A DirectX-Based DICOM Viewer for Multi-User Surgical Planning in Augmented Reality

Menghe Zhang, Weichen Liu, Nadir Weibel, and Jürgen P. Schulze
University of California, San Diego
{mez071, wel008, weibel, jschulze}@ucsd.edu

Abstract. Preoperative medical imaging is an essential part of surgical planning. The data from medical imaging devices, such as CT and MRI scanners, consist of stacks of 2D images in DICOM format. Conversely, advances in 3D data visualization provide further information by assembling cross-sections into 3D volumetric datasets. As Microsoft unveiled the HoloLens 2 (HL2), which is considered one of the best Mixed Reality (XR) headsets in the market, it promised to enhance visualization in 3D by providing an immersive experience to users. This paper introduces a prototype holographic XR DICOM Viewer for the 3D visualization of DICOM image sets on HL2 for surgical planning. We first developed a standalone graphical C++ engine using the native DirectX11 API and HLSL shaders. Based on that, the prototype further applies the OpenXR API for potential deployment on a wide range of devices from vendors across the XR spectrum. With native access to the device, our prototype unravels the limitation of hardware capabilities on HL2 for 3D volume rendering and interaction. Moreover, smartphones can act as input devices to provide another user interaction method by connecting to our server. In this paper, we present a holographic DICOM viewer for the HoloLens 2 and contribute (i) a prototype that renders the DICOM image stacks in real-time on HL2, (ii) three types of user interactions in XR, and (iii) a preliminary qualitative evaluation of our prototype.

Keywords: Volume Rendering · HoloLens 2 · Mixed Reality · 3D User Interaction · DICOM.

1 Introduction

Surgical planning with medical imaging in a DICOM format is a demanding process. Traditionally, viewing and planning on 2D screens make it challenging for clinical practitioners to extract 3D information such as anatomical structures, the regions of interest, and spatial relationships among the image set.

3D volumetric visualization in Mobile Augmented Reality (MAR) solves this problem by integrating the volumetric object with the physical environment for

* This research was funded in part by a grant from The Leona M. and Harry B. Helmsley Charitable Trust. The authors would also like to thank the volunteers who provided their medical data for us to work with under UCSD IRB protocol #182146.
mobile devices. Our previous work on Android developed an application to load, view, and interact with 3D medical volumes in traditional 2D screen and AR scenarios.

To further allow hands-free operations, we adapt it to HoloLens 2 with our graphical C++ MR rendering engine using the native DirectX11 API and HLSL shaders to boost the device’s performance. We also show three user interaction methodologies for different tasks: 1) we design our own interaction set (3D interactions) and interfaces (virtual 2D plane touch) for unique user interactions using OpenXR APIs; 2) with matched functionalities to our Android DICOM Viewer, we make Android phones as input devices to MR applications. Fig. 1 shows the categorized feature list. We will detail the implementation of the extended features in this section.

![Feature List](image)

Fig. 1: List of the functionalities on GLES-based and D3D-based applications.

### 2 Related Work

#### 2.1 Holographic DICOM Viewer Prototypes

Mojica et al. [9] proposed an AR DICOM Viewer prototype for the first generation of the HoloLens. To accelerate volume rendering, they only render the plane under a specific setting to avoid rendering the whole data volume. Remote rendering is another solution. Frohlich et al. [2] proposed it as part of a dedicated workstation with a wireless connection that enables real-time computation on MR headsets. Their latest work [10] is deployed in a two-CPU fashion: parts run
on the HoloLens and parts run on an external PC connected through Wi-Fi to enable real-time experience.

HoloDICOM [3] is a standard DICOM image stack visualization tool targets at the HoloLens 2. It contains tools for volume manipulation, a 2D slicer, and an intuitive floating 2D user interface. However, only one instructor can interact with the application, while others could only observe the interfaces.

2.2 Interaction with 3D Objects

3D selection in mid-air is commonly used in XR applications. However, interacting with far away objects and dense object selection still create problems in terms of user fatigue [4]. We have been exploring to apply 2D touch to achieve 3D selection in AR [14][15].

