# Mixed Reality Surgery Assistant

A. Klaeger A. Paccagnella J. Lehner L. Albuquerque Y. Zhou

## Abstract

*Nowadays, surgeons operate complex fracture surgeries without being able to properly manipulate data during the procedures because of sterilization issues. Oftentimes, they need multiple assistants who will go through the data and tell/show them the relevant pieces of information throughout the surgery. This process is both inefficient and cumbersome as it detaches the surgeon from the actual data and wastes human resources: doctors and nurses who have to assist with data manipulation and who could otherwise be helping more objectively.*

*This application aims to assist surgeons in fracture surgeries (and potentially in other procedures) by providing an intuitive and non-contaminant method for accessing and manipulating medical data through the Microsoft Hololens 2 device[3]. Therefore, human resources can be better managed and surgeries can be performed more efficiently.*

## 1. Introduction

### 1.1. Project Goal

The goal of this project was to develop a Hololens 2 application which is capable of providing surgeons with real-time intuitive access to medical data in a non-contaminant way.

### 1.2. Methodology

This application was developed in a sequence of 5 blocks, each block happening in a period of two weeks. Every block consisted of a burst of project work, advancing and optimizing features, presentations and peer feedback on the most recent choices and accomplishments. Additionally, this project relied on the feedback of a group of surgeons based in Boston (U.S.A.) who had shown an interest in the outcome of this work.

In the first block, a Prototype phase gave rise to the first deployments of the platform. A preliminary user experience was designed, frameworks were installed and technical milestones were planned.

Secondly, the application was advanced further in order to present Interim Demos. The user experience was solidified and the most basic functions were implemented. Feedback started being more concrete both from peers and from the assisting group of surgeons.

Then, after one more round of advancements and adjustments, an Alpha Release was issued and there was a bigger assessment on whether the project was fulfilling its goals regarding the user experience.

Following a strong redesign and an intense feature development, another iteration of the system yielded the Beta release for the fourth block of the project timeline. Meetings were made on extra-official hours for testing and experimentation on non-developers and for inquiring a final feedback from the assisting surgeons' group.

Finally, with the feedback from the Beta Release and a last round of feature implementations, the final system was demonstrated live in a public presentation.

## 2. Implementation

### 2.1. Tools

The core functionality is provided by the Unity game engine[8]. Unity provides built-in Mixed Reality and Hololens support. In particular, the Mixed Reality Toolkit (MRTK)[5] for Unity provides all the basic building blocks for development of Mixed Reality projects for Hololens.

Rather than re-implementing many of the basic components of the App, this approach allowed us to rapidly prototype the basic functionality of displaying and manipulating 3D objects. Additionally, useful UI elements such as the floating menu, adjustable sliders and bounding boxes could be adapted from prefabricated assets, either from the MRTK itself or Unity's extensive Asset Store [9].

### 2.2. CT Scans

A core novel functionality of the App is the display of CT Scans [1] in the Mixed Reality environment. This also represented the biggest technical challenge of the project, as a CT consists of large amount of 3D data, which needs to be processed in real time on the limited hardware of the Hololens.

#### 2.2.1 Input

CT data is typically stored in a standard medical imaging format such as DICOM [2]. This posed the first challenge, since libraries are not commonly available for the Windows-based ARM64 architecture of the Hololens. This architecture additionally makes compiling from source difficult.

Instead, the single scan in question can be exported from the DICOM data using a tool such as MITK-GEM [4], to the much simpler NRRD (Nearly Raw Raster Data) format [6]. This provides the data for a single scan in a simplified file format with minimal headers. We then implemented a reader for this format on the Hololens.

The NRRD format still allows for many different data types, encodings and compression settings. At first, our reader allowed for a number of these settings, but this caused long loading times of the Hololens application, since these needed to be read and converted on the limited hardware of the Hololens at every application start. Instead, we decided to require the given file to already be in the required format, such that no conversion is necessary.

That format is:

- little-endian
- 4-byte float
- scaled such that values range from 0 to 1
- gzip encoded
- axis-aligned

A simple Python script is then provided to convert any NRRD file to the required format in advance using the pynrrd library [7]. The resulting file can then be added to the Unity project as a binary asset.

