Unreal Engine Plugin Integrating AI Automatic Segmentation and 3D Reconstruction of Liver MRI Images for VR

Haoxiang Yang

Bournemouth University

September 20, 2025

MSc Computer Animation and Visual Effects
NCCA - Bournemouth University

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisors Jon Macey and Jian Chang. This master's project was completed under the careful guidance of Jon Macey and Jian Chang, and both the development of the project and the writing of this thesis would not have been possible without their patient instruction. I am particularly grateful to Jon Macey for his tremendous understanding and support during my most difficult times, for encouraging me to continue my studies and develop my master's project, for giving me confidence and courage, and for constantly encouraging me to move forward.

I am grateful to be a member of NCCA at Bournemouth University, and I thank all the teachers who guided me, and all the friends and colleagues who helped and supported me during my MSc Computer Animation and Visual Effects course.

Abstract

Addressing the need for 3D anatomical structure visualization in liver disease diagnosis, treatment, and medical education, this project has developed an Unreal Engine plugin capable of loading medical images, running AI inference, and reconstructing 3D models. The plugin can process liver MRI data in .nii.gz format, implement AI automatic segmentation based on MONAI's SegResNet deep learning, perform 3D reconstruction and result analysis based on the Marching Cubes algorithm. The system adopts a client-server architecture, achieving automatic segmentation of MRI images and 3D visualization. The frontend uses Unreal Engine to provide an interactive interface and real-time rendering, while the backend is based on FastAPI and PyTorch for AI inference. Through RESTful API for cross-language communication, it supports multi-organ segmentation of liver, blood vessels, and tumors, and automatically generates 3D reconstruction models. In VR applications, a VR menu has been developed that allows grasping and rotating operations on the liver and blood vessels in VR, laying the foundation for further development of VR visualization tools.

Table of contents

1 Introduction

The liver is the largest solid organ in the human body, with a complex anatomical structure including liver parenchyma, hepatic blood vessels, and potentially existing tumors. The anatomical structure of the liver is crucial for the diagnosis and treatment of liver diseases. Magnetic Resonance Imaging (MRI), with its advantages of no radiation and high soft tissue resolution, has become the preferred imaging method for liver anatomical structure and lesion assessment. However, liver region segmentation and 3D visualization of MRI images in clinical practice still face technical bottlenecks. On one hand, traditional manual segmentation relies on physician experience, which is not only inefficient but also subject to subjective errors, making it difficult to meet the timeliness requirements of surgical planning and preoperative simulation. On the other hand, existing 3D reconstruction tools are mostly standalone software, such as Mimics and 3D Slicer. Although they can achieve liver model reconstruction, there is a "data gap" in connecting with virtual reality (VR) environments. Reconstructed models need to undergo format conversion and lightweight processing before they can be imported into VR platforms, a process that is cumbersome and prone to losing anatomical details, limiting the application of VR in liver surgery navigation and medical education.

In recent years, artificial intelligence (AI) technology has provided new techniques for automatic liver MRI segmentation. Segmentation models based on U-Net and its improved architectures have achieved precise liver region segmentation, significantly reducing the cost of manual intervention. However, current research mostly focuses on optimizing the accuracy of AI segmentation algorithms, neglecting the full-process integration of "segmentation - reconstruction - VR interaction". Most AI segmentation models only output 2D masks or offline 3D models, lacking deep coupling with mainstream VR development engines (such as Unreal Engine), unable to support real-time rendering and interactive operations of real-time segmentation results in VR. Unreal Engine, as a development tool with photorealistic rendering capabilities and multi-platform VR compatibility, has been applied in the medical VR field, but there is no specialized "AI segmentation - 3D reconstruction" integrated plugin for liver MRI images.

Addressing the above issues, this research aims to develop an Unreal Engine plugin capable of loading medical images, running AI inference, and reconstructing 3D models. The plugin can provide full-process support for data preprocessing, AI segmentation invocation, real-time 3D reconstruction, and VR interaction adaptation, bridging the data interface between AI segmentation models and Unreal Engine, supporting direct embedding of SegResNet-trained segmentation models into the plugin, achieving automatic MRI image segmentation and 3D reconstruction of segmentation results, and laying the foundation for the next step of VR interactive components (such as gesture control, model annotation, and cutting view). This plugin provides efficient and convenient tool support for precise diagnosis and treatment of liver diseases and medical education, promoting the deep integration and development of medical imaging technology and VR technology.

2 Previous Work

In the field of medical image analysis, automatic segmentation technology for liver MRI images has made significant progress in recent years. Deep learning methods, especially U-Net and its variants, have become mainstream. The 3D residual U-Net model proposed in 2023 performed excellently in liver MRI image segmentation, with Dice similarity coefficients for liver parenchyma and tumor segmentation reaching 0.92±0.03, demonstrating the potential of deep learning in complex liver anatomical structure segmentation. However, these high-precision segmentation models mostly focus on improving algorithm accuracy, with less consideration for seamless integration with subsequent visualization and interaction platforms, especially lacking optimization for virtual reality (VR) environments.

3D reconstruction technology for medical images provides important support for clinical diagnosis and surgical planning. Early liver 3D reconstruction mainly relied on traditional image processing algorithms. For example, the Slice Growing Method (SGM) combined with level set algorithm proposed by Alom et al., achieved differentiation between liver and adjacent organs through curvature control, laying the foundation for 3D modeling of liver and vascular structures.

In terms of data acquisition, MRI can non-invasively obtain soft tissue structure information of the liver. The acquired MRI raw data needs to be preprocessed, including denoising, interpolation, and filtering operations to improve image quality and accuracy of subsequent processing. In the image segmentation stage, commonly used segmentation algorithms include threshold method, region growing method, and morphology-based methods, through which liver tissue is separated from MRI images.

Surface reconstruction and volume reconstruction are commonly used 3D reconstruction methods. Surface reconstruction is suitable for organs with clear contours, constructing 3D models by extracting feature points on the liver surface; volume reconstruction is more suitable for displaying the complex internal structure of the liver, such as ray casting method, emitting rays from the viewpoint, passing through 3D volume data, sampling at the intersection of rays and voxels, calculating their color and transparency based on voxel attributes, and then compositing these sampling points' colors and transparency to obtain the final 2D projected image.

VR technology has shown great value in medical applications. Existing virtual reality systems present the complex internal structure of human organs in high-fidelity 3D through 3D visualization technology, but mostly rely on preoperative static reconstruction models, lacking dynamic update capabilities based on real-time AI segmentation, making it difficult to respond to dynamic changes in anatomical structures during surgery.

In terms of visualization engines, Unreal Engine, with its powerful real-time rendering capabilities, has gradually become an important platform for medical image 3D visualization. Existing plugins mainly focus on rendering functions and have not integrated AI automatic segmentation modules, resulting in users still needing to rely on external software to complete image segmentation before importing into the engine for visualization, a process that is cumbersome and difficult to achieve real-time updates. In summary, although liver MRI segmentation, 3D reconstruction, and VR visualization technologies have each made significant progress, there are still obvious technical barriers in clinical applications: high-precision AI segmentation models lack deep integration with VR visualization platforms, existing medical visualization plugins have difficulty meeting real-time and interactive requirements, and these limitations provide a clear innovation direction for this research to develop an Unreal Engine plugin integrating AI automatic segmentation and 3D reconstruction functions.

3 Technical Background

The core technologies of this project involve medical image AI automatic segmentation technology and 3D reconstruction technology.

3.1 Deep Learning Segmentation Methods

Deep learning methods automatically learn image features through neural networks, combined with a large amount of labeled data to train models, with good segmentation accuracy and robustness, and have become a new technology for current liver segmentation.

Encoder-Decoder based segmentation models are the most classic architecture for medical image segmentation, with the core being downsampling (Encoder) to extract high-level semantic features and upsampling (Decoder) to recover spatial details. The representative model is U-Net and its variants.

Basic U-Net model structure features:

Encoder: Gradually reduces feature map size through convolution and pooling, increases channel number, and extracts global semantic features;

Decoder: Gradually enlarges feature map size through transpose convolution, reduces channel number;

Skip connections: Directly pass low-level features from Encoder to corresponding layers in Decoder, solving the detail loss problem during upsampling and improving boundary segmentation accuracy.

3.2 3D Reconstruction Techniques and Algorithms

Methods for converting liver segmentation masks to 3D models mainly include voxel stacking, surface reconstruction, parametric modeling, and deep learning direct generation. A comparison of the four methods is as follows:

Voxel stacking directly stacks 2D masks in sequence to form a 3D voxel array, then displays using a 3D rendering engine. This method is simple to implement and can fully preserve segmentation accuracy, but has large data volume, non-smooth surfaces, average display effect, and poor interactive performance, making it difficult to directly use for 3D printing or fine mesh analysis.

Surface reconstruction extracts isosurfaces from voxel masks to generate triangular mesh models. Common algorithms include Marching Cubes and Contour Stitching. This method generates lightweight models with smooth surfaces, suitable for visualization and interaction, and can be directly exported as STL/OBJ for 3D printing, but requires adjustment of smoothing parameters, otherwise "jagged" edges may appear, and is sensitive to voxel resolution.

Parametric modeling uses geometric shapes or statistical models to fit segmentation results. This method produces compact models, facilitating shape analysis and biomechanical modeling, but has difficulty fitting complex anatomical structures, depends on good initialization and parameter adjustment, and may not fully reflect individual differences.

Deep learning direct generation uses 3D GAN or implicit neural representation (NeRF, SDF) to directly generate 3D models from segmentation masks or original images. This method can generate high-fidelity, smooth models and can learn shape priors from data, but requires a large amount of labeled data for training, has complex models, time-consuming inference, and poor result interpretability.

4 Project Solution

This project is a medical image AI analysis plugin developed based on Unreal Engine, focusing on intelligent analysis and 3D visualization of liver MRI images. The system adopts a front-end and back-end separation architecture, combining deep learning technology to achieve automatic segmentation and diagnostic assistance for liver, blood vessels, and tumors.

4.1 AI Automatic Segmentation System

A three-model cascade AI segmentation scheme is adopted, separately training liver segmentation model, blood vessel segmentation model, and tumor segmentation model.

4.1.1 Deep Learning Network Model Construction

For each segmentation task, SegResNet is used as the deep learning network, which combines U-Net structure with residual block design to enhance feature extraction capability and optimize gradient propagation.