The idea of tuning smartphones info input devices for HMDs attracted the designers’ interest in various interactions. Karan et al. [5] enhanced the AR game experience by creating new interaction methods through a smartphone used as a game controller. Their remote input SDK makes connections between Unity applications developed for XR devices. Buschel et al. [1] contributes an exploration of smartphone-based interaction techniques for 3D pan and zoom, which are two essential tasks in HMD applications. They designed five 2D touch screen gestures mapped to pan & zoom operations in 3D. MARVIS [7] combines smartphones and HMD AR for data analysis.

Until now, few medical imaging-based surgical planning solutions on the HoloLens take full advantage of the flexibilities of the native engine.

![Fig. 2: System structure: Offline DICOM data preprocessing and anonymization (left), Direct3D based DICOM viewer prototype structure (middle), and smartphone as input device (right)](image-url)

---

**Meta Data**

- Image Stack
- DICOM Files
- Standalone Graphical C++ Engine (D3D)
- OpenXR Application Interface
- Volumetric Visualization Core Module
- 3D User Interaction
- gRPC Server
- Smartphone as Controller
3 System Design Overview

Fig. 2 shows our system’s components integrating an offline DICOM data preprocessor, a Direct3D-based DICOM Viewer, and the setup for smartphones as user input. Our previous work[16] introduces the DICOM data preprocessing tool on a desktop that categorizes information and anonymizes sensitive data. Then the graphical engine based on UWP with DirectX 11 API allows developers to manage hardware resources efficiently. The 3D renderer with OpenXR enables potential deployment on a wide range of MR devices. On the project structure side, as HL2 moved to a 64-bit Windows 10 machine that runs UWP applications, our prototype branches into two applications: one for HL2, the other running on a Windows 10 desktop. Fig. 3 illustrates the relationship between the two applications.

![Diagram](image)

**Fig. 3:** Direct3D based DICOM viewer on HoloLens 2 and desktop

4 Direct3D-based DICOM Viewer Implementation

4.1 Smartphones as User Input Devices

One of the advantages of setting up the gRPC server in a local area network is that either HL2 users or other users can more flexibly input and control the AR application. As described in previous work [16], such a server is implemented with a gRPC framework that can run in any environment. We designed the functionalities on this D3D based DICOM Viewer to match those on a smartphone, following previous work [16], which allowed all operations on the smartphone to be translated and passed to the gRPC server seamlessly.

4.2 Functionalities

**Contrast Enhancement** Previous work [16] describes a uniform enhancement of contrast and brightness.
An extension of the contrast enhancement came from the fact that our DICOM data has a resolution of 12 bits per pixel. The strategy of compressing the original 12 bits for a typical 8-bit display reduces the image contrast. Normally, we would discard the last four digits to map the data to 8 bits, or use some linear mapping, but neither solves the problem of displaying these high dynamic range images on regular displays.

Therefore, we chose to integrate Lucknavalai and Schulze’s work [8] to perform real-time 3D Contrast Limited Adaptive Histogram Equalization (CLAHE) to enhance 3D medical image stacks. Fig. 4 compares the same volume data under the same setting only applies different contrast enhancement methods. Notice that for comparison, we only change one factor for each view. However, combined adjustments can be used to improve the quality further.

![Fig. 4: Comparison of different contrast enhancement methods: (a) raw data volume, (b) only increasing the brightness, (c) only increasing the base value, and (d) using CLAHE](image)

**Color Transfer Schemes** After presenting the color transfer schemes applied in previous work, we realized that the piecewise linear paths through RGB color space have two major issues. One of them is that the perceptual flat spots in the color map hide detailed features; the other is the perceptual discontinuities that induce false features. Fig. 5 (a) and (b) show a part of the volume rendered with the HSV color scheme. The clustering of points in the green and red regions produces perceptual flat spots in the color map, thus limiting the effects only by contrast enhancement and opacity adjustment. Also, yellow and red produce false anomalies seen at these spots. Therefore, we extend the color schemes with Fire and CET-L08 from Kovesi’s work [6], which uniform the magnitude of the incremental change in perceptual lightness. As shown in Fig. 5 (c), the two new color schemes perform better for detail perception compared to the previous ones. The figures also show how at fine spatial acuity, human eyes perform much better for luminance gratings than chromatic gratings.