#### 2.2.2 Slicing

This asset is loaded into memory by a **CTReader** GameObject which exposes a method that can be used to request a render of a specific cross-sectional image (slice) of the CT. The parameters to this method consist of a vector indicating the 3D position of the point in the relative coordinate space of the CT that is to be the center of the rendered slice, as well as two directional vectors, representing the 3D orientation and scale of the slice.

As such, any arbitrary slice can be requested with any orientation - even sheared/skewed slices if desired. The **CTReader** then renders the specified slice using a high-speed compute shader running on the GPU hardware of the Hololens, using either a nearest-neighbor or a trilinear interpolation approach.

#### 2.2.3 Predefined Slicing Modes

Rather than interacting with the **CTReader** object directly, we provide two scripts that calculate the required parameters for common slicing modes.

Firstly, the **SliderSlice** script allows an axis and a slider object to be provided. The script will then request slices from the **CTReader** orthogonal to that axis, whereby the position of the slices along the axis is determined by the value of the given slider.

Secondly, the **HandSlice** script will determine the required vectors to define the slice from the current configuration of the user's hand, if it is in view. Three joints of the hand - currently the tip and knuckle of the index finger as well as the knuckle of the little finger - are used to define the slicing plane. The left or the right hand can be selected to be tracked, such that it is also possible to have two different planes rendered simultaneously - one for each hand. A small reference plane can also be rendered on the virtual model of the hand to facilitate visualization of which plane is being sliced.

## 3. Results

The final system presented to the public was a very interesting result to this project. Although much can still be done to improve it before taking it to the field, it was a very satisfactory proof of concept which showed the benefits and usability of such an application.

In its last version, the system counted with two representations of the bones being operated on: one on top, representing the broken bone with all its separate fragments; and one at the bottom, representing the "adjusted" bone, a version where every fragment of the fracture was computationally put back in place for comparison. This can be seen in Figure 1 below.

Figure 1. Broken (top) and Adjusted (bottom) bone meshes.

Even though the adjusted mesh is there solely for comparison, the original broken mesh can be rescaled, its fragments can be translated and rotated independently for better

visualization and, every time a fragment is manipulated, its corresponding piece is highlighted in the adjusted mesh.

For further functionalities, the final application relied on the 6-button menu depicted in Figure 2.


Figure 2. Menu of the application.

The menu could follow the surgeon's gaze around or it could be pinned stationary in one place.

The first button, "Hide adjustment", allows for excluding (and alternatively including) the adjusted bone mesh from the view.

The second functionality, "Edit Opacity", allows for changing the opacity of a given fragment of the bone and provides a way to have a better visualization of specific parts of the fracture without loosing track of the momentarily "unimportant" ones. The effect of this change in opacity for the biggest fragment is shown as an example in Figure 3. For going back to the original opacity settings, the button "Reset Opacity" was included in the menu.

Figure 3. Opacity change in the biggest fragment of the bone, allowing for a clearer visualization of the smaller fragments.

Because (as aforementioned) each individual fragment belonging to the broken bone mesh can be translated and rotated in space, a fourth button was included to "Reset Positions" and take the bone back to its original configuration. Regardless of the position of the pieces, though, the "Zoom" button allowed for viewing the meshes in three different scales.

Finally, the button "Hide Slider"/"Show Slider" toggles the CT Scan functionality of the system, bringing to the scene the visualization of the CT Scans of the fracture and allowing the surgeon to fluidly navigate through the data.

The Slider functionality gives the surgeon two options of accessing the data.

The first option is to select the desired CT Scan by moving sliders across two axes of the bone. The sliders represent planes cutting the bone in the cross-sections depicted by the scans, and by manually positioning them, the surgeon can visually choose the image to be presented on the middle and right screens (Figure 4).

The second option is the so-called "Free-cut", and allows the surgeon to very simply insert his or her hand in the mesh and use it as a free-cutting plane in any desired direction (Figure 5). Not only does this render great room for the surgeon's flexibility and intuition while selecting the data, but it also has the capacity of displaying brand new images not even present in the original CT Scan slices, but arbitrarily chosen by the surgeon depending on the angle of his hand. This functionality proves to be very valuable when there are non-axis-aligned planes, which are not captured by the original CT Scans, but which might be of great interest to the surgeon. With this application, the surgeon can have direct access to these planes during the procedure on the left-most screen over the bone.