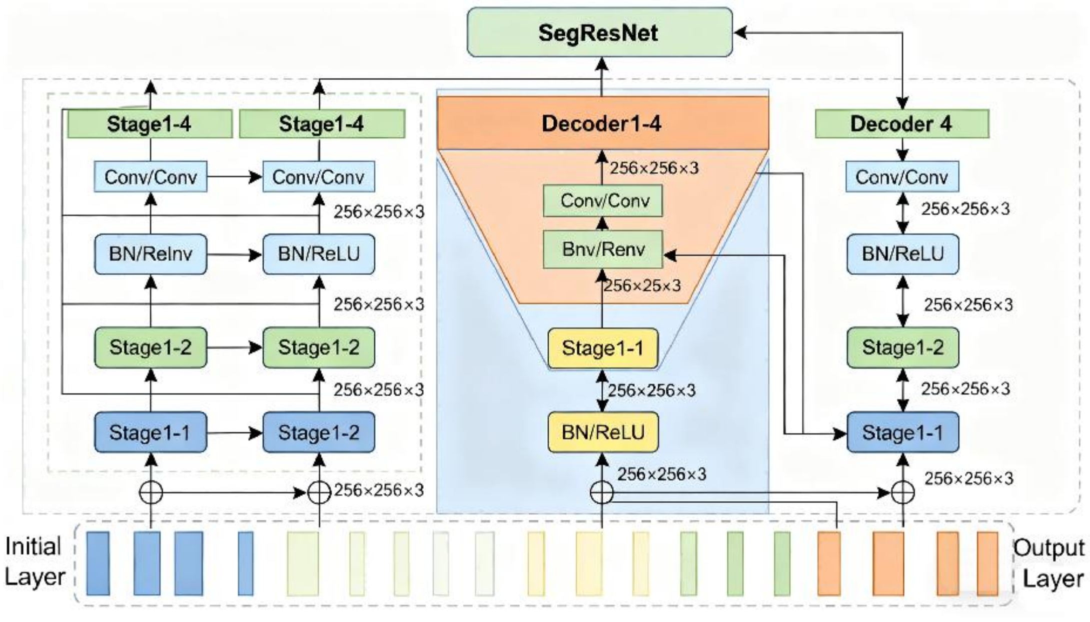


Figure 1 SegResNet Network Structure Diagram

MONAI's built-in 3D SegResNet is selected as the segmentation core. This network is designed based on residual connections, which can effectively alleviate the gradient vanishing problem of deep networks and is suitable for 3D medical image segmentation. The configuration is as follows:

Input channels: in_channels=1 (single-channel MRI);

Output channels: out_channels=2 (corresponding to background class 0, liver class 1);

Network depth: Default 5 layers, adjustable through depth parameter, gradually extracting spatial features from low-dimensional to high-dimensional.

4.1.2 Model Training

(1) Dataset Loader

MONAI's CacheDataset and PyTorch's DataLoader are used to achieve efficient data loading. CacheDataset caches preprocessed data in memory, avoiding repeated preprocessing in each training iteration and improving loading speed; supports multi-threaded preprocessing, further optimizing efficiency. DataLoader can shuffle sample order for training set, dedicated batch processing concatenation, support variable-length data, and infer samples one by one for validation set to avoid sliding window concatenation errors and maintain sample order.

The liver, blood vessel, and tumor datasets are shown in Figures 2 to 7.

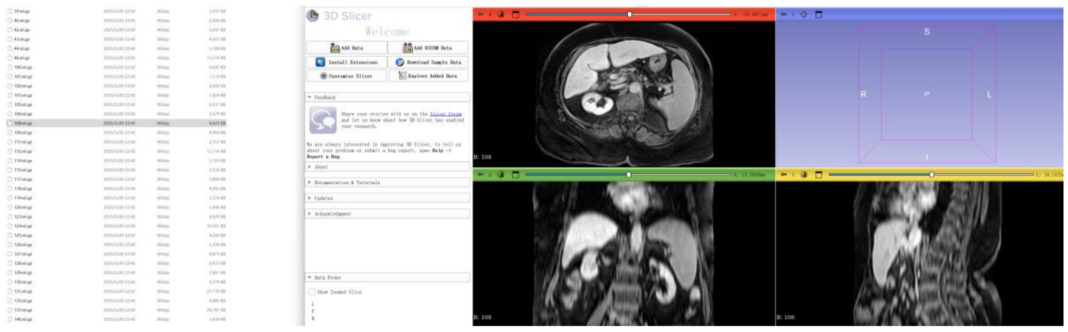


Figure 2 Liver Dataset Image

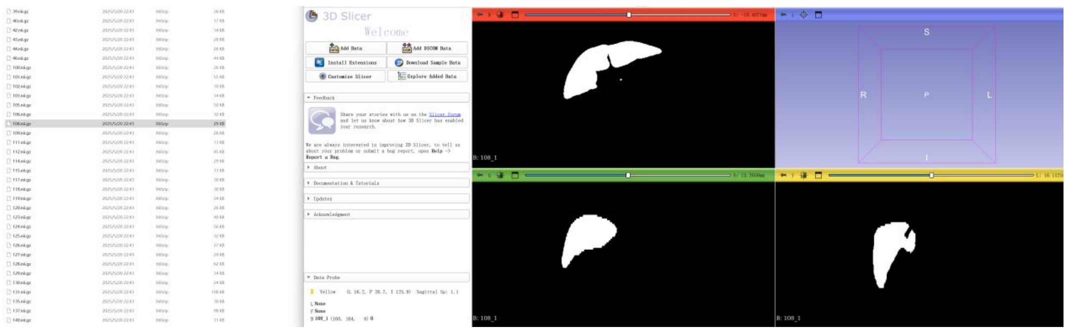


Figure 3 Liver Dataset Label

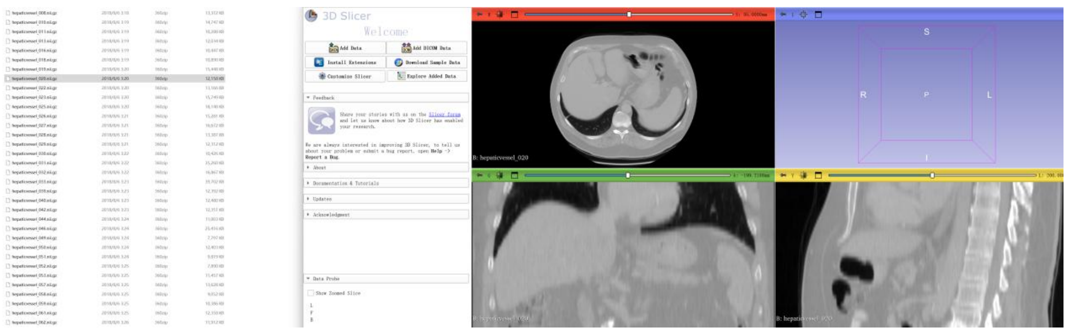


Figure 4 Vessel Dataset Image

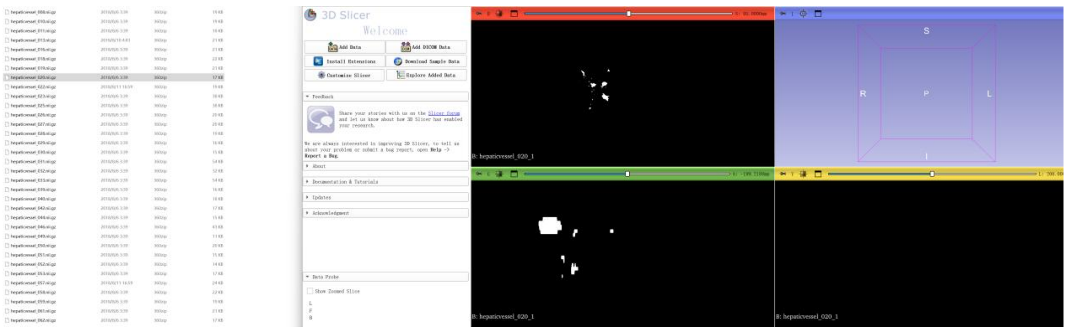


Figure 5 Vessel Dataset Label

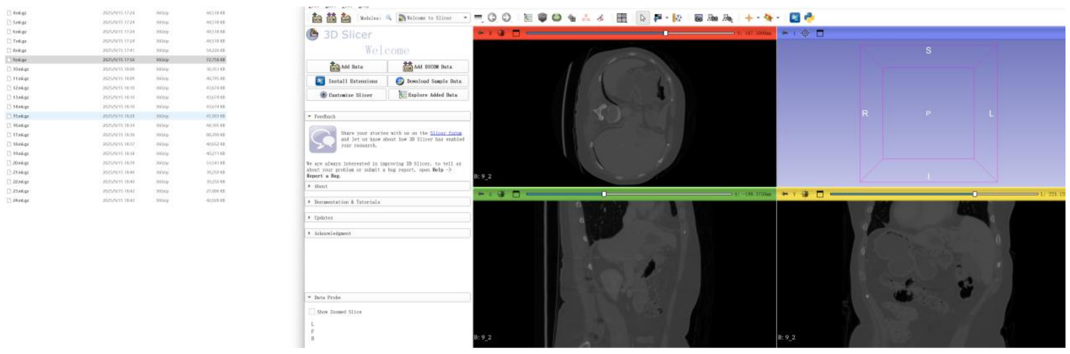


Figure 6 Tumor Dataset Image

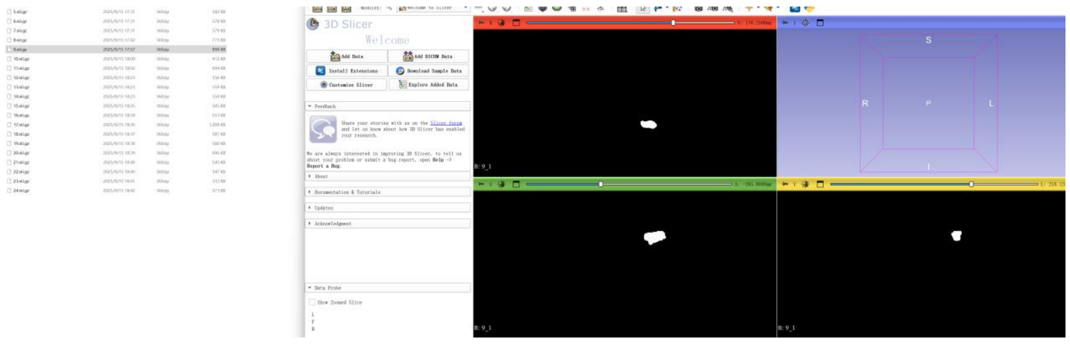


Figure 7 Tumor Dataset Label

(2) Optimizer and Learning Rate Scheduling

Adam optimizer is selected with learning rate LEARNING_RATE=1e-4, momentum parameters betas=(0.9, 0.999), suitable for stable convergence of medical image segmentation tasks.

Learning rate scheduling adopts the Reduce Learning Rate on Plateau strategy. When the validation metric Dice coefficient does not improve for patience=10 consecutive epochs, the learning rate is multiplied by factor=0.5 to decay by 50%; if the scheduling criterion is that the larger the Dice coefficient the better, set mode="max"; if the criterion is that the smaller the validation loss the better, set mode=min.

