Recreating Pelvic Trauma Surgery in Virtual Reality for the Development of Novel C-arm Interfaces

---

Final Report

Han Zhang
Zixuan Liu
Liam Wang

Table of Contents

1. Introduction

1.1 Clinical Motivation

Percutaneous pelvic fracture surgery is a minimally invasive technique that offers improved wound healing, reduced damage to major blood vessels and nearby nerves, and a lower risk of infection compared to traditional open surgery. However, this procedure relies on continuous or intermittent fluoroscopy, typically using a C-arm x-ray machine, which increases ionizing radiation exposure for both patients and medical staff [1]. Additionally, the absence of real-time 3D visualization and the complexity of human abdominal anatomy necessitates a thorough understanding of anatomy and surgical precision to avoid damaging the iliac artery, vein, and other vital structures [2]. Consequently, orthopedic surgeons require extra training in operating C-arm fluoroscopy and wire insertion.

To address these challenges, we propose developing a virtual reality (VR) training environment that simulates percutaneous pelvic fracture surgery under C-arm fluoroscopy. This environment would provide several benefits for orthopedic surgeons, surgical residents, patients, and the overall medical community:

1. Training without ionizing radiation exposure: The VR training environment allows clinicians to practice the procedure without exposure to ionizing radiation, making it a safer and more sustainable option for skill development.
2. Accessible data collection: The VR simulation facilitates the ethical and efficient collection of detailed operation data for further analysis, reducing the pressure on clinicians and enabling the development of computer-integrated surgical support technologies.
3. Statistical analysis: Data collected from the simulation can be used to answer questions about expert techniques, radiation dose expectations, and k-wire placement accuracy and consistency. This information can then be used to train future algorithms that can determine the phase of real surgical procedures, give advice on C-arm positioning, and provide performance feedback to surgical residents learning the procedure.
4. Testing experimental protocols or techniques: The VR environment can also be used to test new surgical techniques or protocols, fostering innovation in the field of orthopedic surgery.
5. Benefiting residents, surgeons, and patients: The VR training environment provides a low-cost, safe, and effective method for skill development, leading to better patient outcomes during percutaneous pelvic fracture surgery. Moreover, it generates expert-generated 2D educational content, POV videos, and procedure statistics from experts that can be used to improve surgical training and develop computer-integrated surgical support technologies.

In summary, we proposed a VR training environment for percutaneous pelvic fracture surgery that has the potential to significantly enhance surgical skill development and contribute to better patient outcomes by

reducing ionizing radiation exposure, enabling accessible data collection, and facilitating the development of new techniques and technologies.

1.2 **Past Work**

(Unberath et al. 2018) proposed a machine learning framework, DeepDRR, that can generate realistic Digital Reconstructed Radiographs from 3D Computed Tomography with more accurate anatomical landmark detection and localization of robot end-effectors than conventional naive DRRs. The traditional DDR is the scan that uses machine learning for material decomposition and scatter estimation in 3D and 2D. And the study demonstrates the effectiveness of DeepDRR for anatomical landmark detection in X-ray images of the pelvis.[3]

(Allen et al. 2022) used a virtual C-arm model in the VR environment to train residents with little experience observing C-arm procedures. They showed an overall significant improvement in C-Arm placement with regards to angular accuracy (mean ~2-degree improvement) and total procedure time (mean 11 minutes less time) for interventional spine procedures. We plan to improve upon the limitations of their work by adding intuitive simulated interactions with surgical tools to simulate the process of inserting orthopedic hardware, as well as adding a simulation of needle-tissue interaction. Additionally, our simulation will use DeepDRR to generate a more accurate and realistic simulated X-ray image than the naive tracing method used by this previous work [1].

1.3 **Goals**

Our project consists of three progressive goals to create a practical virtual reality (VR) training environment for percutaneous pelvic fracture surgery:

1. Baseline Product: Develop a VR environment simulating the surgery using a pre-existing CT data database. This initial phase focuses on creating the foundation for a realistic and immersive training environment.
2. Enhanced Functionality: Integrate interactive C-arm and surgical tools into the virtual environment, utilizing DeepDRR algorithms to generate simulated X-ray images. These images will provide trainees with an accurate view of pelvic anatomy, guiding them in the proper placement of K-wires and screws.
3. Advanced Simulation: Develop a sophisticated simulation of K-wire and screw insertion interactions, accounting for factors such as friction, deformation, and contact forces. This level of realism will ensure a comprehensive and authentic learning experience for trainees.

By accomplishing these goals, we anticipate delivering a cutting-edge VR training environment that accurately simulates percutaneous pelvic fracture surgery and enables the accessible collection of detailed operation data. Our solution will ultimately reduce training costs, enhance patient outcomes, and increase accessibility to C-arm fluoroscopy training, providing orthopedic surgeons with a safe and effective learning platform.

2. Technical Approach

2.1 Architecture Overview

The Unity game engine[8] was used to develop the graphics and interactions for our virtual reality training environment for percutaneous pelvic fracture surgery. Unity was chosen as the development platform because it provided the best support for easily creating complex VR interactions and user interfaces compared to SlicerVR[9] or AMBF[10]. Additionally, Unity supports deployment to a wide variety of consumer virtual-reality headsets including both computer-tethered headsets like the HTC Vive[13] and standalone headsets like the Meta Quest 2[14]. To generate DRR images we used the existing Python library DeepDRR[11] due to its superior speed and realism compared to competing techniques. The ZMQ networking library[12] was used to communicate between the modules of the simulation system. Each component of the system connects has a ZMQ publish and subscribe socket which connects to the central ZMQ publish/subscribe proxy server which re-routes packets to appropriate modules. The networking API works similarly to the publish/subscribe system of the Robot Operating System (ROS) by enabling arbitrary nodes to publish and subscribe to topics. The ZMQ publish/subscribe proxy server is analogous to the ROS core and contains no application-specific code; it simply re-routes published messages to any clients subscribed to particular topics. Packets of data between the modules are encoded using the Cap'n Proto serialization library[15] which can generate Python and C# code for serialization/deserialization of data structures defined in a schema language. This networking system enables flexible real-time communication between the components of the simulation system. For instance, the Unity application can communicate with the DeepDRR server to request real-time DRR generation. The patient mesh server module allows the runtime loading of new patients into the simulation. The recording server module enables continuous data collection of all simulation parameters such as the poses of users, the C-arm, and K-wires, as well as all DRR images collected by users. The ZMQ network connection is also used to synchronize the simulation environment between Unity client instances, enabling multiple users wearing HMDs to be present together in the virtual operating room allowing for interactive mentoring and collaborative training. All server-side modules and the ZMQ proxy application are packaged inside of a Docker[16] container for ease of setup and to enable the option to deploy the server to the cloud. The source code and getting started guide for our server-side code can be found at (https://github.com/PelvisVR/deepdrr_zmq) and the API documentation for our server-side code is located (https://pelvisvr.github.io/deepdrr_zmq/deepdrrzmq.html).