**Video Recording** We implemented a spectator view mode, which only renders the front RGB camera for live preview on the HoloLens 2. The initial motivation
is that when recording for a demonstration video or streaming, the frame rate on the HoloLens 2 would decrease as it does the volumetric rendering for three different views per cycle. Due to the heavy computational intensity and limited pixel fill rate, the resulting low frame rate is not acceptable for interactive usage. Therefore, our solution only renders one view from the RGB camera and doubles the frame rate compared with regular use.

HL2 provides a "Render from the PV camera" option on the Windows Device Portal. However, instead of handling all reprojection calculations and rendering the third view automatically, this option toggles a flag on the Hololens. It makes data coming from the PV camera available programmatically. We still need to enable corresponding OpenXR configs and manually handle the additional reprojected rendering for the PV camera. The data from the PV camera is self-contained enough to perform complete volumetric rendering individually without the need to do any computation from the regular display rendering. The rendered virtual hologram will then be blended with the RGB video captured by the PV camera to produce the final spectator view.

4.3 Marker-based 3D Object Placement

To achieve a marker-based 3D object placement and put multiple users under the same reference frame, we combine the marker tracking and the Hololens tracking. We integrate ArUco marker tracking [11] with OpenCV [12] on the Hololens. Due to performance concerns, we implement the OpenCV thread parallel to the main rendering thread. The OpenCV builds the reference frame and tracks the printed ArUco marker based on the images from the Left-Front VLC camera every few rendering frames. Then we create/adjust a spatial anchor onto the detected ArUco marker and use it as the frame of the reference across all devices.

OpenXR defines XiSpace to allow applications to track real-world objects. In our Hololens application, we create the following XiSpaces:

1. Reference space: defines the original pose of the device.
2. Hand pose space: defines in the hand tracking space locally for hand interaction
3. Sensor space: a space provided by Research Mode device sensor for coordinate alignments.

Most of the spaces can be directly updated by querying OpenXR methods, except for sensor space, which is from Research Mode in the WinRT context. We utilize the sensor’s rig node GUID with the Windows Perception APIs to acquire corresponding space representation in the OpenXR coordinate system.

![Diagram](image)

Fig. 6: (a) Volume positioning with the ArUco Marker in Hololens; (b) The transformation map from volume space to view space with marker tracking.

Given the model matrix of the volume $\text{ModelMatrix}$, the world to view matrix $\text{ViewMatrix}$ and the projection matrix $\text{ProjMatrix}$ provided by the OpenXR Swapchain configuration, a volume is rendered by $\text{ProjMatrix} \ast \text{ViewMatrix} \ast \text{ModelMatrix}$ when it is not positioned by a ArUco marker. In this case, mid-air hand gestures control the volume’s poses (scale, translation, and orientation). The hand movements in the HandSpace are first converted into world space, then multiplied by sensitivity value for finer control of the volume pose, and finally directly applied onto the $\text{ModelMatrix}$.

We handle the case where we use the detected ArUco marker to position the volume differently. OpenCV detects the printed ArUco markers in the sensor’s image and returns the marker’s transform $\text{Marker2Sensor}$ with respect to the LF VLC camera. The sensor space representation in OpenXR provides us up-to-date sensor transform $\text{Sensor2World}$ in the world space. With this information, we can acquire the marker transform in the Hololens’ world space by $\text{Marker2World} = \text{Sensor2World} \ast \text{Marker2Sensor}$. The volume is then rendered with the matrix transformation $\text{ProjMatrix} \ast \text{ViewMatrix} \ast \text{Marker2World} \ast \text{ModelMatrix}$ as shown in Fig. 6. In this case, we make the detected marker the reference frame’s new origin. All 3D interactions are performed locally as to this new reference frame and synchronized across different clients. By default, we render the volume on the marker when there is no 3D interaction (Fig. 6).
5 User Interactions

This section presents three types of user interaction methods under different tasks in the AR application. Fig. 7 categorizes the interaction tasks and assigns the most intuitive interaction method to them.