(3) Loss Function Design

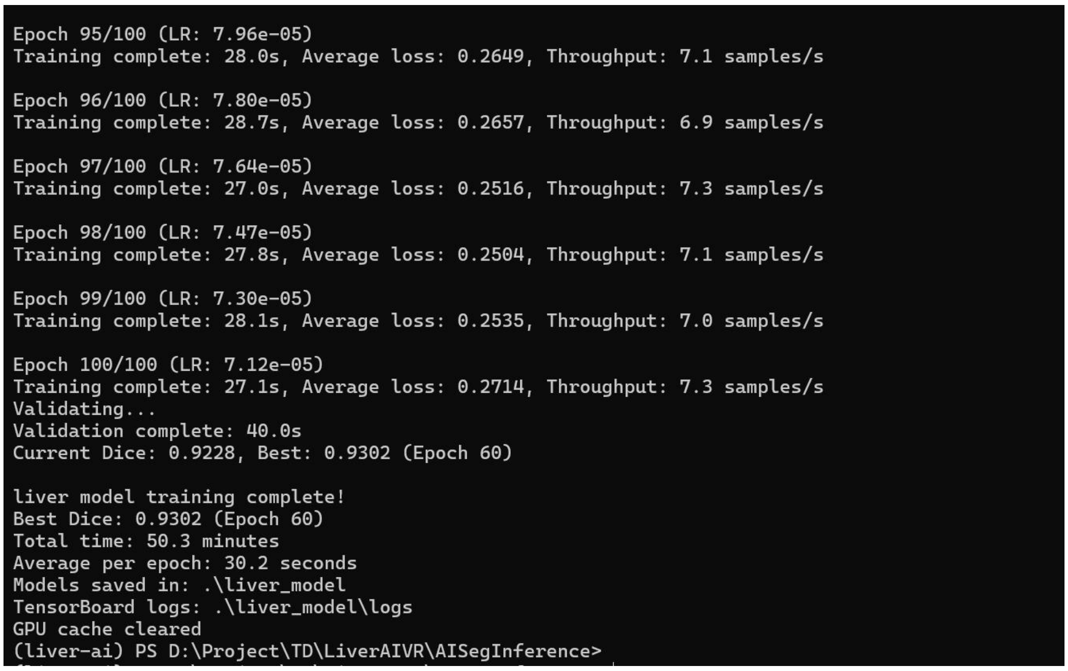
Segmentation loss adopts a hybrid loss composed of Dice loss and CE loss. CE loss calculates loss in pixel units, calculating loss for each pixel and then summing, converging faster. Dice loss calculates in class pixel set units, and after converging to a certain extent, can further show higher precision in segmentation results.

(4) Evaluation Metrics

Dice coefficient is used to measure the overlap between segmentation results and gold standard, with a range of [0, 1], where closer to 1 indicates more accurate segmentation.

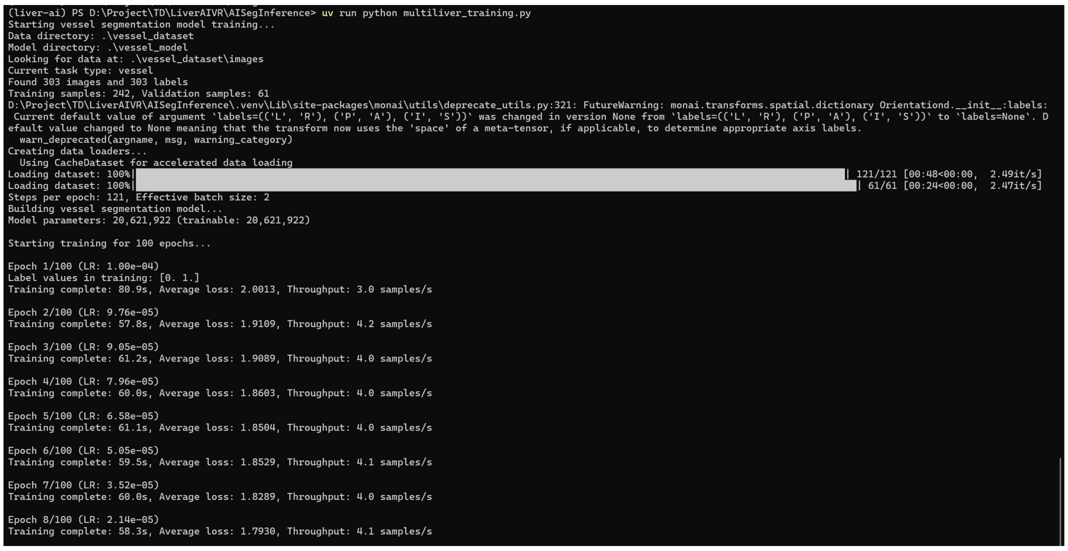
(5) Training Process

Each model is trained for 100 epochs. The liver Dice similarity coefficient is 0.9302, proving the effectiveness of this training method. The liver training process window is shown in Figure 8, the vessel training process window is shown in Figure 9, and the tumor training process window is shown in Figure 10.



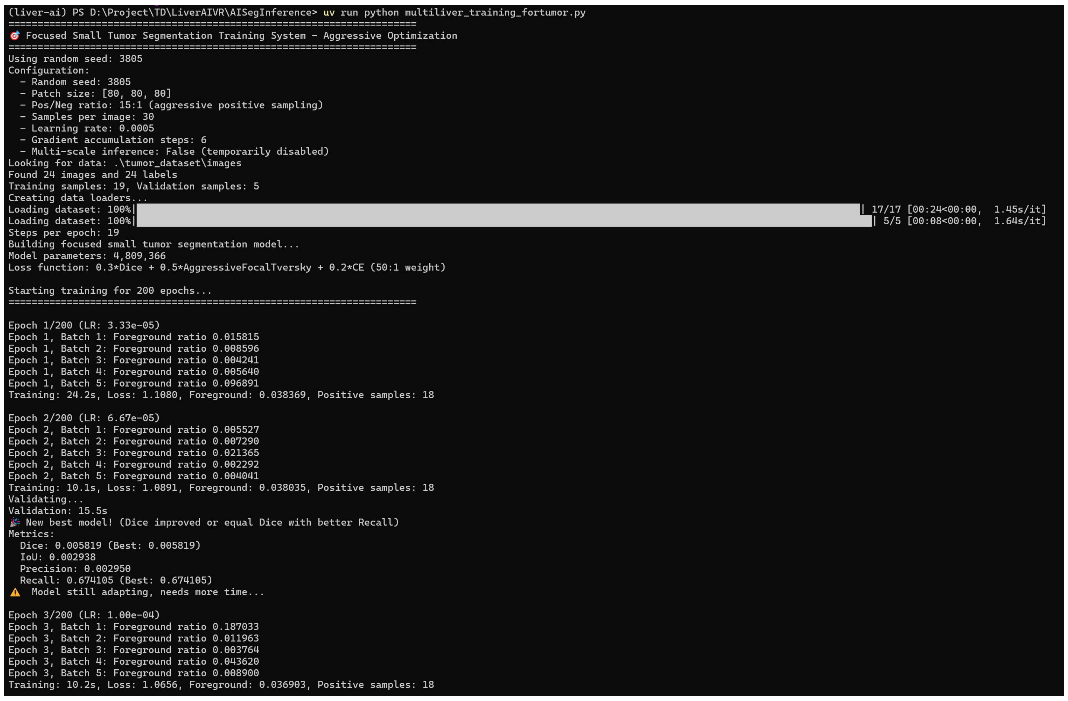
```
Epoch 95/100 (LR: 7.96e-05)
Training complete: 28.0s, Average loss: 0.2649, Throughput: 7.1 samples/s

Epoch 96/100 (LR: 7.80e-05)
Training complete: 28.7s, Average loss: 0.2657, Throughput: 6.9 samples/s

Epoch 97/100 (LR: 7.64e-05)
Training complete: 27.0s, Average loss: 0.2516, Throughput: 7.3 samples/s

Epoch 98/100 (LR: 7.47e-05)
Training complete: 27.8s, Average loss: 0.2504, Throughput: 7.1 samples/s

Epoch 99/100 (LR: 7.30e-05)
Training complete: 28.1s, Average loss: 0.2535, Throughput: 7.0 samples/s

Epoch 100/100 (LR: 7.12e-05)
Training complete: 27.1s, Average loss: 0.2714, Throughput: 7.3 samples/s
Validating...
Validation complete: 40.0s
Current Dice: 0.9228, Best: 0.9302 (Epoch 60)

liver model training complete!
Best Dice: 0.9302 (Epoch 60)
Total time: 50.3 minutes
Average per epoch: 30.2 seconds
Models saved in: .\liver_model
TensorBoard logs: .\liver_model\logs
GPU cache cleared
(liver-ai) PS D:\Project\TD\LiverAIVR\AISegInference>
```

Figure 8 AI Training Liver Model



```
(liver-ai) PS D:\Project\TD\LiverAIVR\AISegInference> uv run python multiliver_training.py
Starting vessel segmentation model training...
Data directory: .\vessel_dataset
Model directory: .\vessel_model
Looking for data at: .\vessel_dataset\images
Current task type: vessel
Found 303 images and 303 labels
Training samples: 242, Validation samples: 61
D:\Project\TD\LiverAIVR\AISegInference\.venv\Lib\site-packages\monai\utils\deprecate_utils.py:321: FutureWarning: monai.transforms.spatial.dictionary Orientationd.__init__:labels:
 Current default value of argument `labels=(('L', 'R'), ('P', 'A'), ('I', 'S'))` was changed in version None from `labels=(('L', 'R'), ('P', 'A'), ('I', 'S'))` to `labels=None`. D
efault value changed to None meaning that the transform now uses the 'space' of a meta-tensor, if applicable, to determine appropriate axis labels.
  warn_deprecated(argname, msg, warning_category)
Creating data loaders...
  Using CacheDataset for accelerated data loading
Loading dataset: 100%|██████████████████████████████████████████████████| 121/121 [00:48<00:00,  2.49it/s]
Loading dataset: 100%|██████████████████████████████████████████████████| 61/61 [00:24<00:00,  2.47it/s]
Steps per epoch: 121, Effective batch size: 2
Building vessel segmentation model...
Model parameters: 20,621,922 (trainable: 20,621,922)

Starting training for 100 epochs...

Epoch 1/100 (LR: 1.00e-04)
Label values in training: [0. 1.]
Training complete: 80.9s, Average loss: 2.0013, Throughput: 3.0 samples/s

Epoch 2/100 (LR: 9.76e-05)
Training complete: 57.8s, Average loss: 1.9109, Throughput: 4.2 samples/s

Epoch 3/100 (LR: 9.05e-05)
Training complete: 61.2s, Average loss: 1.9089, Throughput: 4.0 samples/s

Epoch 4/100 (LR: 7.96e-05)
Training complete: 60.0s, Average loss: 1.8603, Throughput: 4.0 samples/s

Epoch 5/100 (LR: 6.58e-05)
Training complete: 61.1s, Average loss: 1.8504, Throughput: 4.0 samples/s

Epoch 6/100 (LR: 5.05e-05)
Training complete: 59.5s, Average loss: 1.8529, Throughput: 4.1 samples/s

Epoch 7/100 (LR: 3.52e-05)
Training complete: 60.0s, Average loss: 1.8289, Throughput: 4.0 samples/s

Epoch 8/100 (LR: 2.14e-05)
Training complete: 58.3s, Average loss: 1.7930, Throughput: 4.1 samples/s
```