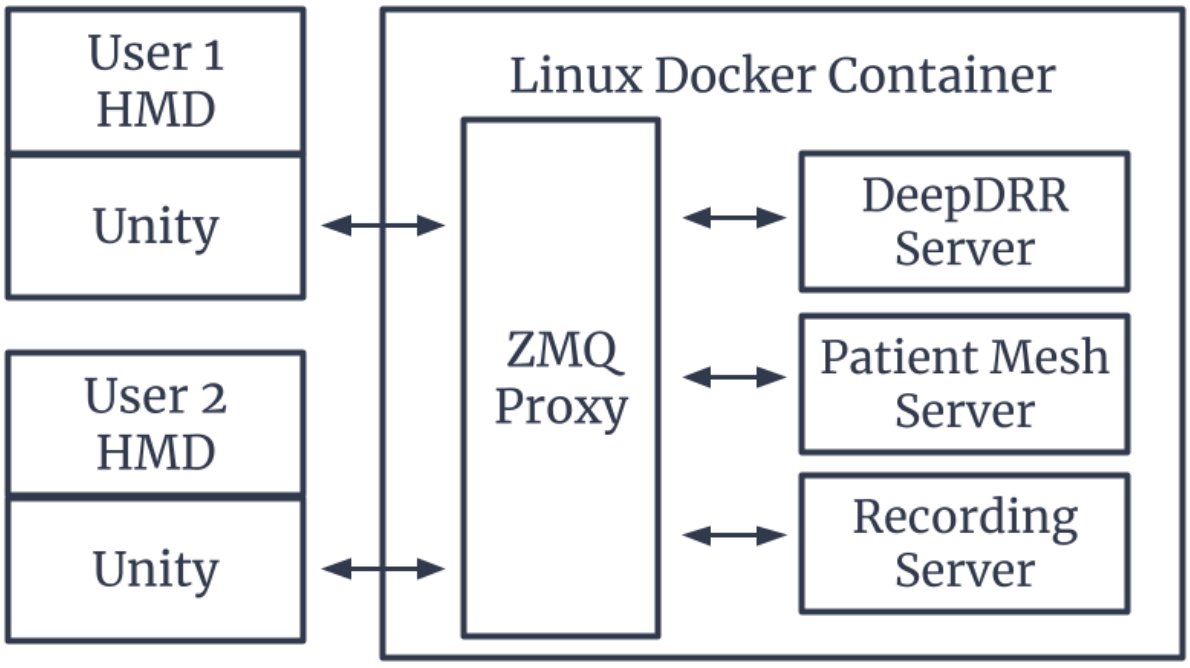

Figure 1. Technical Block Diagram Overview

2.2 DeepDRR Network Connection

We used DeepDRR to provide surgeons with realistic simulated X-ray images from the virtual C-arm, which can be placed at an arbitrary position and orientation relative to the virtual patient. DeepDRR is a Python package for Linux computers with an NVIDIA CUDA-capable GPU with >11 GB of memory. To generate the radiographs, DeepDRR requires a 3D CT volume and information about the relative location/intrinsic camera properties of the fluoroscopy device. Additionally, the locations and density volumes of any surgical tools which need to be included in the simulated radiographs must be provided. Given this information, the DeepDRR program can output a simulated radiograph.

When the user requests a radiograph in the Unity application, the Unity app sends a data structure to the DeepDRR server with information about the position and orientation of the C-arm, patient, and surgical tools, as well as C-arm camera intrinsics. The DeepDRR server processes the request and replies with a DRR image which the Unity application will decode and display on a virtual monitor next to the patient.

2.3 Multiplayer Connection

Multiple users wearing virtual reality headsets can train collaboratively in the virtual surgical room using the multiplayer system we implemented. Users can see each other as virtual avatars in the operating room and see the movement of the C-arm and surgical tools when moved by other users. Additionally, any DRR snapshots taken by any user will be displayed for all users.

Our multiplayer system uses the same ZMQ connection as the other networked modules. When a surgical tool is moved locally by a player, the Unity client publishes a stream of timestamped pose updates to the ZMQ proxy. Each Unity client constantly listens for packets containing timestamped pose updates for objects in the surgical simulation. When a packet is received, the client updates the position of the corresponding object in the local scene. To smooth out position updates, the Unity client attempts to show the position of each object at a small fixed time offset in the past, (e.g. 200 milliseconds) and uses linear interpolation or extrapolation to blend between the last received updates. This algorithm is based on the method described in [17].

Virtual avatars are also displayed using a similar method. Each Unity client creates a unique identifier on startup, which it publishes at a fixed rate to the ZMQ proxy as a heartbeat. Additionally, a stream of poses for the hands and HMD position is published to the ZMQ proxy. Unity clients listen for heartbeats from other Unity clients and instantiate local avatar models for every other client on the network, which are driven using the pose streams from the other clients.

2.4 3D Surgical Instrument Design

In order to provide a realistic simulation of pelvic fracture surgery, the surgical instrument design consists of two main components: an interactable C-arm fluoroscopy model, and pelvic fractures internal fixation tool model. All models are open source and are able to provide immersive interaction.

2.4.1 C-arm fluoroscopy model

The models will be designed to enable trainees to move the C-arm into the proper position to obtain the desired X-ray image of the pelvic region. The model includes the main body and a controller to take the shot. In order to enable all DoFs of the C-arm, we use GE Healthcare OEC One[18] taken from CGmodel[19]. The model was first modified by the 3DS Max to align the axis for each moving component and remove the unnecessary parts for better rendering before importing it into Unity.
In total, we enable all 7-DoF for the c-arm, and limits for each moving component were also assigned to provide a realistic simulation. The source position was calculated by the pose of each moving component using homogenous transformation. The DRR camera matrix will be attached at the source position, generating real-time DRRs based on the dynamic pose of this virtual C-arm model during the simulation.

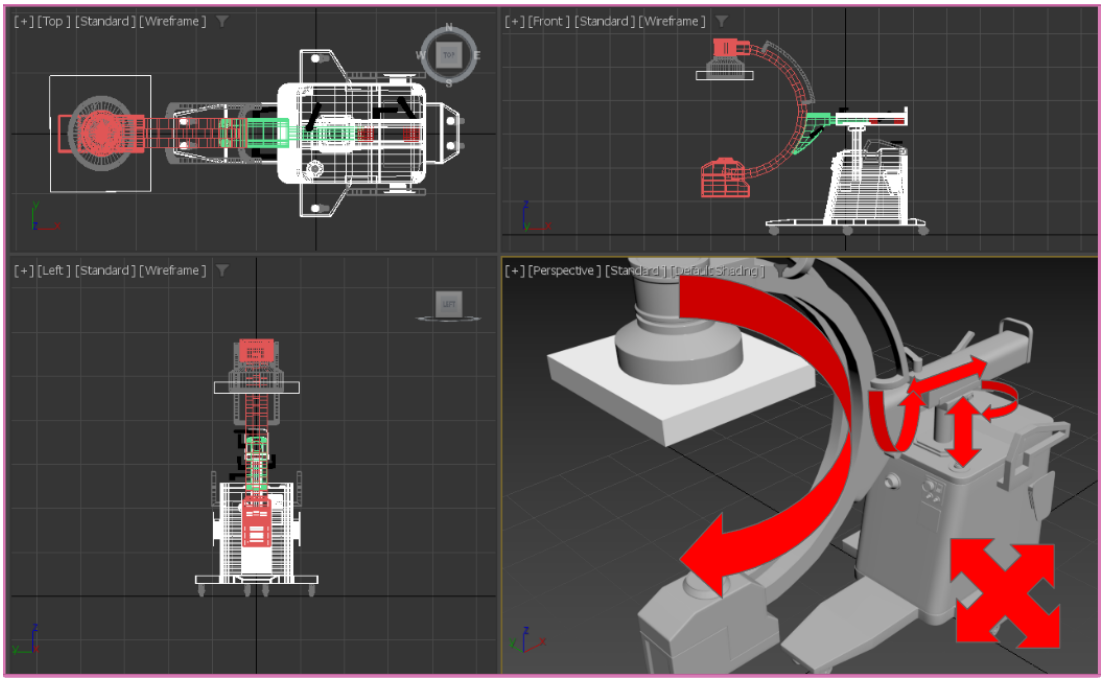

Figure 2. The C-arm components(gantry, cranial/caudal support, wag, and wag support), and all 7-DOF required to calculate the camera matrix.