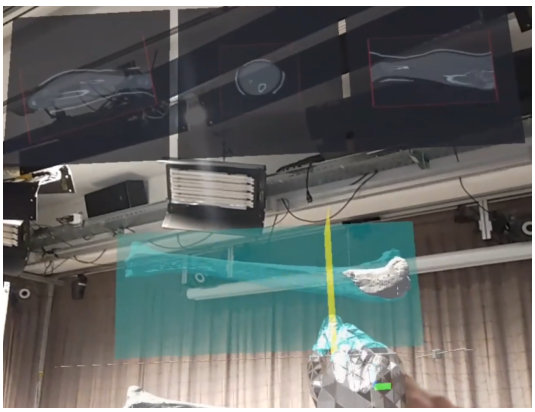

Figure 4. CT Scan visualization by moving axis-aligned sliders.

This sums up all the resulting functionalities developed for the final presentation of the system.

## 4. Discussion and conclusion

We finished a fully functional proof of concept, but certainly there are some limitations to the program in its current form, and additional features that could be developed for a real-world application.

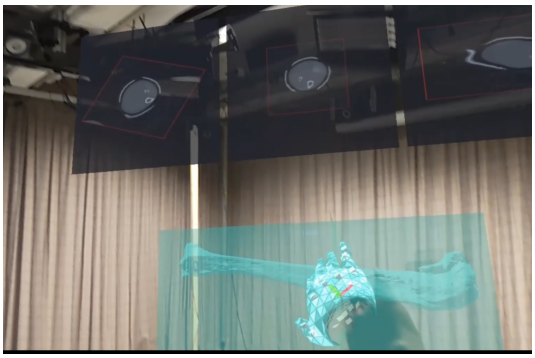

Figure 5. CT Scan visualization by ”Free-cut”.

In general we focused mostly on the surgery process itself, and less on the preparation of the operation, which could also make use of AR. One useful feature for real world use could for instance be integration with a secure patient management system to enable the doctor to easily switch between the bones of different patients. Another feature we considered was measuring the angles and distances between bone fragments. We found that it might be more useful for surgeons to use before the surgery rather than during the surgery - especially since the bone fragments inside the patient's body would inevitably be moved around frequently during the surgery process. Therefore it was not one of our prioritised features. In a finished product it might be good to divide the program into an orientation phase for the doctor to familiarise himself or herself with the 3D models, take measurements and possibly even to practice required surgery routines, and into an execution phase that allows the doctor to focus mostly on the patient in front of him or her. One issue to consider even with our smaller application is the usability: the more features there are, the harder they can be to find. We put much thought into making the menu easy to use and to limit the number of buttons; having more features might require a tutorial to familiarise the doctors with all of them.

Another feature we initially considered was to overlay the 3D models to the real bones, but feedback from the surgeons showed that they actually prefer to be able to see the bones positioned above the patient so they can see the real bones clearly during the operation. In many cases our work was based on our assumptions and to some degree on the assumptions of our cooperating surgeons. Clearly, only user tests can show which features will be most appreciated by surgeons and can help to analyse which changes need to be made to improve the application.

Our system clearly touched the boundaries of the capabilities of the Hololens 2 in its current form. Enabling all our features led the Hololens 2 to at times overheat with prolonged usage and even shut down over time. This was mainly caused by the frequent re-computations of the CT scan slices. These crashes could be avoided by making the computation more efficient or by reducing the refresh rate of the CT scan object down from the usual 60 FPS.

Finally, while the virtual bones already look very realistic now, constantly improving hardware and more processing power it will only enable the rendering of more detailed 3D models and of higher resolution CT scan slices. Better hardware will hopefully also lead to a smaller form factor of the Hololens making it more comfortable to wear for long operations.

## References


[1] Ct scan. 1
[2] Dicom. 2
[3] Hololens 2. 1
[4] Mitk. 2
[5] Mrtk. 1
[6] Nrrd. 2
[7] pynrrd. 2
[8] Unity. 1
[9] Unity assetstore. 1