Figure 9 Vessel Training Process



```
(liver-ai) PS D:\Project\TD\LiverAIVR\AISegInference> uv run python multiliver_training_fortumor.py
======================================================================
🎯 Focused Small Tumor Segmentation Training System - Aggressive Optimization
======================================================================
Using random seed: 3805
Configuration:
  - Random seed: 3805
  - Patch size: [80, 80, 80]
  - Pos/Neg ratio: 15:1 (aggressive positive sampling)
  - Samples per image: 30
  - Learning rate: 0.0005
  - Gradient accumulation steps: 6
  - Multi-scale inference: False (temporarily disabled)
Looking for data: .\tumor_dataset\images
Found 24 images and 24 labels
Training samples: 19, Validation samples: 5
Creating data loaders...
Loading dataset: 100%|██████████████████████████████████████████████████| 17/17 [00:24<00:00,  1.45s/it]
Loading dataset: 100%|██████████████████████████████████████████████████| 5/5 [00:08<00:00,  1.64s/it]
Steps per epoch: 19
Building focused small tumor segmentation model...
Model parameters: 4,809,366
Loss function: 0.3*Dice + 0.5*AggressiveFocalTversky + 0.2*CE (50:1 weight)

Starting training for 200 epochs...
======================================================================

Epoch 1/200 (LR: 3.33e-05)
Epoch 1, Batch 1: Foreground ratio 0.015815
Epoch 1, Batch 2: Foreground ratio 0.008596
Epoch 1, Batch 3: Foreground ratio 0.004241
Epoch 1, Batch 4: Foreground ratio 0.005640
Epoch 1, Batch 5: Foreground ratio 0.096891
Training: 24.2s, Loss: 1.1080, Foreground: 0.038369, Positive samples: 18

Epoch 2/200 (LR: 6.67e-05)
Epoch 2, Batch 1: Foreground ratio 0.005527
Epoch 2, Batch 2: Foreground ratio 0.007290
Epoch 2, Batch 3: Foreground ratio 0.021365
Epoch 2, Batch 4: Foreground ratio 0.002292
Epoch 2, Batch 5: Foreground ratio 0.004041
Training: 10.1s, Loss: 1.0891, Foreground: 0.038035, Positive samples: 18
Validating...
Validation: 15.5s
🎉 New best model! (Dice improved or equal Dice with better Recall)
Metrics:
  Dice: 0.005819 (Best: 0.005819)
  IoU: 0.002938
  Precision: 0.002950
  Recall: 0.674105 (Best: 0.674105)
⚠️  Model still adapting, needs more time...

Epoch 3/200 (LR: 1.00e-04)
Epoch 3, Batch 1: Foreground ratio 0.187033
Epoch 3, Batch 2: Foreground ratio 0.011963
Epoch 3, Batch 3: Foreground ratio 0.003764
Epoch 3, Batch 4: Foreground ratio 0.043620
Epoch 3, Batch 5: Foreground ratio 0.008900
Training: 10.2s, Loss: 1.0656, Foreground: 0.036903, Positive samples: 18
```

Figure 10 Tumor Training Process

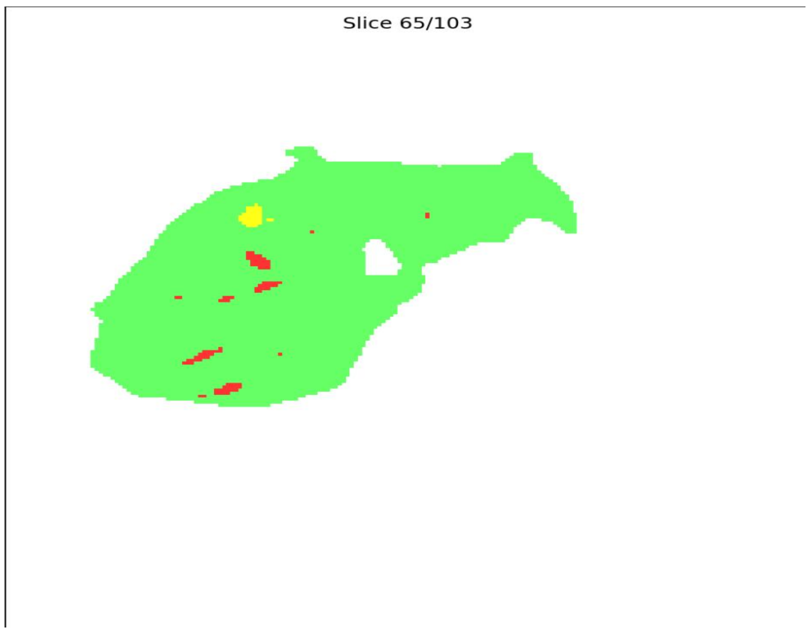


Figure 11 Segmentation Results

Figures 11 and 12 show the model's segmentation results. Through the inference stage, models that meet expectations were obtained. As shown in the figures, green represents liver, red represents blood vessels, and yellow represents tumors, and text reports were generated. Figure 13 is the text report window, showing liver, vessel, and tumor volumes and percentages.

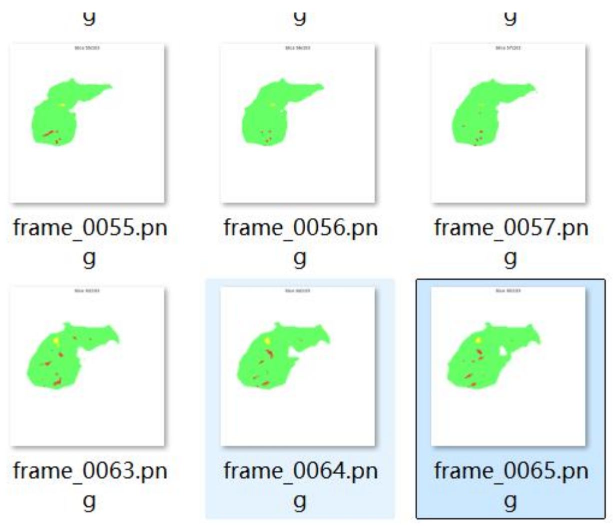
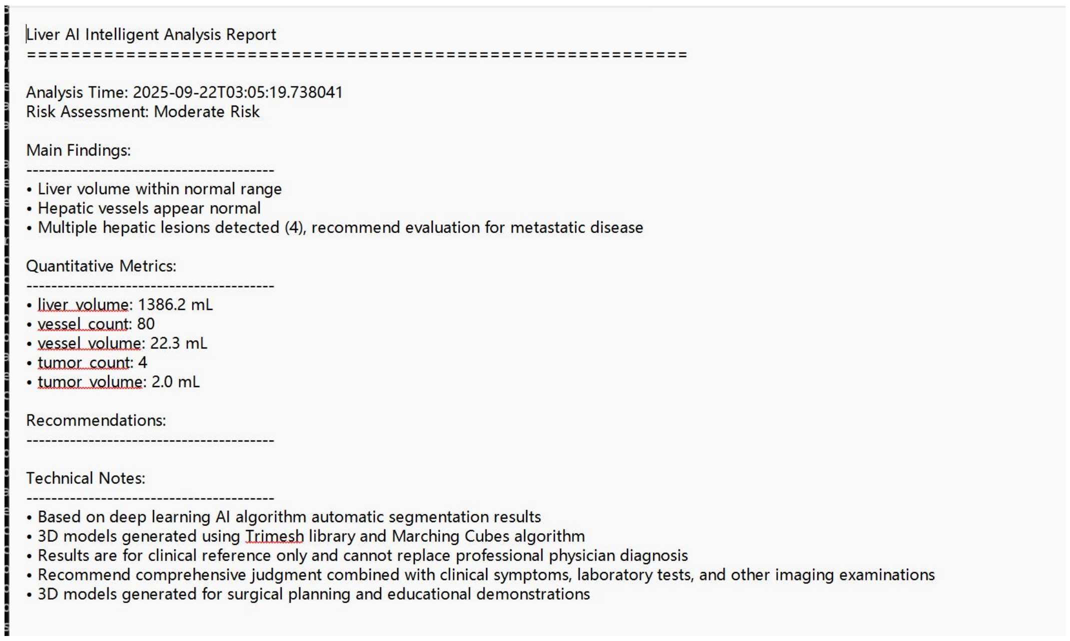


Figure 12 Segmentation Results



```
Liver AI Intelligent Analysis Report
============================================================

Analysis Time: 2025-09-22T03:05:19.738041
Risk Assessment: Moderate Risk

Main Findings:
----------------------------------------
• Liver volume within normal range
• Hepatic vessels appear normal
• Multiple hepatic lesions detected (4), recommend evaluation for metastatic disease

Quantitative Metrics:
----------------------------------------
• liver_volume: 1386.2 mL
• vessel_count: 80
• vessel_volume: 22.3 mL
• tumor_count: 4
• tumor_volume: 2.0 mL

Recommendations:
----------------------------------------

Technical Notes:
----------------------------------------
• Based on deep learning AI algorithm automatic segmentation results
• 3D models generated using Trimesh library and Marching Cubes algorithm
• Results are for clinical reference only and cannot replace professional physician diagnosis
• Recommend comprehensive judgment combined with clinical symptoms, laboratory tests, and other imaging examinations
• 3D models generated for surgical planning and educational demonstrations
```

Figure 13 Inference Result Structured Diagnostic Text Report Window

The model meets the expected functions, segmenting liver, blood vessels, and tumors. If users input MRI images with tumors, yellow tumors can be detected. Of course, due to the limited tumor dataset and small tumor size, missed diagnoses may occur. The model can meet anatomical teaching needs. However, for clinical diagnostic needs, a large amount of liver tumor data needs to be added.

