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

Project Proposal

Han Zhang
Zixuan Liu
Liam Wang

Table of Contents

1. Introduction

1.1 Clinical Motivation

Percutaneous pelvic fracture surgery is a minimally invasive surgical technique that provides better wound healing, less damage to major vessels or nearby nerves, and a lower risk of infection compared to traditional open surgery. However, the use of continuous or intermittent fluoroscopy, typically with a C-arm x-ray machine, increases ionizing radiation exposure to both patients and clinical staff [1]. Moreover, the lack of real-time 3-dimensional visualization feedback and the complexity of the human abdomen requires surgeons to have a comprehensive understanding of anatomy and surgical accuracy to avoid injuring the iliac artery, vein, and other visceral structures [2]. This necessitates extra training for orthopedic surgeons on the operation of C-arm fluoroscopy and the insertion of wires. Thus, there is a need for clinicians to have a training environment for percutaneous pelvic fracture surgery under C-arm fluoroscopy that enables practice of the procedure without exposure to ionizing radiation and allows for easy collection of detailed operation data for further analysis.

To address this need, we will create a virtual reality environment with patient models and an interactable C-arm and surgical tools for recreating internal fixation of pelvic fractures. We will use DeepDRR to provide fast and realistic Digitally Reconstructed Radiographs (DRR) from 3D Computed Tomography (CT) data. Surgeons will view the simulation through a Virtual Reality (VR) Head-Mounted Display (HMD) and interact with the C-arm and surgical tools using VR hand controllers.

1.2 Literature Review

(Unberath at el. 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 who had few 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 will intend to improve upon the limitations of their work by adding intuitive simulated interactions with surgical tools to actually 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 more accurate and realistic simulated x-ray image than the naive tracing method used by this previous work [1].

- DEEPDRR
- Allen et al

1.3 Goals

Our project has three tiers of goals. The first goal is to provide a baseline product of a virtual reality (VR) environment that simulates the percutaneous pelvic fracture surgery using an already-obtained database of CT data. Our second goal is to achieve a target level of functionality by incorporating an interactable C-arm and surgical tool in the virtual environment. The simulated X-ray images generated by the DeepDRR algorithms will be used to provide trainees with a realistic view of the pelvic anatomy and guide the insertion of K-wires and screws. Additionally, we aim to develop a realistic simulation of K-wire and screw-body insertion interactions that account for factors such as friction, deformation, and contact forces.

By achieving these goals, we expect to create a state-of-the-art training environment that accurately simulates percutaneous pelvic fracture surgery and allows for easy collection of detailed operation data. This will ultimately reduce training costs, improve patient outcomes, and increase accessibility to C-arm fluoroscopy by providing a safe and effective training environment for orthopedic surgeons.

2. Technical Approach

2.1 Overview

We will use the Unity (https://unity.com/) game engine to develop our virtual reality environment to simulate the surgical procedure. We plan to distinguish our virtual surgical environment from previous work by focusing on more robust and realistic VR interactivity using the hand controllers. For this reason, Unity was chosen for our development platform rather than SlicerVR or AMBF, as we determined that Unity has the best support for easily creating complex VR interactions and user interfaces.

Since we are using Unity, our virtual environment will be compatible with any SteamVR compatible VR headset. For development, we will use an HTC Vive Pro headset and controllers.

2.2 DeepDRR Network Data Connection

We will use 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. 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.

DeepDRR is a Python package for Linux computers with an NVIDIA CUDA-capable GPU with >11 GB of memory. To connect the DeepDRR program to our virtual reality simulated environment Unity program, we will implement a socket-based DeepDRR server to allow the Unity program to request radiographs from DeepDRR in real-time. The DeepDRR server will have the flexibility to run on the same computer as the Unity VR application, or on a separate computer connected via Ethernet or WiFi to the computer running the Unity application. The Unity VR application will run on a computer running Windows, so the DeepDRR server will have the capability to run inside a Windows Subsystem for Linux virtual machine or Linux Docker Container on the same computer.

