RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONAL POLYTECHNIQUE

Département D'Electronique

Final Year Project Thesis

Submitted in partial fulfillment of the requirements
for the degree of State Engineer in Electronics

Automated 3D Liver Segmentation and Mixed Reality Integration for Preoperative Surgical Planning

---

TOUIL Mohamed Reda & BENZINE Yasser

Supervised by **Prof. Mourad ADNANE**, NHSAST

Co-Supervisor by **Dr.Mohamed Rafik Ait-arab**

Presented and publicly defended on (21/06/2025)

Jury Members:

President:	Pr. Bousbia-Salah Hichem	ENP
Examiner:	Dr . Lani Fatiha	ENP
Jury:	Pr . BOUSBAI M'hamed	ENP

ENP 2025

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONAL POLYTECHNIQUE

Département D'Electronique

Final Year Project Thesis

Submitted in partial fulfillment of the requirements
for the degree of State Engineer in Electronics

Automated 3D Liver Segmentation and Mixed Reality Integration for Preoperative Surgical Planning

---

TOUIL Mohamed Reda & BENZINE Yasser

Supervised by **Prof. Mourad ADNANE**, NHSAST

Co-Supervisor by **Dr.Mohamed Rafik Ait-arab**

Presented and publicly defended on (21/06/2025)

Jury Members:

President:	Pr. Bousbia-Salah Hichem	ENP
Examiner:	Dr . Lani Fatiha	ENP
Jury:	Pr . BOUSBAI M'hamed	ENP

ENP 2025

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

École Nationale Polytechnique (ENP)

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Électronique

Mémoire de Projet de Fin d'Études

Présenté en vue de l'obtention du diplôme
d'Ingénieur d'État en Électronique

Segmentation hépatique 3D automatisée et intégration en Réalité Mixte pour la planification chirurgicale préopératoire

---

TOUIL Mohamed Reda & BENZINE Yasser

Encadré par **Pr. Mourad ADNANE**, NHSAST

Co-encadré par **Dr. Mohamed Rafik Aït-Arab**

Présenté et soutenu publiquement le 21 juin 2025

Membres du jury :

Président :	Pr. Bousbia-Salah Hichem	ENP
Examinateur :	Dr. Lani Fatiha	ENP
Membre :	Pr. Bousbaï M'hamed	ENP

ENP 2025

ملخص :

تعتمد الخطة الجراحية التقليدية للكبد على التفسير اليدوي للصور ثنائية الأبعاد وهو ما يحدّ من دقتها وكفاءتها. يعرض هذا التقرير نظامًا متكاملاً يهدف إلى إحداث ثورة في هذه الممارسة من خلال الجمع بين التقسيم الآلي للكبد باستخدام التعلم العميق وبيئة الواقع المختلط التعاونية.

يعتمد النهج المطوّر على شبكات عصبية متقدمة لتحقيق تقسيم دقيق للكبد والأورام، متبوعًا بإعادة بناء سريعة ثلاثية الأبعاد. تم تدريب النموذج على مجموعة بيانات IRCAD، و LiTS ثلاثي الأبعاد يبلغ 0.92، بنتائج مماثلة على مجموعتي Dice وحقق متوسطًا لـ MDHV.

تم تصدير النماذج ثلاثية الأبعاد الناتجة إلى تطبيق تفاعلي في بيئة الواقع المختلط، مما يتيح تصورًا غامرًا وتفاعلاً بديهيًا. كما يدعم النظام مشاركة متعددة المستخدمين في الوضع المحلي التعاوني، مما يسهل المناقشة والتخطيط المشترك.

هذه المساهمة الفريدة التي تجمع بين التقسيم الآلي والتعاون الفعّال في الواقع المختلط تعزز بشكل كبير من دقة وكفاءة التخطيط الجراحي، وتفتح آفاقًا واعدة لتحسين الرعاية السريرية من خلال التكامل بين الذكاء الاصطناعي والواقع المختلط.