4.2 3D Model Generation

The Marching Cubes algorithm is a commonly used algorithm in medical image 3D reconstruction. Due to its relatively simple and fast computation, it has wide applications. This project adopts the Marching Cubes algorithm. The basic idea of the algorithm is to discretize continuous space, divide volume data into small hexahedral units (voxels), and perform isosurface reconstruction on this basis. The construction of isosurfaces within each voxel is achieved by determining the intersection points between the isosurface and voxel boundaries, and using linear interpolation technology to determine the precise positions of these intersection points. The value of each vertex of the voxel is compared with a preset threshold to determine whether the isosurface is within the voxel. Based on different combinations of vertex values, a voxel can be decomposed into up to five triangles, and the organization of these triangles depends on the intersection situation between the isosurface and the voxel. The algorithm is implemented using an iterative strategy, systematically traversing the entire volume dataset, connecting triangle patches generated by voxel division, and finally completing 3D reconstruction. The eight vertices of a voxel have two states, namely 0 and 1, corresponding to vertex values less than the isosurface value and vertex values greater than the isosurface value, respectively, so there are a total of 2^8 = 256 combination states. According to the translation, rotation, and symmetry characteristics of voxels, the 256 states can be simplified to 15 configurations, as shown in Figure 14.

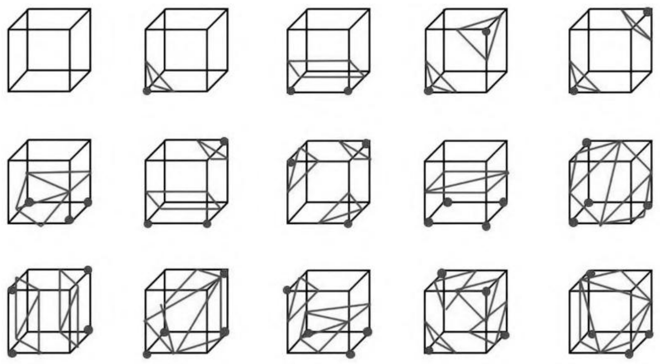


Figure 14 15 Cases of Isosurface Intersecting with Cube

After determining the basic configuration, the specific position of triangle patch vertices can be obtained by calculating the linear interpolation between the isosurface value and the vertex values of the intersecting edges. Where P represents the intersection coordinate, P1 and P2 represent the coordinates of the two endpoints on the edge, V1 and V2 represent the values at these two endpoints, and V represents the isosurface.

$$P = P1 + \frac{(V - V1) * (P2 - P1)}{V2 - V1} \tag{1}$$

Using the Marching Cubes algorithm for modeling yields 3D models of liver, blood vessels, and tumors, as shown in Figure 15:

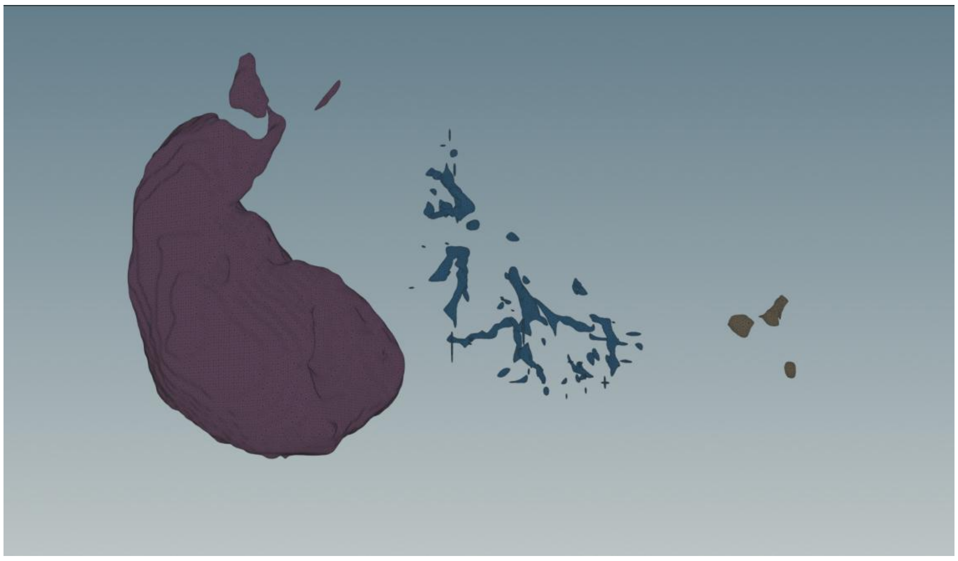
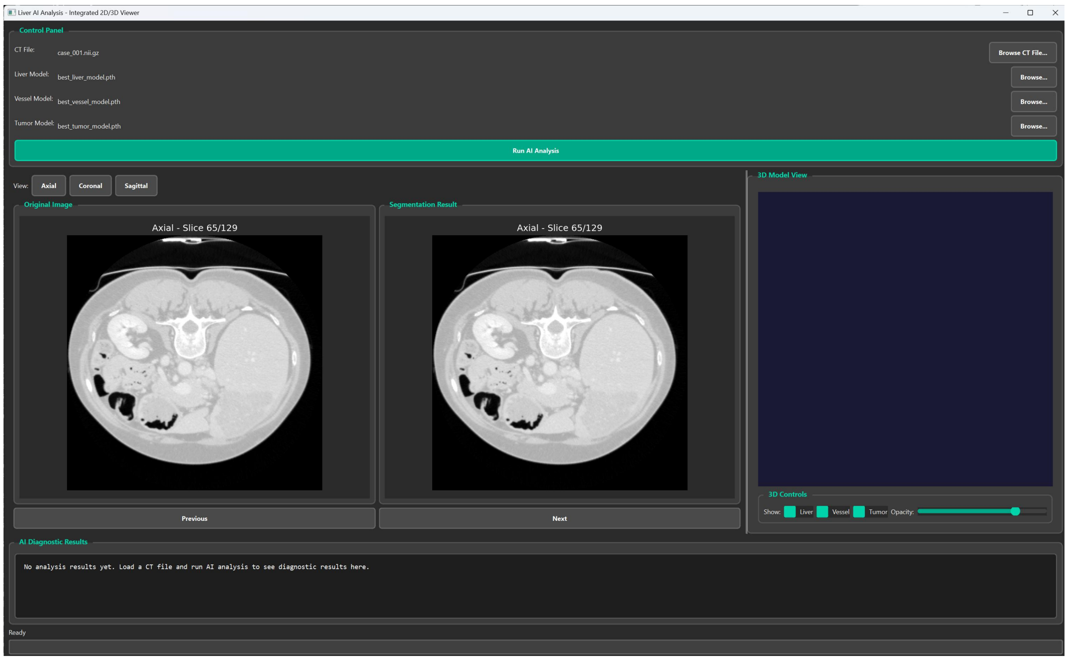


Figure 15 3D Models

Finally, the system's terminal running interface is shown in Figure 16.



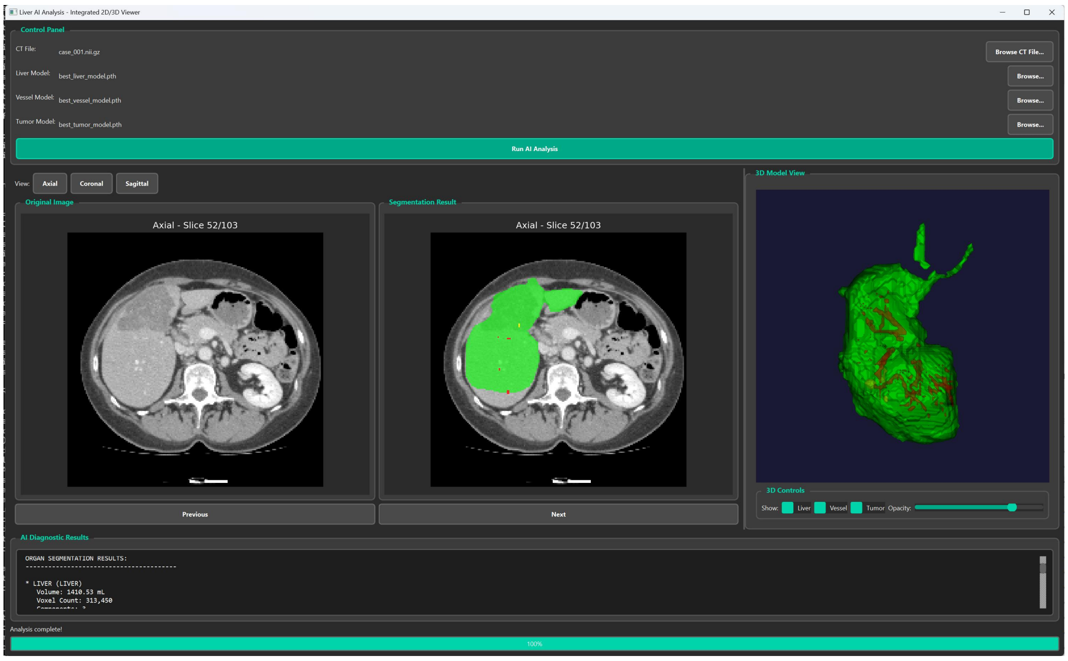
CT File: case_001.nii.gz
Liver Model: best_liver_model.pth
Vessel Model: best_vessel_model.pth
Tumor Model: best_tumor_model.pth

```
No analysis results yet. Load a CT file and run AI analysis to see diagnostic results here.
```

Ready

Figure 16 Terminal Running Interface

The system's terminal running results are shown in Figure 17.



CT File: case_001.nii.gz
Liver Model: best_liver_model.pth
Vessel Model: best_vessel_model.pth
Tumor Model: best_tumor_model.pth

```
ORGAN SEGMENTATION RESULTS:
----------------------------------------

* LIVER (LIVER)
   Volume: 1410.53 mL
   Voxel Count: 313,450
   Components: 3
```

Analysis complete!