We will use the ZeroMQ networking library (https://zeromq.org/) for socket communication over TCP due to its robust pub-sub messaging patterns that will allow the server to be easily extended to serve multiple clients in the future. We will define the network packet serialization data structures using the Cap'n Proto data interchange format (https://capnproto.org/) which can automatically generate both C# (Unity) and Python code to serialize/deserialize data structures from binary buffers.

When the user requests a radiograph in the Unity application, the Unity app will send 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 will process the request and reply with a DRR image which the Unity application will decode and display on a virtual monitor next to the patient.

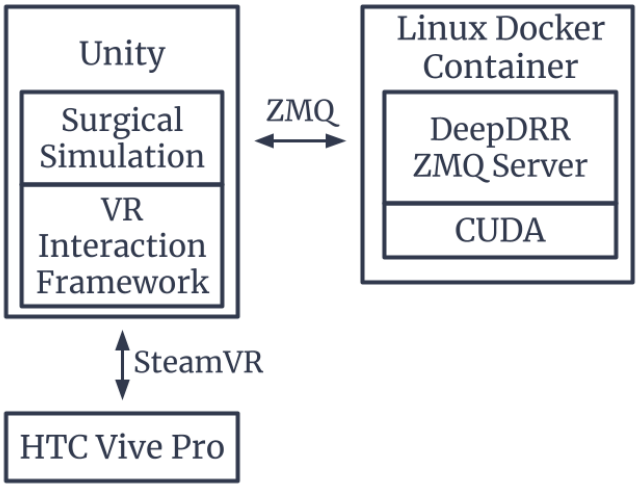

Figure 1. Technical Block Diagram Overview

2.3 Surgical Simulation

C-arm Model and Kinematics

To generate a simulated X-ray from DeepDRR and to track the rotation and translation of K-wires, we use homogeneous frame transformations to calculate the kinematics between the Unity room and DeepDRR world, as shown in the accompanying image. This process enables us to project the simulated X-ray images accurately and place the virtual K-wires in the correct location, orientation, and depth within the pelvic

anatomy. By incorporating this technique, we can provide trainees with a more realistic and immersive training environment that accurately simulates the surgical procedure.

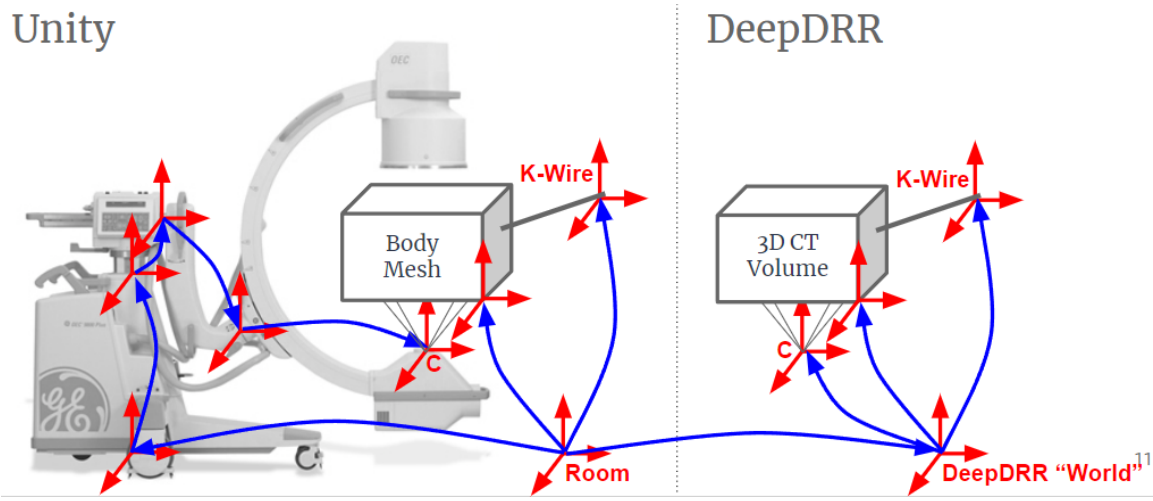

Figure 2. Unity to DeepDRR Transform Chain

K-wire simulation

We will implement a basic Kirschner-wire (k-wire) interaction simulation with different types of tissue to make the wire insertion process more intuitive for surgeons who are familiar with the real procedure. At a minimum, the simulation must be sufficiently realistic such that surgeons feel that their performance in the virtual environment is at least somewhat representative of their skills in real procedures. After discussion with our advisors, we believe that while the experience of actually inserting k-wires will be quite different in our simulation compared to the real world (e.g. no force feedback/less accurate tissue interaction), the skills of adjusting the C-arm and spatial placement of k-wires based off of the x-ray images should translate well to our simulation. As a maximum deliverable, we plan to investigate more sophisticated k-wire/tissue interaction simulation methods if time allows.

