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

Interface Specification

Han Zhang
Zixuan Liu
Liam Wang

Table of Contents

1. 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.

1.1 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.

1.2 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.