Figure 17 Terminal Running Results

4.3 UE Plugin Integrating AI Segmentation and 3D Reconstruction

The plugin adopts a layered design, with each layer focusing on specific functions and achieving cross-layer collaboration through standardized interfaces, which is both convenient for module maintenance and supports subsequent function expansion, such as adding segmentation for kidneys, lungs, and other organs. The functions of each layer are as follows:

User Interaction Layer: Receives doctor operations, displays analysis progress, visualizes 2D slices and 3D models, provides parameter configuration

Communication Coordination Layer: Encapsulates network requests, manages request lifecycle, handles timeout/retry, parses JSON responses

Backend Service Layer: Provides RESTful API, maintains task queue, coordinates AI model invocation sequence, manages task status

AI Inference Layer: Medical image preprocessing, invokes segmentation models to separately segment liver, blood vessels, and tumors, post-processing optimization of segmentation results

Data Processing Layer: Converts segmentation masks to 3D meshes, calculates organ volume and tumor diameter, generates structured diagnostic reports

Key core technology stack:

Frontend: Unreal Engine 5 + Slate UI framework

Backend: Python FastAPI + PyTorch + MONAI

Communication: HTTP/REST API

AI models: SegResNet, UNet

3D reconstruction: Marching Cubes algorithm + ProceduralMesh

4.3.1 User Interaction Layer (UE Interface)

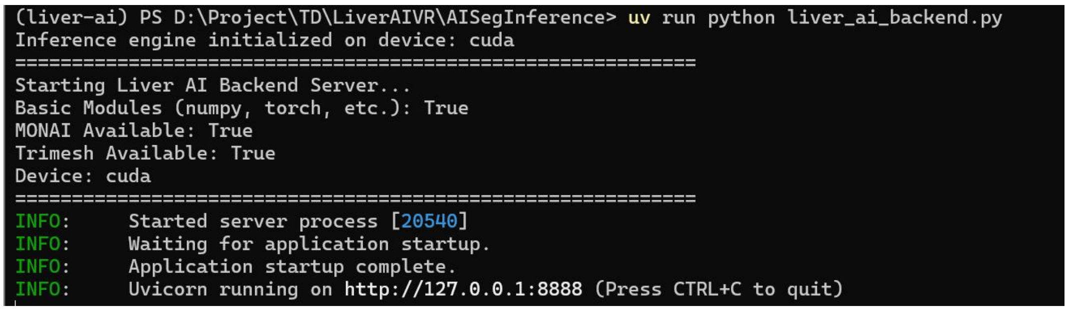
Used for direct interaction between the plugin and users, the user interaction layer needs to balance "functional completeness" and "operational simplicity", with core design around three major scenarios: image input, process monitoring, and result interpretation. Main functions include file and parameter configuration, result visualization, and diagnostic report view. File and parameter configuration supports .nii.gz format file selection, provides path validation for file existence and format compatibility, and configures segmentation tasks such as liver, blood vessel, and tumor segmentation. Result visualization displays organ 3D models through UE, supporting rotation and scaling. Diagnostic report view presents quantified results in structured tables, such as tumor volume and percentage.

4.3.2 Communication Coordination Layer (HTTP Client)

The communication coordination layer encapsulates all network interaction logic between the UE frontend and FastAPI backend, shielding underlying network details and providing a "simple invocation, controllable exceptions" communication interface for the upper layer. Core functions include request encapsulation and sending, response parsing and conversion, and exception handling. Request encapsulation and sending submits analysis tasks and queries progress and results. Response parsing and conversion receives backend JSON responses, decodes Base64 strings of segmentation masks into TArray recognizable by UE, and handles data format differences. Exception handling provides clear error prompts for timeout and connection failures.

4.3.3 Backend Service Layer (FastAPI)

The backend service layer is built based on FastAPI, responsible for receiving frontend requests, scheduling AI models, and managing task states, ensuring stability during simultaneous use by multiple users. RESTful API includes health check interface, task submission interface, progress query interface, and result retrieval interface; task queue and state management uses Celery to build task queue, supports concurrent processing of multiple user analysis requests, avoids backend resource overload; AI model invocation coordination formulates model invocation sequence. Running the PC local server is shown in Figure 18.



```
(liver-ai) PS D:\Project\TD\LiverAIVR\AISegInference> uv run python liver_ai_backend.py
Inference engine initialized on device: cuda
============================================================
Starting Liver AI Backend Server...
Basic Modules (numpy, torch, etc.): True
MONAI Available: True
Trimesh Available: True
Device: cuda
============================================================
INFO:     Started server process [20540]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:8888 (Press CTRL+C to quit)
```

Figure 18 Running PC Local Server

4.3.4 AI Inference Layer (Deep Learning Engine)

The AI inference layer is the "technical core" of the plugin, responsible for converting liver MRI images into precise segmentation masks. The core objective is to balance "high precision" and "high efficiency", adapting to clinical dual demands for accuracy and speed. Core functions include image preprocessing, segmentation model invocation, and post-processing optimization. Image preprocessing performs format conversion and size adjustment; segmentation model invocation includes: liver segmentation using SegResNet model, blood vessel segmentation using SegResNet model; tumor segmentation using UNet+Attention model, enhancing recognition capability for small tumors through attention mechanism. Post-processing optimization includes denoising, hole filling, and result restoration. Denoising removes isolated small regions in segmentation masks through morphological filtering; hole filling fills holes in liver masks to ensure complete organ morphology after 3D reconstruction; result restoration restores segmentation masks to original image size according to the scaling ratio during preprocessing, avoiding position offset.

4.3.5 Data Processing Layer (3D Reconstruction and Analysis)

The data processing layer converts AI-segmented 2D masks into intuitive 3D models and quantified indicators. Core functions include 3D mesh generation, mesh optimization, physical space mapping, and diagnostic report generation.

3D mesh generation: Based on the Marching Cubes algorithm, converts segmentation mask voxel data into triangular meshes, extracting organ surface contours;

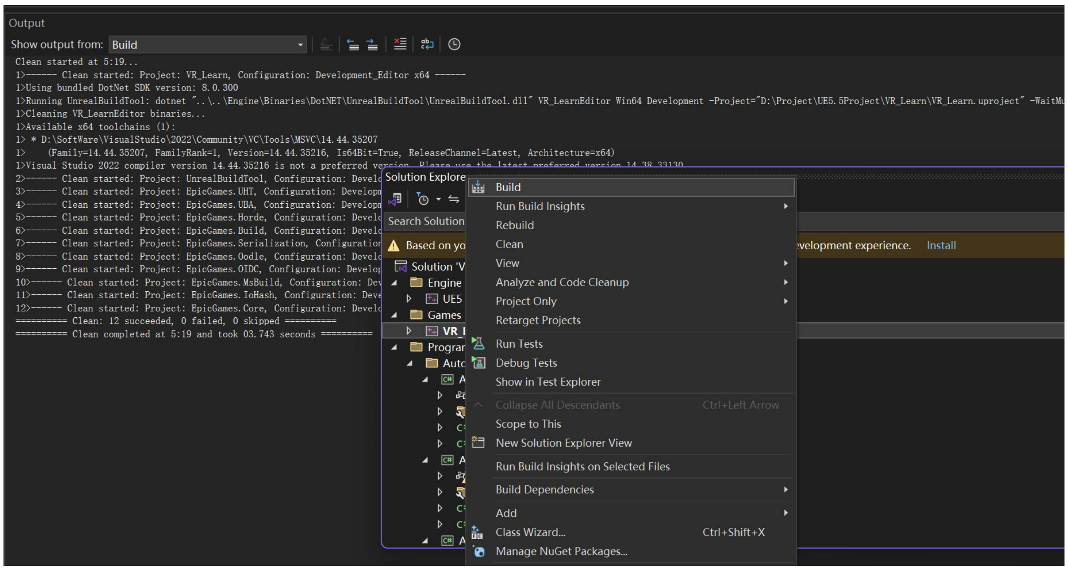
Mesh optimization: Reduces voxel staircase effect through Gaussian smoothing, removes duplicate faces and unreferenced vertices.

Physical space mapping: According to MRI image header files, converts mesh voxel coordinates to physical coordinates, ensuring 3D model size is consistent with real organs.

Diagnostic report generation: Generates text reports in the format of volume and percentage quantified indicators.

4.3.6 UE Plugin Usage Workflow

Compile plugin in VisualStudio and enter UE:



```
Clean started at 5:19...
1>------ Clean started: Project: VR_Learn, Configuration: Development_Editor x64 ------
1>Using bundled DotNet SDK version: 8.0.300
1>Running UnrealBuildTool: dotnet "..\..\Engine\Binaries\DotNET\UnrealBuildTool\UnrealBuildTool.dll" VR_LearnEditor Win64 Development -Project="D:\Project\UE5.5Project\VR_Learn\VR_Learn.uproject" -WaitMu
1>Cleaning VR_LearnEditor binaries...
1>Available x64 toolchains (1):
1> * D:\SoftWare\VisualStudio\2022\Community\VC\Tools\MSVC\14.44.35207
1>    (Family=14.44.35207, FamilyRank=1, Version=14.44.35216, Is64Bit=True, ReleaseChannel=Latest, Architecture=x64)
1>Visual Studio 2022 compiler version 14.44.35216 is not a preferred version. Please use the latest preferred version 14.38.33130
2>------ Clean started: Project: UnrealBuildTool, Configuration: Develc
3>------ Clean started: Project: EpicGames.UHT, Configuration: Developn
4>------ Clean started: Project: EpicGames.UBA, Configuration: Developn
5>------ Clean started: Project: EpicGames.Horde, Configuration: Develc
6>------ Clean started: Project: EpicGames.Build, Configuration: Develc
7>------ Clean started: Project: EpicGames.Serialization, Configuratio
8>------ Clean started: Project: EpicGames.Oodle, Configuration: Develc
9>------ Clean started: Project: EpicGames.OIDC, Configuration: Develop
10>------ Clean started: Project: EpicGames.MsBuild, Configuration: Dev
11>------ Clean started: Project: EpicGames.IoHash, Configuration: Deve
12>------ Clean started: Project: EpicGames.Core, Configuration: Develc
========== Clean: 12 succeeded, 0 failed, 0 skipped ==========
========== Clean completed at 5:19 and took 03.743 seconds ==========
```

Figure 19 VisualStudio Compile Plugin to Enter UE

Ensure plugin is checked:

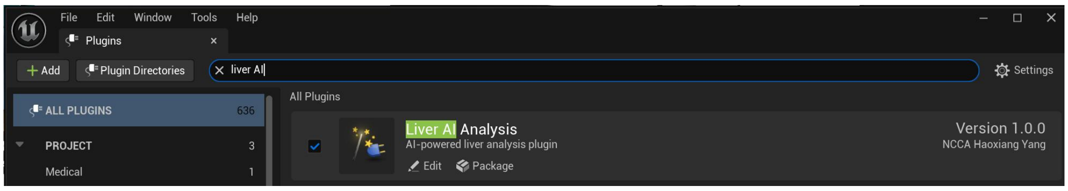


Figure 20 Plugin Checkbox

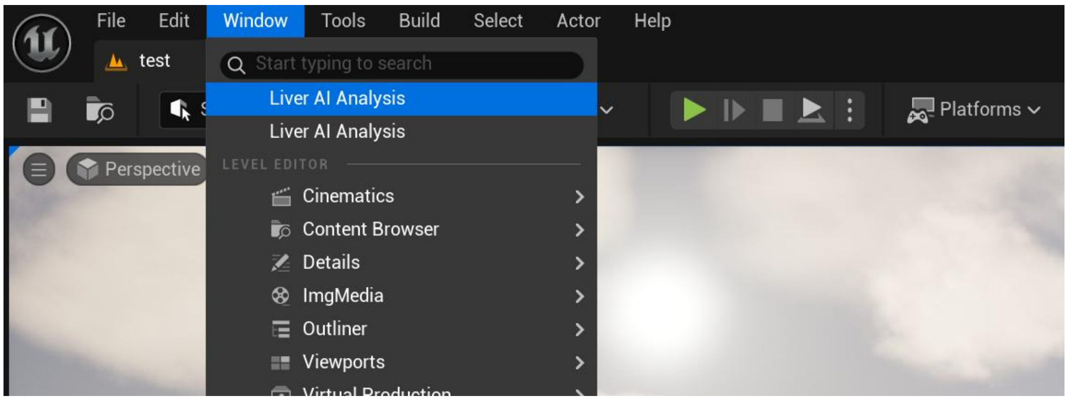
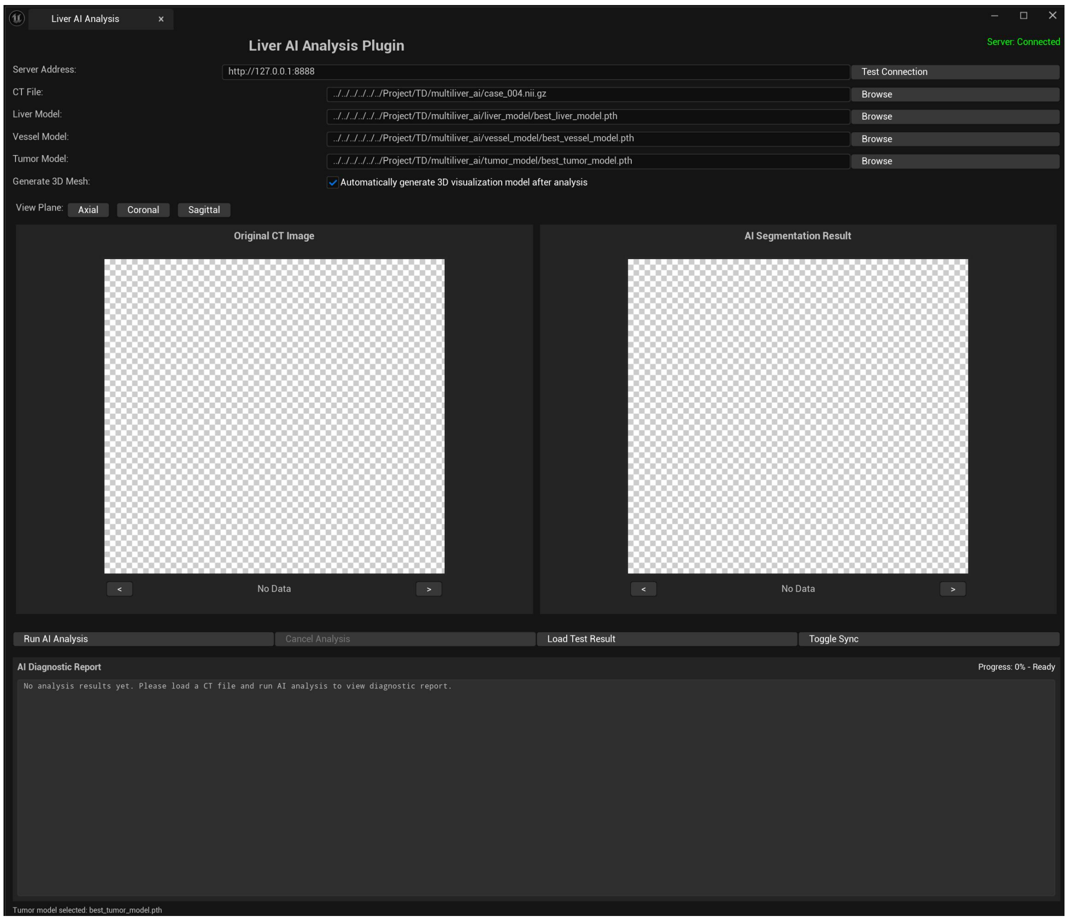


Figure 21 Plugin Location

Open plugin interface and connect to server:



Server Address: http://127.0.0.1:8888

CT File: ../../../../../Project/TD/multiliver_ai/case_004.nii.gz

Liver Model: ../../../../../Project/TD/multiliver_ai/liver_model/best_liver_model.pth

Vessel Model: ../../../../../Project/TD/multiliver_ai/vessel_model/best_vessel_model.pth

Tumor Model: ../../../../../Project/TD/multiliver_ai/tumor_model/best_tumor_model.pth

Generate 3D Mesh: ✓ Automatically generate 3D visualization model after analysis

View Plane:

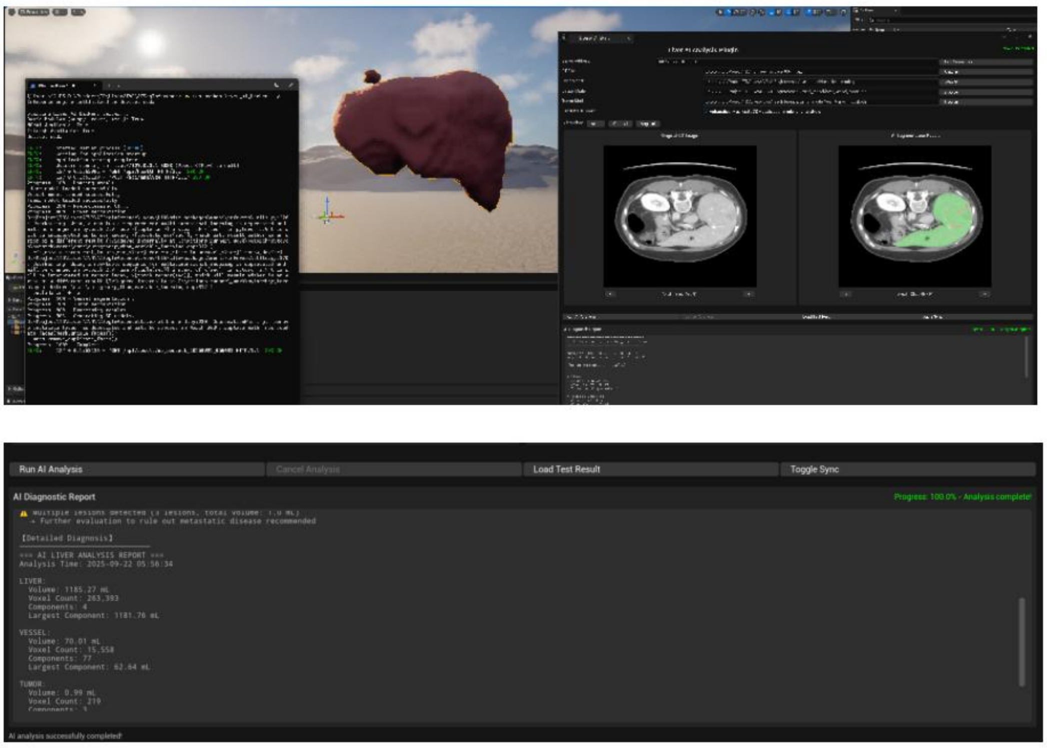
AI Diagnostic Report

Progress: 0% - Ready

```
No analysis results yet. Please load a CT file and run AI analysis to view diagnostic report.
```

Tumor model selected: best_tumor_model.pth

Figure 22 Plugin Open Interface, Connected to Server



AI Diagnostic Report

Progress: 100.0% - Analysis complete!

```
⚠ Multiple lesions detected (3 lesions, total volume: 1.0 mL)
   → Further evaluation to rule out metastatic disease recommended

【Detailed Diagnosis】

=== AI LIVER ANALYSIS REPORT ===
Analysis Time: 2025-09-22 05:56:34

LIVER:
  Volume: 1185.27 mL
  Voxel Count: 263,393
  Components: 4
  Largest Component: 1181.76 mL

VESSEL:
  Volume: 70.01 mL
  Voxel Count: 15,558
  Components: 77
  Largest Component: 62.64 mL

TUMOR:
  Volume: 0.99 mL
  Voxel Count: 219
  Components: 3
```

AI analysis successfully completed!

Figure 23 UE Plugin Running Inference Results

4.4 VR Application

Preliminary work on VR applications has been carried out. Loading and displaying liver 3D models in UE, and displaying liver and blood vessel models in VR:

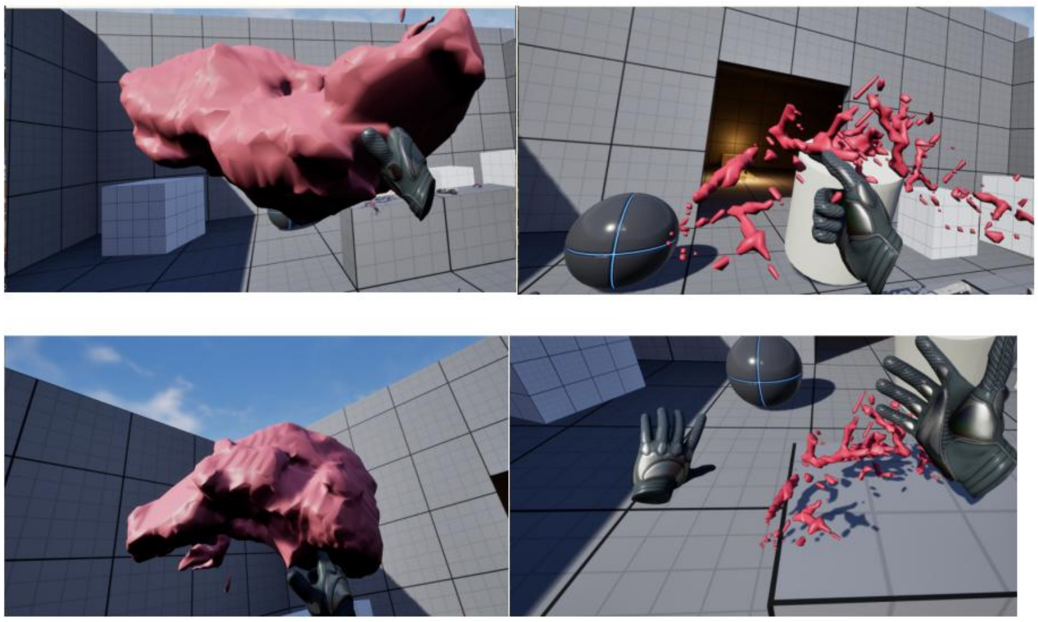


Figure 24 VR Application

This project has developed a VR menu, as shown in Figure 25. In VR, operations such as grasping and rotating can be performed on the liver and blood vessels, as shown in Figures 26, 27, and 28. This lays the foundation for further VR applications.