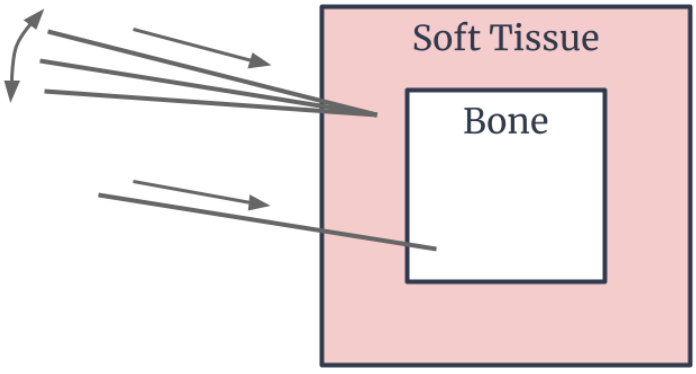

Figure 3. Basic K-wire Simulation Interaction

2.4 VR Interaction Framework

For the user and virtual environment interaction, we will main involve two parts: C-arm and surgical tool interactions. Since the user will manually manipulate the multiple parts C-arm to the desired position, we will provide haptic via joystick controller and audio feedback with unity build-in packages, XR Interaction ToolKit[4]: The user will feel increments of rotation when they tilt, orbit, and wig-wag the C-arm by vibration and get the sound feedback with the movement of the C-arm.

For the surgical tool interactions, we will use the XR Grab Interactable in XR Interaction ToolKit, it enable users to interact with the tools by grabbing them with hand controllers. It also contains different models of grabbing like direct, attach, and toggle, which provide the developer different ways to define the interaction. As this project is applied to the surgical cases, the constraints applied to the tools may become complex, which lead the user may need to control the tool simultaneously by using different button or joystick on the controller. To resolve this, the XR Grab Interactable supports different input types such as triggers, grips, and buttons on hand controllers.

2.5 Interface Definitions

To enable parallel development of the various software components of our project and to enable unit testing, we have defined interface specifications between each component. The major components of our project include the Unity Surgical Simulation which will communicate with the DeepDRR server and will receive user input via our VR Interaction framework.

Unity Surgical Simulation to DeepDRR Server Interface

The interface between the unity surgical simulation and the DeepDRR server consists of Cap’n Proto-defined messages which wrap the native DeepDRR interface to allow DRR requests to be sent over the network.