2.4.2 Pelvic fractures internal fixation tool models

The pelvic fractures internal fixation tool models consist of K-wire sets, surgical screw sets, surgical drills, and surgical driver kits. The K-wire sets come in two lengths (300mm and 450mm) and a diameter of 2.8mm, which is commonly used for percutaneous fixation of pelvic fractures. The surgical screw sets come in a diameter of 6.5mm, thread lengths of 12mm and 32mm, and their length ranging from 30mm to 130mm. These screw sets are used for providing internal fixation after inserting the k-wire. The k-wire and screws have been modeled with proper thread patterns. The surgical drill and driver kits are designed to be used and fit with both the K-wire sets and surgical screw sets precisely. The surgical drill model is designed by 3DSMAX and the rest are designed by Solidworks with high accuracy and real scale to provide an immersive experience for trainees.

In addition to the design, the models have been optimized for performance to enable real-time interaction during the simulation. The models have been created using low-poly meshes and textures, leading to less computation recourse for rendering in real-time and animations.

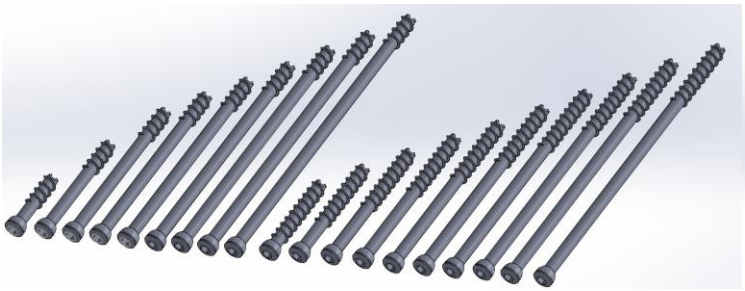

Figure 3. The screw set(left, 6.5mm Diameter, 12 & 32 mm thread length, 30 - 130 mm length), surgical driver set(right), and kwire set(bottom, 300 & 450 mm length, 2.8 mm diameter) in Solidworks

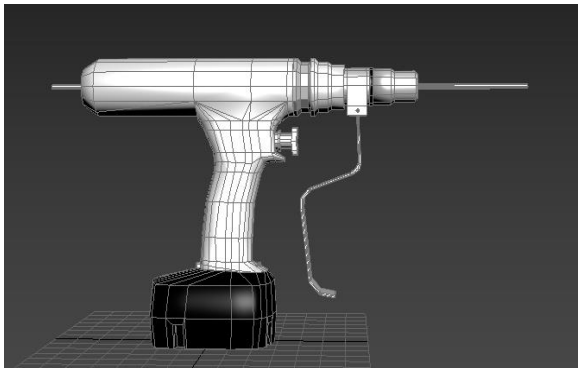

Figure 4. the drill with k-wire in 3DS MAX

2.5 Tool-Tissue Interaction

Tool-tissue interaction is a critical aspect of surgical simulation during the pelvic fracture fixation procedure, as it enables trainees to practice their k-wire or screw insertion skills and experience tissue feedback. When it comes to bone tissue, the interaction is designed to disable rotation and only enable backward movement when inserting the k-wire with bare hands. However, when inserting the k-wire with the drill and trigger on, both forward and backward movement is enabled, as this closely resembles the movement of surgical instruments used in bone surgeries. In addition, high vibration is enabled to emulate the resistance encountered when drilling into bone tissue, or when users tend to insert the K-wire into the bone with their hands.

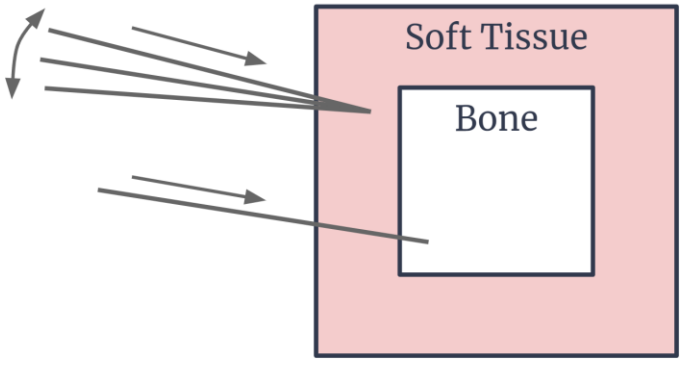

Figure 5. Insertion Simulation

On the other hand, in soft tissues such as muscles and organs, the interaction is designed to enable pivot rotation and forward/backward movement, with a little bit of dampness to mimic the tissue resistance when moving. Furthermore, low vibration is enabled to simulate the softer resistance encountered when moving into the soft tissue.

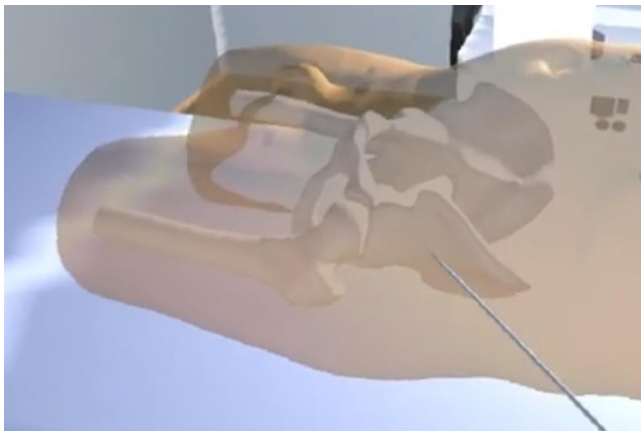

Figure 6. Unity Scene, k-wire can only move forward(with drill only)/backward when in the bone while enabling the pivot rotation when in tissue.

The following is the algorithm for our tool and tissue interaction:

```
if tipInbone \\in bone
        Move = handpos - lasthandpos
        Projection = Project(move, tipYaxis)
        if(Dot(Projection,tipYaxis) > 0 and nodrill)  // if disable the forward movement if not use drill
                Projection  = 0
        newKwirepos  = lastKwirepos + projection
        newKwirerot = lastKwirerot
elseif tipIntissue \\in tissue
        Move = handpos - lasthandpos
        Projection = Project(move, tipYaxis)
    newPosition = lastKwirepos + Projection;
    rotationX = Vector3.Dot(Vector3.Project(move, tipZAxis), tipZAxis)  //pivot rotation on z-axis
    rotationZ = Vector3.Dot(Vector3.Project(move, tipXAxis), tipXAxis)  //pivot rotation on x-axis
    newKwirerot = lastKwirerot* rotationZ * rotationX;
    newKwirepos  = lastKwirepos + projection
lastKwirerot = newKwirerot
newKwirepos = newKwirepos
```

Overall, the tool-tissue interaction in the surgical simulation is designed to closely mimic the real-life interaction between surgical tools and tissues. Our team, PelvisVR, will soon provide more sophisticated tool-tissue interaction, which will be Raycasted engine-based, enabling an even more realistic and immersive simulation experience.

