functionality and ergonomics, as well as compare them with current clinical practices.

This work further supports conclusions drawn by prior efforts, e.g., [1,3–5,7,14,19,20,24] and references therein, that holographic visualization with HMDs is a very promising technology toward a practical bedside immersion of the physician into medical imaging and image-based patient specific renderings. Second, it introduced a computational platform for implementing human-machine and human-information interfacing to perform computing via an HMD. It should be noted, however, that eventual adoption of such immersive human-computer interfaces will be determined in the clinical realm. Clinical adoption is based on three criteria: patient outcome, learning curve, and cost-effectiveness. Accordingly, the underlying conclusion of our early studies with the FI3D was the same: using an HMD must be seamlessly integrated in the daily workflow of a clinical site, which translates to intuitive and ergonomic front-ends that reduce workload, streamline a procedure, and avoid errors. Further developments on HMDs may prove its merit and contribution to improve patient outcome.

Hardware and software specifications

FI3D can be executed in 32/64-bit version of Windows, Mac OS, and Linux. To build the source, Qt, VTK, and the vtkDICOM package is required. The FI implementation was implemented using Unity 3.6.8 and the MRTK 2.0 RC1.

Mode of availability

FI3D adheres to Qt's license [31], and FI's HoloLens implementation adheres to Unity's license [35]. The framework is currently in a private repository but is planned to be available publicly in the future. For current availability email NVT (nvtsekos@central.uh.edu) or JDV (jdelazco@uh.edu).

Declaration of Competing Interest

The authors of this submission have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Acknowledgments

This work was supported by the National Science Foundation awards CNS-1646566, DGE-1746046, and DGE-1433817. All opinions, findings, conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect the views of our sponsors.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cmpb.2020.105779.