The following are the key Cap'n Proto struct definitions for this interface:

```
struct CameraIntrinsics {
   sensorHeight @0 :UInt32 = 1536; # Height of the sensor in pixels
   sensorWidth @1 :UInt32 = 1536; # Width of the sensor in pixels
   pixelSize @2 :Float32 = 0.194; # Size of a pixel in mm
   sourceToDetectorDistance @3 :Float32 = 1020; # Distance from the source to the detector in mm
}

struct CameraProjection {
   intrinsic @0 :CameraIntrinsics; # Camera intrinsics
   extrinsic @1 :Matrix4x4; # Camera extrinsics
}

struct Device {
   camera @0 :CameraProjection; # Camera projection of the device
}

struct NiftiLoaderParams {
   path @0 :Text; # Path to the nifti file on the server
   worldFromAnatomical @1 :Matrix4x4; # Transformation from the world coordinate system to the
anatomical coordinate system
   useThresholding @2 :Bool = true; # Segment the materials using thresholding (faster but less
accurate)
   useCached @3 :Bool = true; # Use a cached segmentation if available.
   saveCache @4 :Bool = false; # Save the segmentation to a cache file.
   cacheDir @5 :Optional(Text) = (none = void); # Directory to save the cache file to.
   segmentation @6 :Bool = false; # If the file is a segmentation file, then its "materials"
correspond to a high density material.
}

struct DicomLoaderParams {
   path @0 :Text; # Path to the dicom file on the server
   worldFromAnatomical @1 :Matrix4x4; # Transformation from the world coordinate system to the
anatomical coordinate system
   useThresholding @2 :Bool = true; # Segment the materials using thresholding (faster but less
accurate)
   useCached @3 :Bool = true; # Use a cached segmentation if available.
   saveCache @4 :Bool = false; # Save the segmentation to a cache file.
   cacheDir @5 :Optional(Text) = (none = void); # Directory to save the cache file to.
   segmentation @6 :Bool = false; # If the file is a segmentation file, then its "materials"
correspond to a high density material.
}

struct VolumeLoaderParams {
   union {
       nifti @0 :NiftiLoaderParams;
       dicom @1 :DicomLoaderParams;
   }
}

struct ProjectorParams {
   volumes @0 :List(VolumeLoaderParams); # List of volumes to project
   priorities @1 :List(UInt32); # List of priorities for each volume
   device @2 :Device; # Device to project from
   step @3 :Float32 = 0.1; # Size of the step along projection ray in voxels. Defaults to 0.1.
   mode @4 :Text = "linear"; # Interpolation mode for the kernel. Defaults to "linear".
   spectrum @5 :Text = "90KV_AL40"; # Spectrum array or name of spectrum to use for projection.
Options are `'60KV_AL35'`, `'90KV_AL40'`, and `'120KV_AL43'`.
   scatterNum @6 :UInt32 = 0; # the number of photons to sue in the scatter simulation.  If zero,
scatter is not simulated.
   addNoise @7 :Bool = false; # Whether to add Poisson noise.
   photonCount @8 :UInt32 = 10000; # the average number of photons that hit each pixel. (The
expected number of photons that hit each pixel is not uniform over each pixel because the detector
is a flat panel.)
   threads @9 :UInt32 = 8; # Number of threads to use. Defaults to 8.
   maxBlockIndex @10 :UInt32 = 1024; # Maximum GPU block. Defaults to 1024.
   collectedEnergy @11 :Bool = false; # Whether to return data of "intensity" (energy deposited per
photon, [keV]) or "collected energy" (energy deposited on pixel, [keV / mm^2]).
```

```
   neglog @12 :Bool = true; # whether to apply negative log transform to intensity images. If True,
outputs are in range [0, 1]. Recommended for easy viewing.
   intensityUpperBound @13 :OptionalFloat32 = (none = void); # Maximum intensity, clipped before
neglog, after noise and scatter. A good value is 40 keV / photon.
   attenuateOutsideVolume @14 :Bool = false; # Whether to attenuate photons outside the volume.
}

struct StatusResponse {
   code @0 :UInt16; # Status code
   message @1 :Text; # Status message
}

struct ProjectRequest {
   requestId @0 :Text; # Unique request id
   projectorId @1 :Text; # Unique projector id
   cameraProjections @2 :List(CameraProjection); # List of camera projections to project from
   volumesWorldFromAnatomical @3 :List(Matrix4x4); # List of transformations from the world
coordinate system to the anatomical coordinate system
}

struct ProjectResponse {
   requestId @0 :Text; # Unique request id
   projectorId @1 :Text; # Unique projector id
   status @2 :StatusResponse; # Status of the request
   images @3 :List(Image); # List of images
}

struct ProjectorParamsResponse {
   projectorId @0 :Text; # Unique projector id
   projectorParams @1 :ProjectorParams; # Projector parameters
}

struct ProjectorParamsRequest {
   projectorId @0 :Text; # Unique projector id
}
```

The Cap'n Proto protocol will be used as follows:
All communication shall occur over ZMQ pub/sub sockets.

When the unity surgical simulation requires a DRR from the DeepDRR server, it will send a ProjectRequest struct on the "project_request/" topic, which contains a unique projector ID to be used for the request, as well as position data for the imaging device and all volumes.

If the projector ID has not been initialized yet on the server, the server will send a ProjectorParamsRequest on the "projector_params_request/" topic to the unity surgical simulation, which will reply with a ProjectorParamsResponse containing initialization information for the projector and any volumes contained within. The server will then initialize the projector.