2.6 **User Interaction**

The user will interact with our virtual reality environment using a VR Head-mounted Display (HMD) and VR hand controllers. The HMD will allow the user to view the virtual reality surgical room in 3D, and the hand controllers will act as a proxy for the user's hands and be used to manipulate the C-arm and any interaction with tools and scenes. Here is the overview of our unity scene and the controller manual:

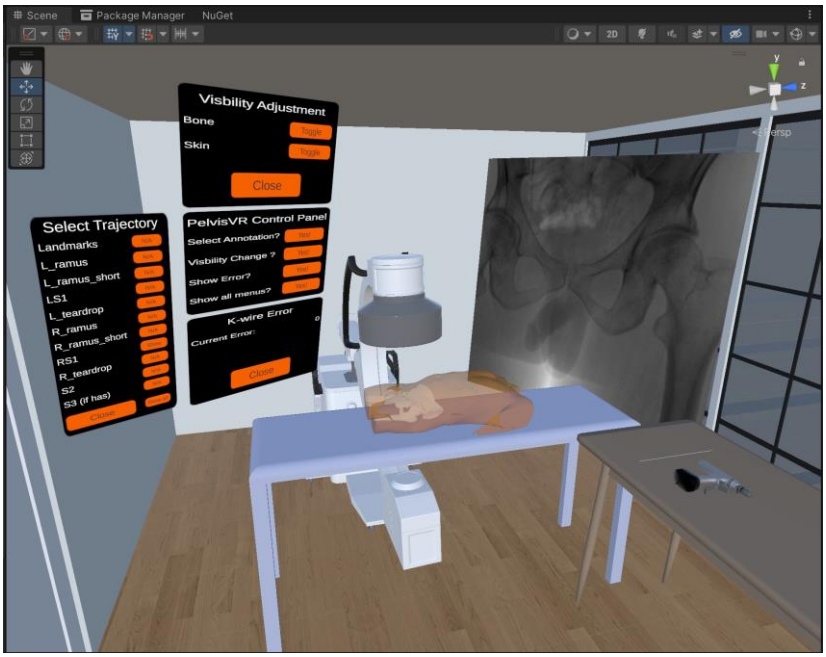

Figure 7. The Overview of the Virtual Reality Surgical Room

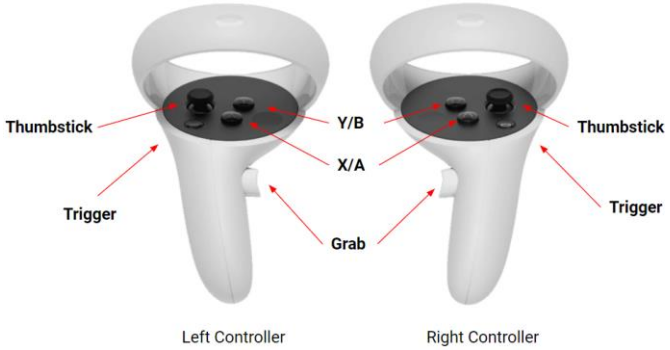

Figure 8. The Quest 2 Joystick Controller

In our virtual reality surgical room scene, we have multiple objects, including an interactable C-arm, interactable surgical tools, a patient model, a DRR display, and a canvas for the control menu. We mainly utilize XR Toolkit[20] to provide the desired interaction with objects. The XR Grab Interactable enables users to interact with tools by grabbing them with hand controllers, such as grabbing the k-wire, screws, drill, etc. Users are able to get notified by brief vibration when their controllers are near any objects and can press the “Grab” button to grab them.

For the interaction with the C-arm, multiple grabbable regions (colliders) were assigned based on the desired movement, such as swing arm distal/proximal, rotate arm over/back, tilt arm distal/proximal, push base in/out, slide base distal/proximal. Users can use their left controller thumbstick to control the C-arm, raising the arm up/lowering the arm down (y-axis), pushing the arm in/pulling the arm out (x-axis). The figure shows the typical movement language of the C-arm.

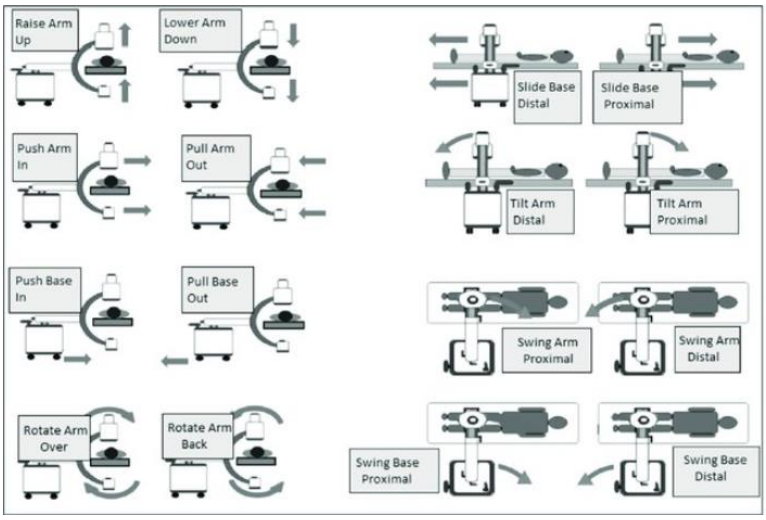

Figure 9. Standard Universal C-arm Language[21]

https://www.researchgate.net/publication/329804862_A_standard_universal_C-arm_language_Assessing_its_need_and_its_likelihood_of_acceptance

For user locomotion, we enable both head tracking and teleport options. Users can either physically move their body or activate the ray by pressing the "X/A" button to transport them to the target position by pressing the "grab" button, achieved by the XR Ray Interactor script from XR Toolkit. Moreover, the interaction with the canvas is achieved by the activated ray and pressing the "trigger" buttons from the controller.

Combined with realistic models, XR Toolkit, and providing multi-dimensional feedback, the intuitive and responsive VR interactions will allow users to easily manipulate the C-arm and surgical tools, providing a comprehensive training environment that can improve patient outcomes and reduce training costs.

2.7 **Annotation**

2.7.1 Annotation and Patient Loading

The patient model utilized in this project has been annotated based on CT scans obtained from actual patients' pelvis. Although pelvic annotation can be complex and extensive, our primary focus is on the following annotations: landmarks, ramus, short ramus, SI, and the teardrop from left to right, S2, and S3. Depending on the patient, it is possible that S3 may not be present. In such cases, our focus will only concentrate on the rest of the annotations. Below figure 1 shows an example case of the annotated patient. Each annotation comprises a line trajectory and two endpoint fiducials. The landmarks consist of four fiducials, two of which are affixed to the left and right ASIS, while the other two are affixed to the left and right Pelvis.