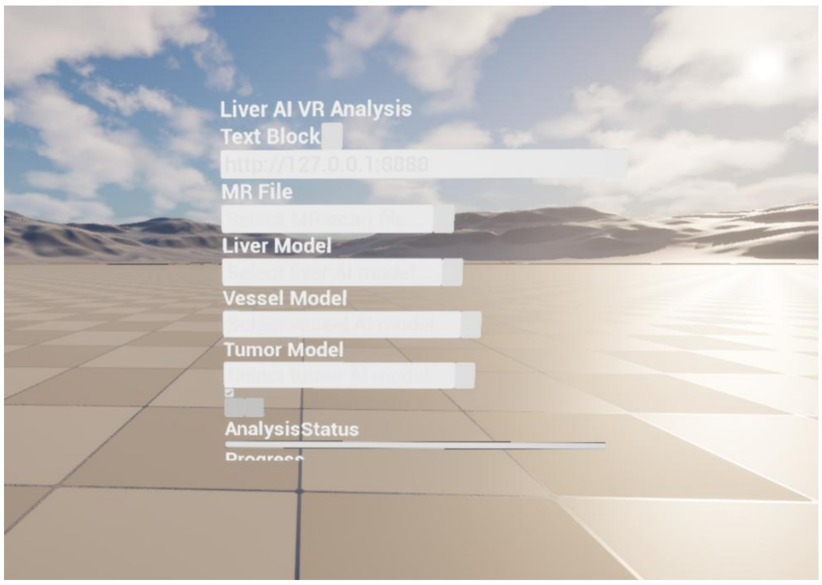


Figure 25 VR Menu

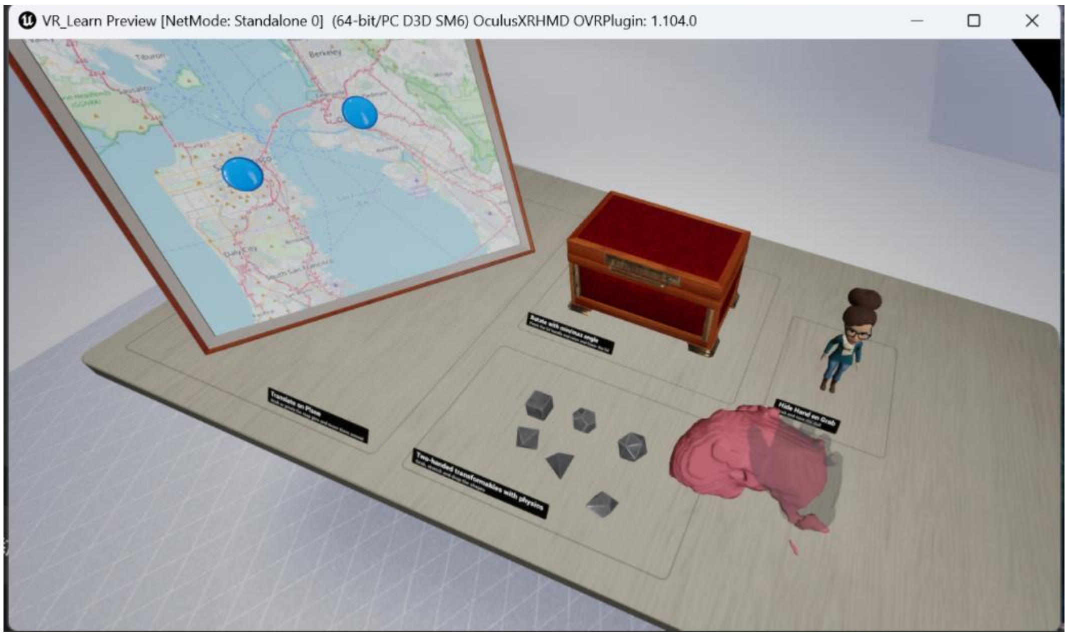

VR_Learn Preview [NetMode: Standalone 0] (64-bit/PC D3D SM6) OculusXRHMD OVRPlugin: 1.104.0
Translate on Plane
Two-handed transformables with physics
Hide Hand on Grab

Figure 26 VR Liver Grasping

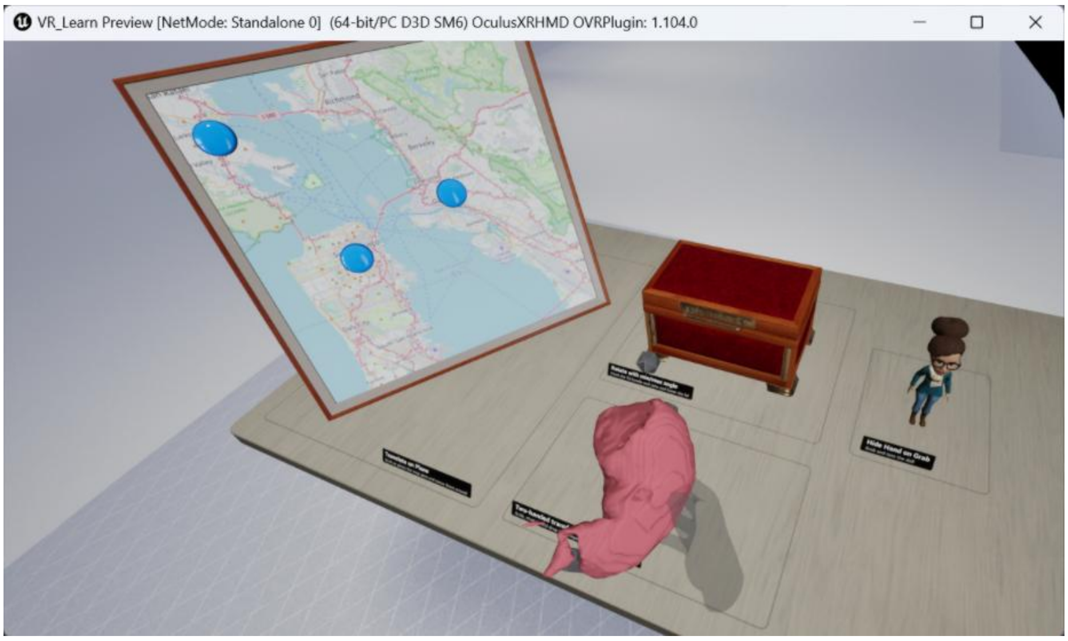

VR_Learn Preview [NetMode: Standalone 0] (64-bit/PC D3D SM6) OculusXRHMD OVRPlugin: 1.104.0
Translate on Plane
Hide Hand on Grab

Figure 27 VR Liver Rotation

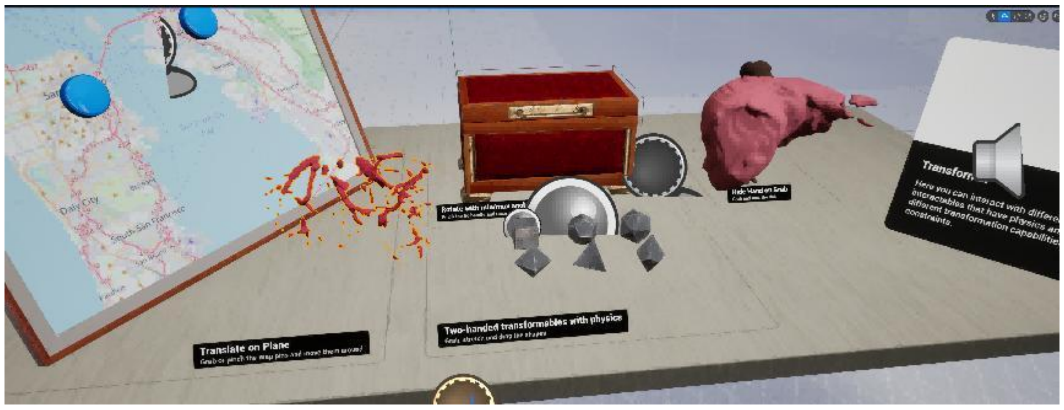

Translate on Plane
Two-handed transformables with physics
Hide Hand on Grab
Here you can interact with different interactables that have physics and different transformation capabilities constraints.

Figure 28 VR Liver and Blood Vessels

5 Conclusion

This project adopts a client-server architecture, achieving automatic segmentation of MRI images and 3D visualization. The frontend uses Unreal Engine to provide an interactive interface and real-time rendering, while the backend is based on FastAPI and PyTorch for AI inference. Through RESTful API for cross-language communication, it supports multi-organ segmentation of liver, blood vessels, and tumors, and automatically generates 3D reconstruction models. The developed Unreal Engine plugin integrates liver MRI image AI automatic segmentation and 3D reconstruction, laying the foundation for VR applications. The plugin can process liver MRI data in .nii.gz format, implement AI automatic segmentation based on MONAI's SegResNet deep learning, perform 3D reconstruction and result analysis based on the Marching Cubes algorithm. The plugin integrating AI segmentation and 3D reconstruction ensures the precision of AI segmentation while achieving intuitive 3D visualization through UE, and provides quantified indicators and reports. Results show that the system achieved good performance in liver, blood vessel, and tumor segmentation tasks, with high-quality 3D models. The project has achieved its expected goals. Future work can develop VR visualization tools based on this foundation, providing users with immersive observation and operation on VR devices.

5.1 Advantages and Innovations of the Plugin

The Unreal Engine plugin integrating liver MRI image AI automatic segmentation and 3D reconstruction for VR developed in this project has the following main advantages and innovations:

(1) MRI image AI automatic segmentation based on SegResNet deep learning network framework improves segmentation performance and system robustness.

(2) The system frontend uses Unreal Engine to provide an interactive interface and real-time rendering, while the backend is based on FastAPI and PyTorch for AI inference.

(3) VR visualization tools can be further developed based on the plugin.

5.2 Future Work

Based on the current plugin development achievements and experience, future work includes:

(1) Develop VR visualization and interaction tools that can run on VR devices, allowing users to wear VR devices, observe the liver and its internal structures in a virtual environment, and interact naturally with the models, such as rotating, scaling, and cutting models.

(2) Expand to multiple organs: Add segmentation models for organs such as kidneys, lungs, and pancreas in the "AI Inference Layer", enabling the plugin to support multi-site medical image analysis.

6 References

[1] Alom M Z, et al. Automatic Slice Growing Method Based 3D Reconstruction of Liver with its Vessels[C]. ALOM, 2014.

[2] Li S, Tso G K F, He K J. Bottleneck feature supervised U-Net for pixel-wise liver and tumor segmentation[J]. Expert Systems with Applications, 2020, 145: 113131.

Kim J, et al. Automated 3D liver segmentation from hepatobiliary phase MRI for enhanced preoperative planning[J]. PubMed, 2023.

[3] HSU C-Y, CHANG C, CHEN T W, et al. Brain Tumor Segmentation (BraTS) Challenge Short Paper: Improving Three-Dimensional Brain Tumor Segmentation Using SegResnet and Hybrid Boundary-Dice Loss[J]. 2022: 334-344.

[4] ROSS T, TANNA R, LILAONITKUL W, et al. Deep Learning for Automated Image Segmentation of the Middle Ear: A Scoping Review[J]. Otolaryngology, 2024, 170(6): 1544-1554.

[5] LORENSEN W E, CLINE H E. Marching cubes: A high resolution 3D surface construction algorithm[J]. ACM SIGGRAPH Computer Graphics, 1987: 163-169.

[6] WANG M, LUO H, CUI Q. Three-dimensional reconstruction based on improved marching cubes algorithm[J]. Journal of Mechanics in Medicine and Biology, 2020, 20(09): 2040002.