Finally, the server will reply with a ProjectResponse struct on the "project_response/" topic, which contains the DRR image(s) requested by the client.

VR Interaction to Unity Surgical Simulation Interface

In this project, VR interaction comprises three main components: the VR controller, surgical tools, and the C-arm. To ensure seamless integration of these components, it is essential to define their respective interfaces, given the numerous interactions and parameter transitions involved. The VR controller acts as a proxy for the user's hands within the VR environment, enabling interaction with the virtual world. The surgical tools encompass various instruments that a surgeon would utilize during pelvic trauma surgery. The

C-arm facilitates X-ray imaging during the operation, with its movement and degree of freedom limited by realistic kinematics. The subsequent section delineates the primary interfaces of each component, elucidating the mode of communication among the three constituents.

- VR Controller
    - setTrackingType(Type): The time during which the controller samples tracking input within the frame. Input: a certain tracking type.
    - setPositionAction(boolean, reference): To locate the position of object. Input: a boolean to judge whether using the reference; a reference of the input device.
    - setRotationAction(boolean, reference): To set the rotation of object. Input: a boolean to judge whether using the reference; a reference of the input device.
    - setTrackingStateAction(boolean, reference): To get he tracking state of object. Input: a boolean to judge whether using the reference; a reference of the input device.
    - setSelectAction(boolean, reference): To select a game object while the input type is a boolean value. Input: a boolean to judge whether using the reference; a reference of the input device.
    - setSelectActionValue(boolean, reference): To select a game object while the inut value is a float number. Input: a boolean to judge whether using the reference; a reference of the input device.
    - setActivateAction(boolean, reference): To enable a reference to a game object while the input type is a boolean value. Input: a boolean to judge whether using the reference; a reference of the input device.
    - setActivateActionValue(boolean, reference): To enable a reference to a game object while the input type is a float number. Input: a boolean to judge whether using the reference; a reference of the input device.
    - setHapticDeviceAction(boolean, reference): To set the haptic feedback from a game object to device. Input: a boolean to judge whether using the reference; a reference of the input device.
    - setLineType(Line): To set the line type of the teleportation line. Input: a certain type of the line.
    - setEndPointDistance(float): Set the end point distance of the teleportation line. Input: a float number.
    - setEndPointHeight(float): Set the highest height of the teleportation line. Input: a float number.
    - setControlPointDistance(float): Set the control point distance of the teleportation line. Input: a float number.
    - setControlPointHeight(float): Set the control point height of the teleportation line. Input: a float number.
- Surgical Tool
    - onToolGrabbed(Controller): Grab the tool. Input: a certain input from the controller.
    - onToolReleased(Controller):Release the tool. Input: a certain input from the controller.
    - setInteractionManager(): Manage the Interactors and Interactables.
    - setMovementType(): Set different movement type of the tool.
    - attachTransform(Game Object): Set the coordinate that binds with the tool to be moved
    - setInteractableEvents(): Set the interactale events of the tool.

    - setMass(float): Set the mass of the tool.
    - setKinematics(): Set whether the tool follows the kinematics.
- C-arm
    - setMovement(boolean): Enable the movement of the C-arm. Input: a boolean value.
    - rotateCC(angle): Set the transition of the C-arm related to the world coordinate. Input: an angle value.
    - moveWage(transition): Set the transition of Wage related to the C-arm. Input: an angle value
    - moveTable(transition): Set the transition of Table related to the C-arm. Input: an angle value
    - rotateGantry(angle): Set the rotation of Gantry related to the C-arm. Input: an angle value
    - setMonitor(boolean): Enable the monitor of the C-arm. Input: a boolean value.

2.6 Testing Plan

Our testing plan consists of three major categories.

Code Validation Deliverables: We must ensure that our code is robust and stable to ensure a quality user experience and to ensure our simulation is easily extensible as a platform for other data collection and experiments.

- A set of unit tests to verify correctness of mathematical calculations.
- A set of networking tests to ensure Unity-DeepDRR network communication can recover from unstable network conditions and invalid data.
- A set of manual test procedures to ensure the VR user interface works correctly.