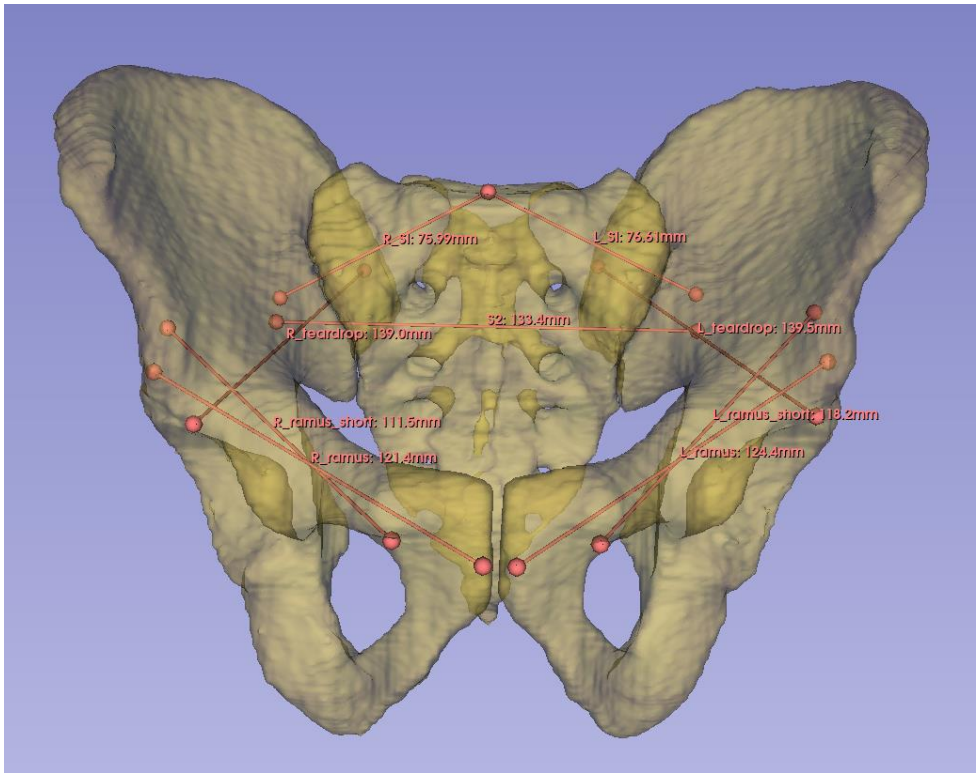

Figure 10. Pelvic annotation in 3D Slicer

This project comprises a total of 40 annotated cases. All annotations and the pelvis model are stored on the server and will be loaded in real time when the user opens the application with a specific case input.

2.7.2 Error Calculation

To ensure accurate alignment of the K-wire with the annotation during insertion, we have implemented error calculations and color changes of the K-wire as an assistance mechanism for the user. When the user moves the K-wire close to the annotation corresponding to the current insertion operation, the K-wire turns orange. Subsequently, when the user aligns the K-wire with the annotation to form a straight line (i.e., if the user only moves the K-wire back and forth, they will necessarily insert it at the specified location), two situations can arise. The first is when the user inserts the K-wire from the entry point of the annotation; in this case, the K-wire turns green. However, if the user inserts the K-wire from the exit point of the annotation (i.e. if they insert it the wrong way), the K-wire turns blue.

2.7.2.1 The Translation Error and The Rotation Error

To achieve the color change, we use two types of errors: the translation error and the rotation error. The rotation error is simply the difference in angle between the two lines, i.e., the angle difference between the annotation and the K-wire. As for the translation error, it is calculated as follows.

For each annotation, we have two endpoint coordinates, a_1 and a_2, which correspond to the points where the annotation intersects with the pelvis. Similarly, we have two endpoint coordinates for the K-wire, k_1 and k_2, which correspond to the endpoints of the K-wire.

To determine the alignment between the K-wire and the annotation, we first calculate two direction vectors:

$e_1 = a_2 - a_1$ (the direction vector of the annotation) $e_2 = k_2 - k_1$ (the direction vector of the K-wire)

We then calculate the normal vector n, which is perpendicular to both direction vectors, by taking the cross product of e1 and e2:

$$n = e_1 \times e_2$$

If the magnitude of n is zero, this indicates that the two lines are parallel and do not intersect. In this case, the shortest distance between the two lines is 0, and we can use this value as the translation error.

If the two lines are not parallel, we proceed to calculate the shortest distance between them. This is achieved by first calculating the difference vector between the endpoint coordinates of the two lines:

$$d = k_1 - k_2$$

We then take the dot product of d and the normal vector n:

$$dot = d \cdot n$$

Finally, we divide the absolute value of dot by the magnitude of n to obtain the translation error:

$$translation_{error} = \frac{|dot|}{||n||}$$

The translation error represents the distance between the K-wire and the annotation along the normal vector direction.

By using both the translation and rotation errors, we can determine the alignment between the K-wire and the annotation. This information is then used to change the color of the K-wire to indicate whether it is aligned with the annotation or not, thereby assisting the user in accurately inserting the K-wire.

2.7.2.2 Line Segment Error and Tip Translation Error

The line segment error is defined as the shortest distance between two line segments, which is the minimum Euclidean distance between any point on one line segment and any point on the other line segment. The tip translation error means the shortest distance between the tip of the K-wire to the annotation, which is essentially the shortest distance between a point and a line segment. As mentioned previously, the K-wire changes color

when the user moves it toward the annotation. The threshold for color change is determined by the line segment error. When this distance is smaller than the threshold value, the K-wire changes color to indicate that it is approaching the correct alignment with the annotation. If the distance between the two lines is greater than or equal to the threshold value, the K-wire retains its original color. The threshold value is chosen based on the desired level of precision and accuracy for the insertion procedure. The tip translation error may provide additional assistance to users in determining whether the K-wire has been correctly inserted into the exit point of the annotation.

To calculate the line segment error, we employ a two-step approach to determine the shortest distance between the given line segments. The first step involves checking whether the line segments are parallel. If they are parallel, the distance between one line segment and a point on the other line segment is calculated. If the line segments are not parallel, the function proceeds to identify the closest points on both line segments and computes the distance between these points. To check whether the two line segments are parallel, the process is the same as before. If they are parallel, the distance between them can be calculated by finding the closest point on one line segment to a point on the other line segment:

Suppose we have two endpoints of a line segment, a and b, and one end point on another line segment p, then the closest point coordinate on the lines segment to the point on another line segment is calculated by:

$$closestpoint = a + (b - a) \cdot Clamp01(\frac{(p - a) \cdot (b - a)}{\sqrt{|(b - a)|}})$$

The shortest distance between two line segments can be calculated using the Euclidean distance between their respective endpoints, and this is also the method used to calculate the tip translation error.

In cases where two line segments are not parallel, the nearest points on each segment are initially computed by projecting the segments onto one another, which entails identifying the points on each segment in closest proximity to the opposite segment. Subsequently, the Euclidean distance between these nearest points is calculated to ascertain the minimal distance between the two line segments. This method resembles the approach employed for parallel segments, albeit with the supplementary step of projecting the segments onto each other to pinpoint the closest points.

2.8 **UI Interface**

Our system's user interface has undergone several modifications and optimizations, and it now consists of three main parts. The first part is a large screen located at the center of the operating room that primarily displays a user guide, informing users about the functionalities of the various buttons on the VR controller. The second part is a large screen situated in front of the operating table, which is primarily used to display real-time digital

radiography (DRR) images. If the DRR images have not yet been fully loaded, this screen will appear red. The third part (shown in Figure 2) is a control menu located on the left side of the operating table, which is primarily used to manage the interactive features related to surgical operations. While the first and second parts are relatively straightforward, this section will primarily focus on the third main part.