الكلمات المفتاحية: التخطيط الجراحي للكبد، التقسيم الآلي، التعلم العميق، الواقع المختلط، إعادة البناء ثلاثي الأبعاد، التعاون الطبي.

Résumé

La planification chirurgicale hépatique classique repose sur l'interprétation manuelle d'images 2D, ce qui limite sa précision et son efficacité. Ce projet propose un système intégré combinant segmentation automatique par deep learning et visualisation immersive en réalité mixte (MR). Le modèle atteint un Dice médian de 0,92 sur le dataset IRCAD, avec des résultats comparables sur LiTS et MDHV.

Les modèles 3D générés sont visualisables en MR via une application interactive multi-utilisateur. Ce système novateur améliore considérablement la précision de la planification et ouvre la voie à une meilleure prise en charge clinique grâce à la synergie entre intelligence artificielle et réalité mixte.

Mots-clés : Planification chirurgicale hépatique, segmentation automatique, deep learning, réalité mixte, reconstruction 3D, collaboration médicale.

Abstract

Traditional liver surgical planning, relying on manual interpretation of 2D images, is often limited in precision and efficiency. This report introduces an integrated system designed to revolutionize this practice by combining automated deep learning-based liver segmentation with a collaborative Mixed Reality (MR) environment.

The developed approach leverages advanced neural network architectures for accurate liver and tumor segmentation, followed by rapid 3D reconstruction. The segmentation model achieved a median Dice score of 0.92 on the IRCAD dataset, with comparable performance on LiTS and MDHV. The generated 3D models are then imported into an interactive MR application, enabling immersive visualization and intuitive manipulation.

Furthermore, the system supports multiple simultaneous users in local collaborative mode, facilitating joint discussion and planning. This unique contribution, merging automated segmentation with immersive MR collaboration, significantly enhances the precision and efficiency of surgical planning, offering substantial potential for improving clinical outcomes. The emphasis on these key figures and the system's unique contribution highlights that the project's value lies not only in the performance of its individual components but also in the synergy created by integrating AI and MR to optimize a complex clinical workflow.

Keywords: Liver surgical planning, automated segmentation, deep learning, Mixed Reality, 3D reconstruction, clinical collaboration.

Acknowledgements

First and foremost, I thank God, the Almighty, for giving me the strength, determination, and courage to accomplish this humble thesis.

I am deeply grateful to my beloved parents, without whom I could never have come this far. I also extend my heartfelt thanks to my entire family and my friends for their continuous support and encouragement throughout this journey.

I would like to express my sincere gratitude to my supervisors, Professor Mourad Adnan and Dr. Mohamed Rafik Ait-Arab, for giving me the opportunity to work on such a relevant and impactful topic. I especially thank them for their guidance, availability, and constant support throughout this project.

I also wish to thank my friend from the Electronics Department for his valuable assistance, as well as all those who contributed to the success of this work in any way — whether through technical help, advice, or encouragement. Your support has been truly appreciated.

BENZINE Yasser

Acknowledgements

الحمدُ للهِ الذي أنشأ الخَلْقَ وبَرأه، وخلقَ الماءَ والثَّرَى، وأبدعَ كلَّ شيءٍ وذرَا، الرَّحمنُ على العرشِ استوى، والصلاةُ والسلامُ على سيِّدِنا محمدٍ المبعوثِ في أمِّ القرى، الذي بكى على أمتِه، صلاةً وسلامًا دائمَيْنِ إلى يومِ الدين، وعلى آلِهِ وأصحابِه ومَن تبعهم بإحسانٍ إلى يوم الدين.

أمَّا بَعْدُ، فإننا طلبنا العلمَ ابتغاءَ وجهِ اللهِ تعالى، وما أردنا بذلك إلا رضاه، وذلك فضلُ اللهِ يُؤتيه من يشاء، وقد قال تعالى:
﴿ يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ﴾ (سورة المجادلة: ١١).