5.1 Virtual 2D Plane Touch

We first created an xrHand system to achieve direct touch on a virtual 2D plane instead of pinch gesture as the "select" operation. The xrHand system actively tracks the right/left-hand index tip. We draw a small sphere around the tip and check the distance between the sphere to the target plane.

When drawing on a modern digital tablet, an intuitive stroke changes the brush size adaptively according to the pressure on the touch screen. The function that transforms the pressure into brush size is called pressure sensitivity. However, as the mid-air interaction doesn’t have physical planes to provide actual pressure, we need to construct a virtual pressure mapping to gain pressure sensitivity. Our method is inspired by VirtualForce [17], which proposed calculating the sketching force based on the difference between the user’s physical hand position and their hand avatar in the Virtual Reality (VR) environment.

We transformed this concept into AR. As the index finger plunges into the virtual canvas, the hand-unaligned distance \(d\) determines the pen pressure on the canvas. The deeper the finger goes, the more pressure will be on the canvas (Equ.[1]). Then with a linear pressure sensitivity function, we map the virtual pressure to the sketch stroke size for planar data annotation (Fig. 8).
\[ d = \text{planeNormal} \cdot (P_r - \text{planePoint}), P_v = P_r - d \ast \text{planeNormal} \] (1)

Fig. 8: Virtual-force sketch for 2D data annotation: (a) screenshot of 2D sketching; (b) side-by-side illustration of the volume and annotation data.

Fig. 9: Four different free-hand 3D interactions: (a) translation; (b) rotation; (c) scaling; (d) cutting plane

5.2 3D User Interaction

We provide single-hand and two-hand gesture control for basic volume interactions and a cutting plane with 6 degrees of freedom. The user can manipulate the translation of the volume by firstly grabbing and then moving the single hand around as shown in Fig. 9a. With two hand gestures, the user can firstly hold the volume with both hands and then control the orientation and scale of the volume. Fig. 9b shows that grabbing and rotating two hands relatively will apply the same amount of rotation onto the volume. As shown in Fig. 9c, moving
two hands away from each other enlarges the volume. Conversely, moving two hands closer will shrink the volume. In the Fig. 9d we use the palm poses to control the cutting plane directions and depth.

6 Experiments

We measured the frame rate while moving the volume on five scripted paths: rotating the volume around its y-axis and its x-axis, respectively, while being either 2 m or 1 m away from the user. Each of the rotations took 10 seconds. Furthermore, we moved the volume along the z-axis, starting again at 2 m to 0 m distance. This movement path required 10 seconds to complete. All frame rates were measured by averaging over 500 ms periods.

The experiment volume contained 144 scans, each of $512 \times 512$ pixels. The measurements are on three rendering methods with parameters as shown in Fig. 10.

In Fig. 11(a), we show the frame rates for the rotation around the x-axis and y-axis at 1 m and 2 m. View-aligned rendering methods reach the highest frame rate to 46 fps rotating around the x-axis at a 2 m distance. This method performs the best overall for reducing the number of slices that need blending to get a decent view yet fewer artifacts—followed by the texture-based rendering method, which manages to exceed 30 fps rotating at a 2m distance. The raycasting method is overall costly, although it accumulates its values adaptively. It performs better for a comparably static view scenario.

Fig. 11(b) shows the frame rates for uniform rectilinear motion from 2m to 0m away. The plot shows the frame rate change across the 2m movement in 10 seconds. For the View-aligned and Texture-based methods, from the beginning 2m to 1m, the frame rate drops slowly while it drops faster from 1m to 0m. As the volume moves closer, it fills the entire two views, making the alpha-blending much slower for reading from and writing to back buffer. The change of the Raycasting method is minor because instead of alpha-blending, it works within a compute shader to sample along a ray within limited steps.
7 Conclusions

In this paper, we presented a prototypical Mixed Reality DICOM viewer based on a DirectX-based native engine. The functionalities of this prototype match a previously developed DICOM Viewer-Android system, while optimizing for two-views head-mount displays. The matched functionalities make it possible and intuitive to remote control the view inside a HoloLens with a connected smartphone. It also enables new possibilities, such as supporting collaborative work across platforms. Our testing results show that it is difficult to achieve 60 fps rendering on two views simultaneously while maintaining a high-quality rendering for a volume. However, by switching among different rendering methods while adjusting the parameters, one can find the best balance between frame rate and quality.