The Control Menu

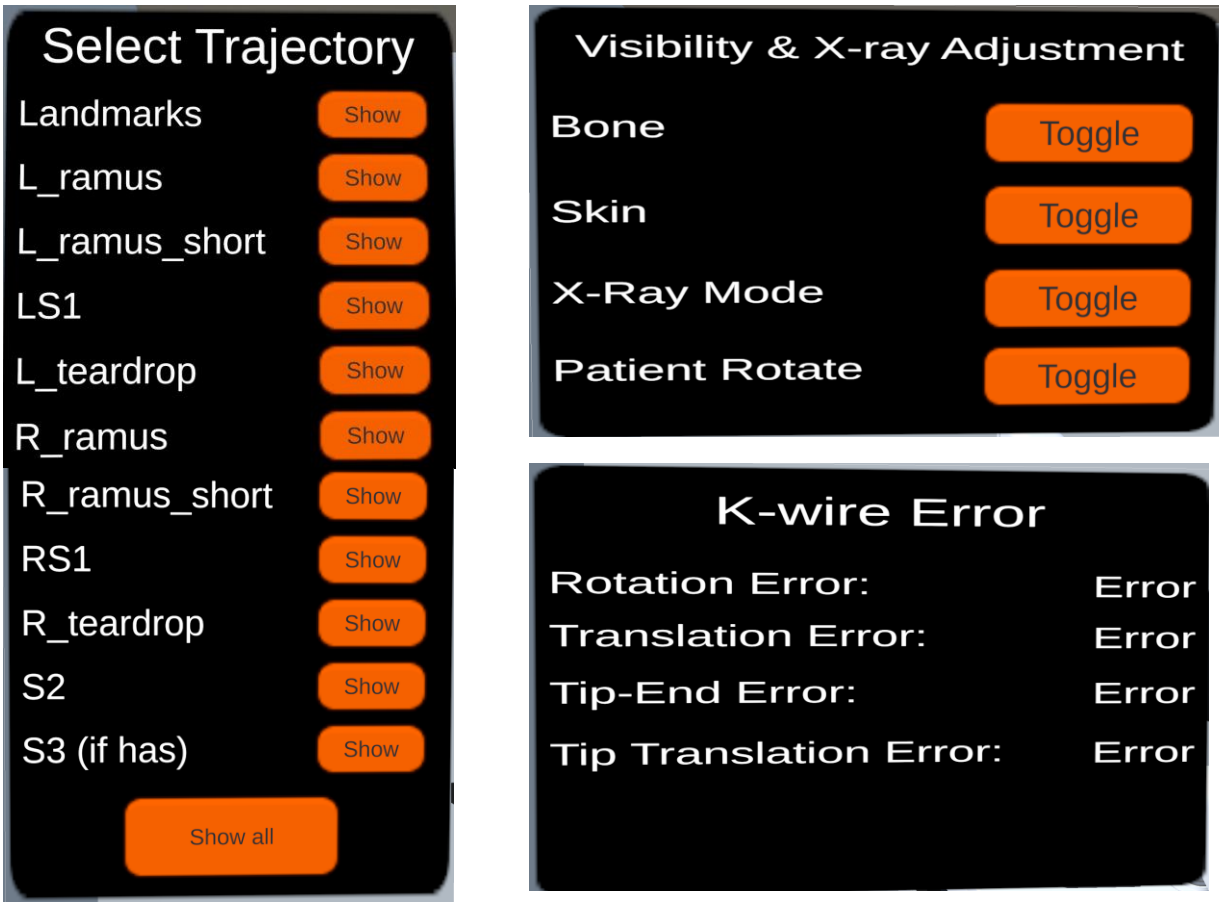

Figure 11. Control Menu UI

2.8.1 Functional Aspect

In this section, the functional performance will be explained in detail. The control menu can be divided into three main parts. The first part is labeled "Select Trajectory," which is primarily used to control the display and disappearance of annotations. The second part, named "Visibility & X-ray Adjustment," is primarily used to manipulate the opacity, rotation, and real-time DRR image display mode of the patient model. The third part, titled "K-wire Error," is primarily used to display real-time K-wire errors when the user begins the process of inserting a K-wire.

Firstly, since loading annotations into the system takes a certain amount of time, typically within a few seconds, this control menu is primarily intended for manipulating annotations. To protect the annotation loading integrity, the menu will only appear after all annotations have been fully loaded. This also serves as a signal to the user that the annotations have finished loading and they may proceed with their operations.

Within the Select Trajectory menu, users are limited to selecting and showing only one annotation at a time. This design choice is in line with actual surgical scenarios where K-wires are inserted one at a time. When a user selects to show an annotation, the corresponding button will turn green, and the text on the button will change to "hide". At this point, the user cannot select to show any other annotations until the current annotation is hidden. Once the user has hidden the current annotation, they are then able to select another one.

However, when the user selects “show all” to show all annotations, they will be displayed simultaneously. In this scenario, the K-wire will not have the color-changing property, and the “K-wire Error” interface will not display any real-time errors of the K-wire with respect to the annotation.

The green color and "hide" text on the button provide a clear indication to the user that the annotation is currently being displayed. This design choice allows users to keep track of which annotations are currently visible and which are hidden. Moreover, it allows users to focus their attention on the current annotation without any distractions from other annotations.

The Visibility & X-ray Adjustment page provides users with straightforward methods for manipulating various aspects of the patient model. Users can adjust the opacity of both the Bone and Skin layers, select between continuous and single-shot X-ray modes, and choose whether or not to flip the patient model. The flipping option is especially useful for users who need to train on specific trajectory positions, as it provides them with a more optimal viewing angle.

Overall, the Visibility & X-ray Adjustment page is designed to provide users with the tools they need to visualize the patient model in a way that optimizes their workflow. The straightforward methods available on this page allow users to make adjustments quickly and easily, which ultimately leads to a more efficient and productive experience.

The K-wire Error menu will exhibit the error values of K-wire concerning annotations in real time. All calculations for these errors have been meticulously elucidated in the preceding section. It is noteworthy that the real-time display of errors is synchronized with the color-changing attribute of the K-wire. In other words, when the K-wire approaches the annotation sufficiently and begins to turn orange, the error information will be displayed, as it now provides a reference value. When the K-wire has not yet been moved to the vicinity of the current annotation, all default error values are set to zero and will be displayed on the respective menu.

2.8.2 Non-Functional Aspect

The application's user interface adheres scrupulously to the eight golden rules of interface design[22]. For example, the UI endeavors to maintain consistency. All buttons exhibit a uniform, appealing color, and their style remains consistent throughout. Moreover, the font style across various interfaces are also congruent with one another, particularly among those possessing analogous functionality. Third, all buttons provide real-time, multi-directional feedback. For instance, when a user aims the VR controller's ray at a button they wish to select, the button will subtly turn green, indicating that it has been successfully targeted. Upon the user's actual click with the VR controller, the button synchronously turns solid green. The reverse situation holds true as well. Within the UI of this system, every button is configured in the manner described above, offering users ample feedback and enabling them to identify any errors they might have made when failing to successfully click a button.

The application's user interface strictly adheres to the eight golden rules of design, emphasizing consistency in button style and font, as well as offering real-time, multi-directional feedback. The UI ensures an intuitive and user-friendly experience, allowing users to easily identify errors and successfully interact with the system.

2.9 **Data Logging**

Our virtual reality surgical simulation environment must be able to collect all data related to user interactions to be an effective training and data collection tool. To achieve this, we leverage the fact that all relevant information for data collection is already sent through the ZMQ Proxy server. This includes all DRR snapshot requests from the Unity application and the response DRR images generated by DeepDRR. Additionally, all pose information for each user's hand controllers and HMD, as well as the positions of all surgical tools and the C-arm are already sent through the ZMQ Proxy server for multi-user synchronization. Therefore, the recording server simply logs all network packets routed through the ZMQ proxy server. The log files are saved as a sequence of Cap'n Proto-serialized messages containing the timestamp, topic name, and data of each message. The log files are automatically sharded into a sequence of compressed file chunks and saved to the server hard drive and can be easily loaded and parsed for future analysis.

3. Progress Evaluation

3.1 **Dependencies**

The figure presented herein delineates the dependencies for our project. All requisite dependencies have been procured in accordance with initial expectations. Furthermore, we have submitted an Institutional Review Board (IRB) application, with the anticipation of approval prior to May 30th. This will facilitate the execution of the user study, which is slated for commencement during the summer months.