Testing of our C-Arm Model Deliverables: We need to ensure that our C-Arm kinematics mirror actual C-Arm kinematics so that surgeons will feel familiar with the motion characteristics of the C-Arm. To keep the scope of our project reasonable, we will begin by implementing the kinematics and visuals for one C-Arm brand, the Siemens Cios, but our simulation will be easily extendible to other C-Arm models.

- A document showing the results of our comparison of the Unity C-arm kinematics with DeepDRR C-arm kinematics.
- A document evaluating the accuracy of the C-Arm 3D model by comparing its movements and positions to actual C-Arm devices.

Simulation Verification Deliverables: We need to investigate how accurately our simulation mimics real pelvic trauma surgery.

- (Maximum Deliverable): A paper describing the results of a user study involving experienced surgeons performing simulated procedures and providing qualitative feedback. A set of quantitative procedure metrics and compare to real pelvic fracture surgery outcomes. We will have succeeded if expert surgeons believe their use of the simulation is a fair enough representation of their skills. Additionally, statistics such as the approximate number of X-ray images taken should match analogous real procedures.

3. System Development

3.1 Project Deliverables & Key Steps

Based on our corresponding minimum, expected, and maximum goals, we divided them into several specific milestones.The key milestones and deliverables are shown below.

	Key Milestones/Activities	Deliverables
Minimum	Completion of the CT data annotation	Annotated CT data that includes landmarks and K-wire paths
	Literature & background review	Project proposal document
	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.
	Model 3D C-arm and surgical tool models	Static Unity prefabs containing C-arm and tool models.
	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.
Expected	Vive Controller C-Arm movement controls	Documented MonoBehavior 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.
	Vive Controller interface for picking up different surgical tools (k-wire, screw, drill)	Documented MonoBehavior 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.
	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.
Maximum	Implement a basic K-wire-skin and K-wire-bone interaction simulation	Documented MonoBehavior 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 .
	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.
	Implement a mouse and keyboard interface for patient selection and task configuration	GUI for the user headset. Documentation and instructions for using the GUI.
	Conduct a user study with expert surgeons to analyze the usability and accuracy of our simulation	A paper containing the results of our user study and a comparison of our system to prior similar systems.

Table 1. Milestones & Deliverables

3.2 Dependencies

We will build our system in unity and display in a head-mounted display., which need a few physical dependencies. The dependencies are listed below. We have already got some software installed and hardwares such as the computation recourse and head-mounted display.

Dependency	Need	Status	Followup	Expected Date	Hard Deadline	Contingency Plan
Pelvis CT Dataset	Input to DeepDRR and segment for ground truth	Acquired	Perform annotations	02/03	02/23	N/A
3D Slicer	Annotate landmarks	Done	Perform annotations	02/03	02/23	Python Package
Computing Power	Running DeepDRR and VR application	Acquired	Install DeepDRR and Unity	02/13	02/23	`pong` workstation
DeepDRR	Generate realistic X-ray images	Done	Develop DeepDRR ZMQ server	02/15	03/03	Python Package
HTC Vive Pro	Display VR simulation	Acquired	Connect to PC	02/13	03/03	N/A

Unity	Develop VR environment	Acquired	Create Scene and Asset	02/13	03/03	Use Slicer VR
C-Arm 3D Model	Simulate the C-arm in VR	Done	Rig with kinematics	02/20	03/03	Open-source Asset
IRB Approval	Approval to conduct the user study	Planned	Run the user study	03/31	After semester (only needed for maximum deliverable)	N/A

Table 2. Dependencies

3.3 Timeline & Gantt Chart

The figure below is our timeline. We will first concentrate on the minimum requirements, then the expected and try our best to complete the maximum. The black arrow of some tasks indicate the topological dependency order. The user study testing plan is a maximum deliverable, and will likely not be completed within the end of this semester, but we intend to continue work on this project to complete the user study even after this course.