References

- [1] A.D. Chen, S.J. Lin, Discussion: mixed reality with HoloLens: where virtual reality meets augmented reality in the operating room, *Plast. Reconstr. Surg.* 140 (2017) 1071–1072 <https://doi.org/10.1097/PRS.00000000000003817>.
- [2] F. Cutolo, C. Freschi, S. Mascioli, P. Panchi, M. Ferrari, V. Ferrari, Robust and accurate algorithm for wearable stereoscopic augmented reality with three indistinguishable markers, *Electronics (Basel)* 5 (2016) 59 <https://doi.org/10.3390/electronics5030059>.
- [3] M. Kersten-Oertel, I.J. Gerard, S. Drouin, K. Petrecca, J.A. Hall, D. Louis Collins, in: *Towards Augmented Reality Guided Craniotomy Planning in Tumour Resections*, Springer, Cham, 2016, pp. 163–174. https://doi.org/10.1007/978-3-319-43775-0_15.
- [4] K.H. Kim, The potential application of virtual, augmented, and mixed reality in neurology, *Int. Neurol.* 20 (2016) 169–170 <https://doi.org/10.5213/inj.1620edi005>.
- [5] I. Kuhlmann, M. Kleemann, P. Jauer, A. Schweikard, F. Ernst, Towards X-ray free endovascular interventions – using HoloLens for on-line holographic visualisation, *Healthc. Technol. Lett.* 4 (2017) 184–187 <https://doi.org/10.1049/htl.2017.0061>.
- [6] L. Qian, A. Barthel, A. Johnson, G. Osgood, P. Kazanides, N. Navab, B. Fuerst, Comparison of optical see-through head-mounted displays for surgical interventions with object-anchored 2D-display, *Int. J. Comput. Assist. Radiol. Surg.* 12 (2017) 901–910 <https://doi.org/10.1007/s11548-017-1564-y>.
- [7] O.M. Tepper, H.L. Rudy, A. Lefkowitz, K.A. Weimer, S.M. Marks, C.S. Stern, E.S. Garfein, A.D. Chen, S.J. Lin, Mixed reality with HoloLens: where virtual reality meets augmented reality in the operating room, *Plast. Reconstr. Surg.* 140 (2017) 1066–1070 <https://doi.org/10.1097/PRS.00000000000003802>.
- [8] R.W. Brown, Y.C.N. Cheng, E.M. Haacke, M.R. Thompson, R. Venkatesan, *Magnetic Resonance Imaging: Physical Principles and Sequence Design*, second ed., John Wiley & Sons Ltd, Chichester, UK, 2014 <https://doi.org/10.1002/9781118633953>.
- [9] F.K. Wacker, D. Elgort, C.M. Hillenbrand, J.L. Duerk, J.S. Lewin, The catheter-driven MRI scanner: a new approach to intravascular catheter tracking and imaging-parameter adjustment for interventional MRI, *Am. J. Roentgenol.* 183 (2004) 391–395 <https://doi.org/10.2214/ajr.183.2.1830391>.
- [10] E. Christoforou, E. Akbudak, A. Ozcan, M. Karanikolas, N.V. Tsekos, Performance of interventions with manipulator-driven real-time MR guidance: implementation and initial in vitro tests, *Magn. Reson. Imaging* 25 (2007) 69–77 <https://doi.org/10.1016/j.mri.2006.08.016>.
- [11] M.S. Hansen, T.S. Sørensen, Gadgetron: an open source framework for medical image reconstruction, *Magn. Reson. Med.* 69 (2013) 1768–1776 <https://doi.org/10.1002/mrm.24389>.
- [12] J.D. Velazco Garcia, N.V. Navkar, D. Gui, C.M. Morales, E.G. Christoforou, A. Ozcan, J. Abinayed, A. Al-Ansari, A. Webb, I. Seimenis, N.V. Tsekos, A platform integrating acquisition, reconstruction, visualization, and manipulator control modules for MRI-guided interventions, *J. Digit. Imaging* 32 (2019) 1–13 <https://doi.org/10.1007/s10278-018-0152-1>.
- [13] M. Huellebrand, D. Messroghli, L. Tautz, T. Kuehne, A. Hennemuth, An extensible software platform for interdisciplinary cardiovascular imaging research, *Comput. Methods Programs Biomed.* 184 (2020) 105277 <https://doi.org/10.1016/j.cmpb.2019.105277>.
- [14] E. Gobetti, P. Pili, A. Zorcolo, M. Tuvieri, Interactive virtual angiography, in: *Proc. Vis. '98 (Cat. No.98CB36276)*, IEEE, 1998, pp. 435–438. <https://doi.org/10.1109/VISUAL.1998.745337>.
- [15] J.N.A. Silva, M.K. Southworth, A. Dalal, G. Hare, J.R. Silva, Improving visualization and interaction during transcatheter ablation using an augmented reality system: first-in-human experience, *Electrophysiol. Arrhythm. Nov. Tools Tech. Arrhythmia Ablation* 136 (2017) A15358 https://www.ahajournals.org/doi/abs/10.1161/circ.136.suppl_1.15358, accessed April 13, 2020.
- [16] J. Liu, S.J. Al'Aref, G. Singh, A. Caprio, A.A.A. Moghadam, S.-J. Jang, S.C. Wong, J.K. Min, S. Dunham, B. Mosadegh, An augmented reality system for image guidance of transcatheter procedures for structural heart disease, *PLoS ONE* 14 (2019) e0219174 <https://doi.org/10.1371/journal.pone.0219174>.
- [17] M. Mahvash, L.B. Tabrizi, A novel augmented reality system of image projection for image-guided neurosurgery, *Acta Neurochir. (Wien)* 155 (2013) 943–947 <https://doi.org/10.1007/s00701-013-1668-2>.
- [18] B.J. Park, S.J. Hunt, C. Martin, G.J. Nadolski, B.J. Wood, T.P. Gade, Augmented and mixed reality: technologies for enhancing the future of IR, *J. Vasc. Interv. Radiol.* (2020) 0 <https://doi.org/10.1016/j.jvir.2019.09.020>.
- [19] R. Rahman, M.E. Wood, L. Qian, C.L. Price, A.A. Johnson, G.M. Osgood, Head-mounted display use in surgery: a systematic review, *Surg. Innov.* 27 (2019) 88–100 <https://doi.org/10.1177/1553350619871787>.
- [20] C.M. Morales Mojica, N.V. Navkar, N.V. Tsekos, A. Webb, T. Biribilis, I. Seimenis, D. Tsagkaris, A. Webb, T. Biribilis, I. Seimenis, Holographic interface for three-dimensional visualization of MRI on HoloLens: a prototype platform for MRI guided neurosurgeries, in: *Proc. - 2017 IEEE 17th Int. Conf. Bioinform. Bioeng. BIBE 2017, IEEE, 2017*, pp. 21–27. <https://doi.org/10.1109/BIBE.2017.00-84>.
- [21] C.M. Morales Mojica, J.D. Velazco-Garcia, H. Zhao, I. Seimenis, E.L. Leiss, D. Shah, A. Webb, A.T. Becker, N.V. Tsekos, Interactive and immersive im-