Dependency	Need	Followup	Expected Date	Hard Deadline	Contingency Plan	Status
Pelvis CT Dataset	Input to DeepDRR	Perform annotations	02/03	02/23	N/A	✅
3D Slicer	Annotate landmarks	Perform annotations	02/03	02/23	Python Package	✅
Computing Power	Running DeepDRR and VR application	Install DeepDRR and Unity	02/13	02/23	`pong` workstation	✅
DeepDRR	Generate realistic X-ray images	Develop DeepDRR ZMQ server	02/15	03/03	Python Package	✅
HTC Vive Pro	Display VR simulation	Connect to PC	02/13	03/03	N/A	✅
Unity	Develop VR environment	Create Scene and Asset	02/13	03/03	Use Slicer VR	✅
C-Arm 3D Model	Simulate the C-arm in VR	Rig with kinematics	02/20	03/03	Open-source Asset	✅
IRB Approval	Approval to conduct the user study	Run the user study	05/30	06/15	N/A	✅

Figure 12. Dependencies

3.2 **Project Deliverables & Key Steps**

Relative to the checkpoint status, we have revised the project milestones. All initial minimum, expected, and maximum deliverables have been successfully accomplished. At present, our focus has shifted to achieving objectives that surpass the original maximum expectations: the user study, the implementation of multiplayer functionality, and the collection of data from the user study. We anticipate the completion of these additional deliverables over the course of the summer months.

	Key Milestones/Activities	Results/Deliverables	Deadline	Status
Minimum	Completion of the CT data annotation	Annotated CT data that includes landmarks and K-wire paths	Feb 3	Done
	Literature & background review	Project proposal document	Feb 15	Done
	Set up network connection between Unity and DeepDRR ZMQ server	A documented and unit-tested python DeepDRR server package and Unity project that can display live X-ray images from a CT model corresponding to different angles of the camera.	Feb 17	Done
	Model 3D C-arm and surgical tool models	Static Unity prefabs containing C-arm and tool models.	Feb 20	Done
	Rig C-arm kinematics and calculate C-arm and K-wire to DeepDRR frame transformations	A Unity prefab that can be imported and is capable of reporting the live K-wire position and orientation in the simulated X-ray images, along with corresponding documentation and unit tests.	Feb 28	Done
Expected	Vive Controller C-Arm movement controls	Documented MonoBehaviour script components that allow the user to adjust the C-Arm model in VR. A set of manual test procedures for ensuring working VR C-Arm movement interactions.	Mar 20	Done
	Vive Controller interface for picking up different surgical tools (k-wire, screw, drill)	Documented MonoBehaviour script components that allow the user to operate the tools smoothly and realistically. A set of manual test procedures for ensuring working VR interactions. Documentation for VR interaction interfaces.	Mar 20	Done
	Write a guide for extending our VR simulation for future enhancements	A documented guide to extending our simulation environment for future C-arm devices, use-cases, and techniques.	Mar 23	Done
Maximum	Implement a basic K-wire-skin and K-wire-bone interaction simulation	Documented MonoBehaviour script components in the project that can restrict the movement of the K-wire insertion in different stages. A series of test cases to verify the behavior functions as defined .	Apr 15	Done
	Implement Vive controller haptics feedback for K-Wire insertion simulation	Documented MonoBehaviour script components which provide VR controller vibration haptics to help the user recognize and adjust their movements during the K-wire insertion process.	Apr 15	Done
	Implement a mouse and keyboard interface for patient selection and task configuration	GUI for the user headset. Documentation and instructions for using the GUI.	May 1	Done
	Multiplayer VR environment	A Unity-based environment that the mentor can provide real-time guidance to user on the surgical simulations.	Summer	Incomplete
	Conduct a User Study for the usability of VR surgery	A paper analyzing the effectiveness of virtual pelvic surgery training	Summer	Incomplete
	Annotate C-arm pose, multi camera view and X-ray data from user study	A dataset of surgical training procedures	Summer	Incomplete

Figure 13. Milestones

3.3 Management Summary

The team consists of three members and two mentors, and the responsibilities of each person are listed here:

1. Han Zhang, Biomedical Engineering Master's Student: Design of surgical tool models(CAD), Unity Interaction, Tool & Tissue Interaction, and User Study.
2. Zixuan Liu, Computer Science Master's Student: Annotation, Patient Loading, UI Interface, and Documentation.
3. Liam Wang, Biomedical Engineering & Computer Science Undergraduate Junior: DeepDRR Network & Multiplayer Connection, and Data Logging.
4. Benjamin Killeen, Ph.D. Student in Computer Science(Team Mentor).
5. Mathias Unberath, Assistant Professor in Computer Science (Team Mentor).

The team held weekly meetings that included a student team meeting for brainstorming and a mentor meeting for progress reports. Communication took place through various platforms, including GitHub for code, Zoom meetings, Discord text, email, and messaging for team communication, and Google Drive for write-ups. The team also used Jira and GanttProject for task timelines and administrative tasks. Those management methods ensure everyone is on the same page and progress can be tracked effectively throughout the project.

4. Conclusion

4.1 Result & Verification

To ensure our system is stable, we've used different strategies to verify that our system fully meet the software requirements in the software development specification. The testing part mainly consists of three components: functional requirement test, object orientation test, and the walkthrough method.

4.1.1 Functional Requirement Testing

To guarantee that our system possesses the characteristics outlined in the original design, we have conducted a comprehensive evaluation of the functional requirements for the VR training environment application created for percutaneous pelvic trauma surgery. This evaluation has involved rigorous testing and verification procedures, and we are pleased to report that the application has been found to satisfy 90% of the designated functional requirements.

1. Patient Model Selection
   Verified that users can choose an arbitrary patient model from the models that stored in the server with different pelvic fracture types and anatomical variations, providing diverse training experiences.
2. C-arm Interaction

Confirmed that users can manipulate the C-arm model in real-time using VR hand controllers, successfully positioning and orienting the C-arm during the surgical procedure as the same as the real C-arm.

3. Surgical Tool Interaction
   Tested user interactions with various surgical tools, such as K-wires, screws, and drills, using VR hand controllers. The simulation of K-wire insertion and tool interaction was found to be accurate and realistic.
4. Realtime X-ray Simulation
   Evaluated the DeepDRR deployment and verified the generation of realistic, real-time simulated X-ray images based on the position and orientation of the C-arm, patient model, and surgical tools. The DeepDRR can generate high quality X-ray images in the system in different modes.
5. Haptic and Audio Feedback
   Experienced haptic and audio feedback during C-arm and surgical tool interactions, which effectively enhanced the immersive experience for the user.
6. Error feedback
   Confirmed that the system automatically collects and analyzes the K-wire and annotation transform, providing the error information for users to assist the K-wire insertion procedure.
7. Cross-platform Compatibility
   Tested the system's compatibility with various VR headsets and controllers, ensuring accessibility and adaptability for different hardware configurations.

After careful consideration, certain functional requirements have not been implemented at present. The User Authentication and Scalability and Customization features were deemed unnecessary for the current stage of development. Additionally, the Performance Metrics and Evaluation and Tutorial and Guidance System functionalities could not be implemented until the completion of the user study. Although there is some basic guidance already incorporated into the system, further development of the Tutorial and Guidance System must wait until after the user study is complete.