References

1. Büschel, W., Mitschick, A., Meyer, T., Dachselt, R.: Investigating smartphone-based pan and zoom in 3d data spaces in augmented reality. In: Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services. pp. 1–13 (2019)
2. Fröhlich, M., Bolinhas, C., Depeursinge, A., Widmer, A., Chevrey, N., Hagmann, P., Simon, C., Kokje, V.B., Gobron, S.: Holographic visualisation and interaction of fused ct, pet and mri volumetric medical imaging data using dedicated remote gpgpu ray casting. In: Simulation, Image Processing, and Ultrasound Systems for Assisted Diagnosis and Navigation, pp. 102–110. Springer (2018)
3. HOLODICOM: Diagnostic medical image in augmented reality, http://www.holodicom.com/
4. Jang, S., Stuerzlinger, W., Ambike, S., Ramani, K.: Modeling cumulative arm fatigue in mid-air interaction based on perceived exertion and kinetics of arm motion. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. pp. 3328–3339 (2017)

5. Karan, M.S., Berkman, M.I., Çatak, G.: Smartphone as a paired game input device: An application on hololens head mounted augmented reality system. In: Game+ Design Education, pp. 265–277. Springer (2021)

6. Kovesi, P.: Good colour maps: How to design them. arXiv preprint arXiv:1509.03700 (2015)

7. Langner, R., Satkowski, M., Büschel, W., Dachselt, R.: Marvis: Combining mobile devices and augmented reality for visual data analysis. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. pp. 1–17 (2021)

8. Lucknavalai, K., Schulze, J.P.: Real-time contrast enhancement for 3d medical images using histogram equalization. In: International Symposium on Visual Computing, pp. 224–235. Springer (2020)

9. Mojica, C.M.M., Navkar, N.V., Tsekos, N.V., Tsagkaris, D., Webb, A., Birbilis, T., Seimenis, I.: Holographic interface for three-dimensional visualization of mri on hololens: a prototype platform for mri guided neurosurgeries. In: 2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE). pp. 21–27. IEEE (2017)

10. Morales Mojica, C.M., Velazco-Garcia, J.D., Pappas, E.P., Birbilis, T.A., Becker, A., Leiss, E.L., Webb, A., Seimenis, I., Tsekos, N.V.: A holographic augmented reality interface for visualizing of mri data and planning of neurosurgical procedures. Journal of Digital Imaging 34(4), 1014–1025 (2021)

11. Munoz-Salinas, R.: Aruco: a minimal library for augmented reality applications based on opencv. Universidad de Córdoba (2012)

12. OpenCV: OpenCV. [https://opencv.org/](https://opencv.org/) (May 2020), https://opencv.org/

13. Ungureanu, D., Bogo, F., Galliani, S., Sama, P., Duan, X., Meekhof, C., Stühmer, J., Cashman, T.J., Tekin, B., Schönberger, J.L., Olszta, P., Pollefeys, M.: Hololens 2 research mode as a tool for computer vision research. CoRR abs/2008.11239 (2020), https://arxiv.org/abs/2008.11239

14. Zhang, M., Lucknavalai, K., Liu, W., Alipour, K., Schulze, J.P.: Arcalvr: augmented reality playground on mobile devices. In: ACM SIGGRAPH 2019 Appy Hour, pp. 1–2 (2019)

15. Zhang, M., Lucknavalai, K., Liu, W., Schulze, J.P.: Calar: A c++ engine for augmented reality applications on android mobile devices. Electronic Imaging 2020(13), 364–1 (2020)

16. Zhang, M., Schulze, J.P.: Server-aided 3d dicom viewer for mobile platforms. Electronic Imaging 2021(13), 179–1 (2021)

17. Ziyang, Z., et al.: Virtualforce: Simulating writing and sketching on a 2d-surface in virtual reality. Electronic Imaging 34, 1–5 (2022)