فنسأل الله أن يرفع قدرَنا في الآخرة كما رفعه في الدنيا، وأن يجعل علمَنا حُجّةً لنا لا علينا، وأن ينفعنا بما علّمَنا ويزيدنا علمًا وهدًى وتوفيقًا، وأن يجعله خالصًا لوجهه الكريم، نافِعًا لعباده. ونعوذ بالله أن يكون عملُنا هذا رياءً أو سمعةً، بل نسألُه الإخلاصَ في القول والعمل، والتوفيقَ والسدادَ فيما قدَّمنا وما سنُقَدِّم.
وما كان في هذا العمل من صوابٍ فمن الله وحده، وما كان فيه من خطأ أو تقصيرٍ فمني ومن الشيطان، وأسأل الله العفوَ والمغفرة، وألتمس منكم العذرَ والدعاء.

أهدي هذا العمل المتواضع إلى والديَّ العزيزين، اللذين كانا بعد الله تعالى السبب في وجودي، ونبراس حياتي، وملاذي في الشدة والرخاء.

إلى أمي الحنونة، صاحبة القلب الكبير، التي غرست في نفسي القيم والمبادئ، واحتضنتني بدعواتها الصادقة، فكانت السند والداعم الأول في مسيرتي **وإلى أبي الكريم**، الذي كان مثالًا في الكفاح والصبر، وغمرني بحكمته وتوجيهه، وعلمني معنى الجد والاجتهاد. فلهما مني كل الحب والوفاء. أسأل الله أن يجزيهما عني خير الجزاء، وأن يبارك في عمرهما ويرزقني برَّهما في الدنيا والآخرة

كما أرفع أسمى آيات الشكر والعرفان إلى كل من علمني حرفًا أو قدم لي نصيحة صادقة، وإلى أساتذتي الكرام الذين كانوا نعم القدوة والدليل في طريق العلم والمعرفة.

ولا أنسى زملائي الذين شاركوني هذه الرحلة العلمية، فكانوا خير رفقة ودعم في أوقات الجد والتعب، جزاهم الله عني كل خير.

ولا يفوتني أن أخص بالشكر دكاترة مختبر LDCCP الذين شاركتهم العمل طيلة ستة أشهر، فلم يكن بيننا مجرد مكان يجمعنا، بل تقاسمنا لحظات من التعاون والتشجيع والإنسانية الصادقة. لقد ترك كل واحد منهم أثرًا جميلًا في نفسي وذاكرتي، فجزاهم الله عني خير الجزاء، وبارك في علمهم وعطائهم.

فلكم جميعًا أصدق عبارات الامتنان والدعاء، وأسأل الله أن يجمعنا دائمًا على الخير والمحبة، وأن يكتب لنا التوفيق والسداد في الدارين والسلام عليكم ورحمة الله تعالى وبركاته.

طويل محمد رضا

TOUIL Mohamed Reda

Contents

List of Abbreviations

- CT: Computed Tomography
- MR: Mixed Reality
- VR: Virtual Reality
- AR: Augmented Reality
- XR: Extended Reality
- CNN: Convolutional Neural Network
- IoU: Intersection over Union
- MRI: Magnetic Resonance Imaging
- SDK: Software Development Kit
- APK: Application
- ADB: Android Debug Bridge
- IRB: Institutional Review Board
- IRCAD: Institut de Recherche contre les Cancers de l'Appareil Digestif
- Att: attention
- Res: residual
- LiTS: Liver Tumor Segmentation
- MDHV: Medical Decathlon Hepatic Vessel
- HMD: Head-Mounted Display
- GPU: Graphics Processing Unit
- LIME: Local Interpretable Model-Agnostic Explanations
- ROI: Region of Interest
- 2D: Two-Dimensional
- 3D: Three-Dimensional

List of Figures

List of Tables

Introduction

Background and Motivation

Liver surgery is one of the most challenging procedures in abdominal surgery due to the organ's complex vasculature and segmental anatomy. Accurate localization of tumors and vascular bifurcations is critical to choose the optimal surgical approach that maximizes the chances of a successful outcome and to avoid complications [34].