4.1.2 Object Orientation Test & Walkthrough

As our system was developed in Unity and C# script using the object-oriented design, we conducted thorough testing of the object orientation to ensure that each class correctly encapsulated the relevant information, and that all operations were free from any data leakage. To validate the constraints, contracts, and method specifications, we utilized the role-play method, which involved simulating different scenarios based on use cases [23]. These use cases were evaluated from various perspectives, such as from the server's standpoint for data networking, or from the user's perspective for K-wire interaction and patient model evaluation. During the UI design phase, we utilized the walkthrough method to evaluate the system's user interface. Given that we were unable to involve an actual user at this stage, we presented the UI prototype to developers who were not involved in the interface design process. Together, we thoroughly examined all aspects of the user interface, discussing each element's design and intended purpose. The UI developer explained how the interface was designed for specific reasons, and then made

incremental and iterative improvements based on our feedback. Through this comprehensive approach to testing, we were able to verify the system's compliance with all relevant specifications and requirements.

4.2 **Discussion**

Overall, our project aimed to develop a virtual reality training environment for percutaneous pelvic fracture surgery that simulates the procedure under C-arm fluoroscopy. We sought to create a platform that reduces ionizing radiation exposure, enhances surgical skill development, and collects detailed operation data for further AI training. By integrating DeepDRR algorithms, multiplayer networking, interactive C-arm machines, and a sophisticated simulation of user interaction, we have achieved our primary goal.

We consider ourselves fortunate to serve as a bridge between the high-tech and medical fields. Through the development and implementation of our VR training environment, we have learned several key lessons.
First, we learned how to manage our separate tasks as both of us worked on the same unity project, the ability to work in parallel. We created a git repository and synced that folder to our computers. After each pulling, pushing, and merging process, we have reduced the possibility of overwriting and conflict problems, which is one of the most annoying problems we met during the project. Second, we learned a lot about how good documentation is for improving the software development process, which will provide seamless and clear integration of various components, such as network connections, user interfaces, and tool-tissue interactions. Finally, effective project management and communication among team members and our mentor, Benjamin, ensured the timely completion of project milestones and the successful development of our project.

In short, our team has successfully eliminated some limitations of traditional training in percutaneous pelvic surgery by leveraging the power of virtual reality and advanced simulation techniques. In addition to verifying our virtual reality training environment in the user study, we have identified several directions and potentials for our project. First, although our current simulation focuses on percutaneous pelvic fracture surgery, it is possible to adapt for other fluoroscopy-guided orthopedic procedures, such as spinal surgeries and knee joint replacements, benefiting various intraoperative fluoroscopy surgical training as a comprehensive platform. Second, due to the limitation of VR hardware, the haptic feedback from tool-tissue interaction is only provided by controller vibration. Further integrating advanced kinesthetic haptic feedback could enhance the realism of tool-tissue interactions. For example, when k-wire inserts into the bone, user movement could be limited by applying an external force constraint mechanism, such as a 3D-printed hardware mechanism. Lastly, machine learning algorithms for surgical performance assessment could be developed, enhancing the overall learning experience for trainees and integrating surgical training curricula.

4.3 **Next Steps**

In the upcoming summer, we plan to conduct a user study to evaluate the effectiveness of our virtual reality training environment. This study will provide participants with a unique opportunity to improve their imaging acquisition skills and surgical tool manipulation in our simulated environment.

The user study will involve participants wearing a head-mounted display(Meta Quest 2) and utilizing handheld controllers to interact with the virtual simulation. A tutorial session will be provided to acquaint participants with the VR hardware. Throughout the simulation training, a co-investigator will be present as a virtual avatar, serving as a mentor and offering relevant feedback to guide the procedures.

The evaluation sessions, conducted before and after the training, will record participants' surgical performance and imaging acquisition skills, including the total number of X-rays taken, k-wire insertion errors, and their understanding of effective image acquisition. Additionally, all object transform matrices with timestamps will be recorded. Our team aims to contribute valuable knowledge to the field of medical education and training, with the potential to enhance patient outcomes and overall healthcare quality. The target conference to show our system and user study findings is IPCAI 2024.

5. References

[1]Daniel R. Allen, Collin Clarke, Terry M. Peters & Elvis C.S Chen (2022) Development and evaluation of an open-source virtual reality C-Arm simulator, Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, DOI: 10.1080/21681163.2022.2152374

[2]Rami Mosheiff, Chip Routt.Percutaneous fixation of pelvic fractures. orthoinfo - aaos. OrthoInfo. (n.d.). Retrieved February 8, 2023, from https://orthoinfo.aaos.org/en/treatment/internal-fixation-for-fractures

[3]Unberath, M., Zaech, J.-N., Lee, S. C., Bier, B., Fotouhi, J., Armand, M., & Navab, N. (2018). DeepDRR – A Catalyst for Machine Learning in Fluoroscopy-guided Procedures. arXiv. https://doi.org/10.48550/ARXIV.1803.08606

[4]XR Interaction ToolKit: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/xr-grab-interactable.html

[5]Rommens, P. M., Graafen, M., Arand, C., Mehling, I., Hofmann, A., & Wagner, D. (2020). Minimal-invasive stabilization of anterior pelvic ring fractures with retrograde transpubic screws. Injury, 51(2), 340-346.

[6]Wang, Z.-h. and Li, K.-n. (2019), Regional Injury Classification and Treatment of Open Pelvic Fractures. Orthop Surg, 11: 1064-1071. https://doi.org/10.1111/os.12554

[7]Würfl, T., Hoffmann, M., Christlein, V., Breininger, K., Huang, Y., Unberath, M., & Maier, A. K. (2018).

Deep learning computed tomography: Learning projection-domain weights from image domain in limited angle problems. IEEE transactions on medical imaging, 37(6), 1454-1463.

[8]https://unity.com/

[9]Pinter C, Lasso A, Choueib S, Asselin M, Fillion-Robin JC, Vimort JB, Martin K, Jolley MA, Fichtinger G. SlicerVR for Medical Intervention Training and Planning in Immersive Virtual Reality. IEEE Trans Med Robot Bionics. 2020 May;2(2):108-117. doi: 10.1109/tmrb.2020.2983199. Epub 2020 Mar 26. PMID: 33748693; PMCID: PMC7977740.

[10]Varier, Vignesh Manoj, et al. "AMBF-RL: A real-time simulation-based Reinforcement Learning toolkit for Medical Robotics." 2022 International Symposium on Medical Robotics (ISMR). IEEE, 2022.

[11]https://github.com/arcadelab/deepdrr

[12]https://zeromq.org/

[13]https://www.vive.com/us/product/vive-pro2-full-kit/overview/

[14]https://www.meta.com/quest/products/quest-2/?utm_source=gg&utm_medium=ps&utm_campaign=18854243668&utm_term=meta%20quest%202&utm_content=656722224892&utm_funnel=dcap&gclid=Cj0KCQjwu-KiBhCsARIsAPztUF3--EEVP9ylUET7S6wYK7o1Tq-NNDVraIC08XOFm3WDuXKok4dpRSQaAjMcEALw_wcB&gclsrc=aw.ds

[15]https://capnproto.org/

[16]https://www.docker.com/

[17]https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking#:~:text=4%20Dead%202-,Entity%20interpolation,-By%20default%2C%20the

[18]https://www.gehealthcare.com/products/surgical-imaging/oec--one

[19]https://www.cgmodel.com/model/347633.html

[20]https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/xr-grab-interactable.html

[21]https://www.researchgate.net/publication/329804862_A_standard_universal_C-arm_language_Assessing_its_need_and_its_likelihood_of_acceptance

[22]Shneiderman, Ben, et al. *Designing the user interface: strategies for effective human-computer interaction*. Pearson, 2016

[23] Dennis A, et al. *SYSTEMS ANALYSIS & DESIGN An Object-Oriented Approach with UML.* WILEY, 2015