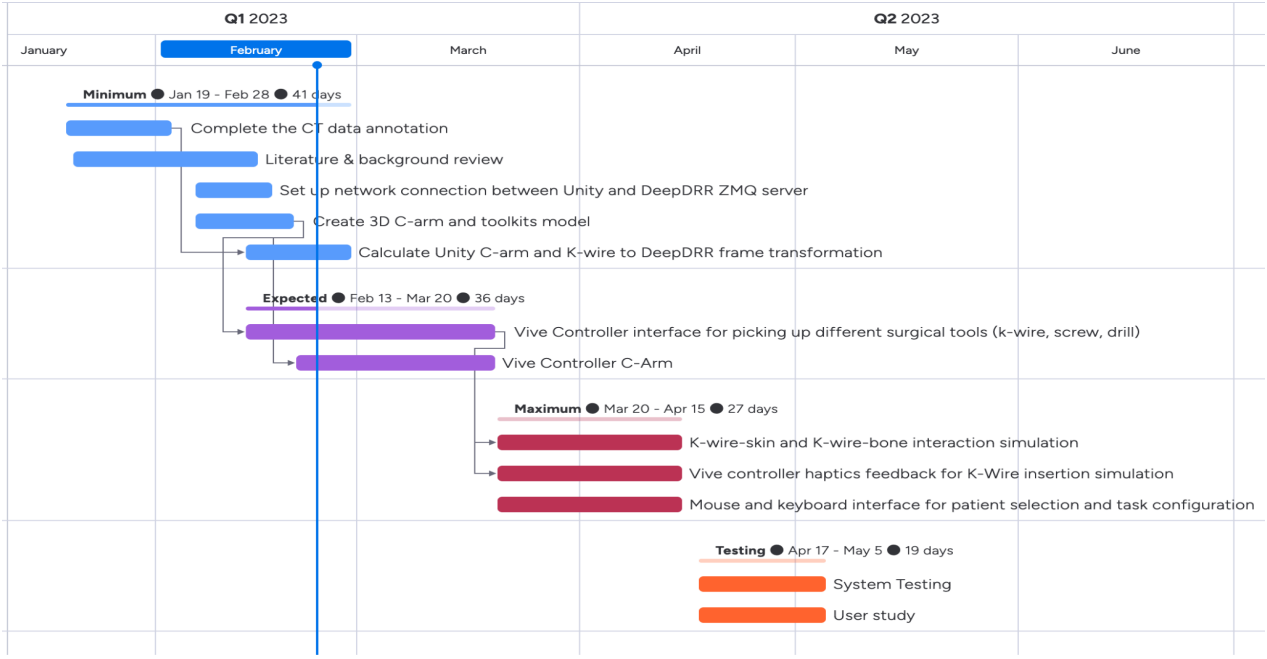

Figure 3. Project Timeline

3.4 Team Responsibility

Han Zhang

Master's Student, BME Major, responsible for interactive C-arm & Tool model and Kinematics.

Zixuan Liu

Master's Student, CS Major, responsible for VR tools interaction and k-wire simulation.

Liam Wang

Undergraduate Junior, BME & CS Major, responsible for DeepDRR ZMQ Server and Unity Client.

3.5 Management Plan

The team will hold weekly meetings that include a student team meeting for brainstorming and a mentor meeting and lab meeting for progress reports. Communication will take 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 will also use Jira and GanttProject for task timeline and administrative tasks. This plan ensures that everyone is on the same page and progress can be tracked effectively throughout the project.

4. Reading List

- 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
- 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
- Meyer, Q. (2006). *Camera Calibration of C-Arm devices.* http://wwwmayr.in.tum.de/konferenzen/MB-Jass2006/courses/1/slides/h-1-5.pdf
- *Orthopedic Hardware.* (n.d.). UW Radiology. Retrieved February 22, 2023, from https://rad.washington.edu/about-us/academic-sections/musculoskeletal-radiology/teaching-materials/online-musculoskeletal-radiology-book/orthopedic-hardware/
- Unberath, M., Zaech, J.-N., Gao, C., Bier, B., Goldmann, F., Lee, S. C., Fotouhi, J., Taylor, R., Armand, M., & Navab, N. (2019). *Enabling machine learning in X-ray-based procedures via realistic simulation of image formation*. International Journal of Computer Assisted Radiology and Surgery, 14(9), 1517–1528. https://doi.org/10.1007/s11548-019-02011-2

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.