- age-guided control of interventional manipulators with a prototype holographic interface, in: Proc. - 2019 IEEE 19th Int. Conf. Bioinforma. Bioeng. BIBE 2019, Athens, IEEE, 2019.
- [22] C.M. Morales Mojica, J.D. Velazco-Garcia, N.V. Navkar, S. Balakrishnan, J. Abin角度, W. El Ansari, K. Al-Rumaihi, A. Darweesh, A. Al-Ansari, M. Gharib, M. Karkoub, E.L. Leiss, I. Seimenis, N.V. Tsekos, A prototype holographic augmented reality interface for image-guided prostate cancer interventions, Proc. Eurographics Work. Vis. Comput. Biol. Med (2018) 17–21 <https://doi.org/10.2312/VCBM.20181225>.
- [23] J.W. Meulstee, J. Nijsink, R. Schreurs, L.M. Verhamme, T. Xi, H.H.K. Delye, W.A. Borstlap, T.J.J. Maal, Toward holographic-guided surgery, Surg. Innov 26 (2019) 86–94 <https://doi.org/10.1177/1553350618799552>.
- [24] J. Jang, C.M. Tschabrunn, M. Barkagan, E. Anter, B. Menze, R. Nezafat, Three-dimensional holographic visualization of high-resolution myocardial scar on HoloLens, PLoS ONE 13 (2018) e0205188 <https://doi.org/10.1371/journal.pone.0205188>.
- [25] N.V. Navkar, Zhigang Deng, D.J. Shah, N.V. Tsekos, A framework for integrating real-time MRI with robot control: application to simulated transapical cardiac interventions, IEEE Trans. Biomed. Eng. 60 (2013) 1023–1033 <https://doi.org/10.1109/TBME.2012.2230398>.
- [26] N.V. Navkar, Z. Deng, D.J. Shah, K.E. Bekris, N.V. Tsekos, Visual and force-feedback guidance for robot-assisted interventions in the beating heart with real-time MRI, in: Proc. - IEEE Int. Conf. Robot. Autom., Institute of Electrical and Electronics Engineers Inc., 2012, pp. 689–694. <https://doi.org/10.1109/ICRA.2012.6224582>.
- [27] Microsoft The Leader in Mixed Reality Technology | HoloLens, Microsoft, 2017 <https://www.microsoft.com/en-us/hololens> accessed November 16, 2019.
- [28] J.D. Velazco-Garcia, N.V. Navkar, S. Balakrishnan, J. Abin角度, A. Al-Ansari, G. Younes, A. Darweesh, K. Al-Rumaihi, E.G. Christoforou, E.L. Leiss, M. Karkoub, P. Tsiamyrtzis, N.V. Tsekos, Preliminary evaluation of robotic transrectal biopsy system on an interventional planning software, in: Proc. - 2019 IEEE 19th Int. Conf. Bioinforma. Bioeng. BIBE 2019, IEEE, 2019.
- [29] J.D. Velazco-Garcia, N.V. Navkar, S. Balakrishnan, J. Abin角度, A. Al-Ansari, A. Darweesh, K. Al-Rumaihi, E.G. Christoforou, E.L. Leiss, M. Karkoub, P. Tsiamyrtzis, N.V. Tsekos, Evaluation of interventional planning software features for MR-guided transrectal prostate biopsies, in: Proc. - 2020 IEEE 20th Int. Conf. Bioinforma. Bioeng. BIBE 2020, IEEE, 2020.
- [30] G. Molina, J.D. Velazco-Garcia, D. Shah, A.T. Becker, N.V. Tsekos, Automated segmentation and 4D reconstruction of the heart left ventricle from CINE MRI, in: Proc. - 2019 IEEE 19th Int. Conf. Bioinforma. Bioeng. BIBE 2019, IEEE, 2019.
- [31] The Qt Company Qt | Cross-Platform Software Development for Embedded and Desktop, 2019 (accessed July 23, 2019). <https://www.qt.io>.
- [32] Kitware, VTK - the visualization toolkit, (2019). <https://www.vtk.org/> (accessed July 23, 2019).
- [33] D. Gobbi, DICOM for VTK, (2019). <http://dgobbi.github.io/vtk-dicom/> (accessed July 23, 2019).
- [34] JSON, (2019). <http://www.json.org/> (accessed November 16, 2019).
- [35] Unity Technologies, Unity real-time development platform | 3D, 2D VR and AR visualizations, (2019). <https://unity.com/> (accessed September 24, 2019).
- [36] Microsoft, Mixed reality toolkit, (2019). <https://github.com/microsoft/MixedRealityToolkit-Unity> (accessed October 17, 2019).