Advancements in artificial intelligence (AI), particularly deep learning for medical image segmentation, combined with mixed reality (MR) visualization technologies, present a unique opportunity to enhance surgical planning. This project leverages these innovations to provide a more accurate, immersive, and collaborative planning solution for liver resection procedures.

Problem Statement

Liver surgery requires precise spatial understanding of complex intrahepatic structures. However, the current clinical workflow for surgical planning relies primarily on two-dimensional (2D) imaging modalities such as computed tomography (CT) and magnetic resonance imaging (MRI) [38]. While these techniques offer high-resolution anatomical information, they suffer from several fundamental limitations:

- Lack of depth perception and spatial orientation.
- Overlapping structures that obscure anatomical boundaries.
- Difficulty distinguishing tissues with similar radiodensity, particularly in hepatic vasculature.
- Interpretation is dependent on radiologists, introducing inter-observer variability.

These limitations directly impact a surgeon's ability to accurately assess tumor margins, vascular proximity, and resectability, which can compromise surgical outcomes.

Current Surgical Planning Workflow

The conventional preoperative workflow typically follows these steps:

1. **Imaging Acquisition:** CT or MRI scans are acquired to visualize the liver, lesions, and surrounding structures.
2. **Radiological Analysis:** A radiologist interprets the scans and provides annotated reports of tumors, vessels, and anomalies.
3. **Surgical Planning:** The surgeon mentally constructs a three-dimensional (3D) model based on 2D images and radiological descriptions.
4. **Mental Visualization:** The surgeon memorizes spatial relationships between key anatomical features and uses this mental model intraoperatively.

This process places a high cognitive burden on surgeons, relying heavily on experience and visualization skills. It also introduces the risk of subjective interpretation and inconsistent surgical decisions.

Mental Reconstruction Challenges and Risks

The reliance on mental reconstruction for spatial reasoning poses several challenges:

- **Cognitive Load:** Memorizing detailed anatomical structures in 3D space is mentally taxing.
- **Subjectivity:** Surgeons' spatial reasoning abilities vary, which may lead to inconsistent planning.
- **Potential for Error:** Misinterpretation of spatial relationships can result in incomplete tumor resection or accidental vascular damage.

These issues collectively underscore the need for improved tools that support intuitive, accurate, and reproducible surgical planning. Combining automated segmentation via deep learning with immersive 3D visualization in mixed reality environments offers a promising solution to overcome these limitations.

Objectives

The primary objective of this project is to develop an end-to-end system that assists surgeons in liver resection planning through intelligent image analysis and intuitive visualization. Specifically, the system aims to:

- Automate the segmentation of liver parenchyma, hepatic vessels, and tumors from 3D medical images using deep learning.
- Generate 3D mesh models from segmented volumes for immersive visualization.
- Develop a mixed reality application for the Meta Quest 3 headset to interact with the anatomical models in real-time.
- Enable local multi-user collaboration for surgical discussion and planning.

Structure of the Report

This report is structured as follows:

- Chapter 1: Presents the anatomical and clinical background of liver surgery and tumor management.
- Chapter 2: Reviews the state of the art in deep learning-based liver segmentation and mixed reality in surgical applications.
- Chapter 3: Describes the methodology for segmentation, reconstruction, and model conversion.
- Chapter 4: Details the development of the mixed reality (MR) application for Meta Quest 3 headset.

Chapter 1

Medical and Clinical Background

1.1 Liver Anatomy and Segmentation Relevance

A comprehensive understanding of internal liver anatomy is critical in liver disease diagnosis and treatment. Claude Couinaud, a renowned French surgeon and anatomist, published his work on segmental liver anatomy in 1957 [6]. He emphasized that the functional anatomy of the liver is primarily governed by its vascular and biliary systems, a foundation that modern hepatic surgery still relies upon today.

1.1.1 Organ

Figure 1.1 shows the general liver anatomy on (A) a physical model and (B) a CT slice. The liver is the largest internal organ, roughly 2–3% of total body weight. Morphologically, it is divided into two lobes, but functionally it is segmented based on vasculature rather than surface anatomy [21].

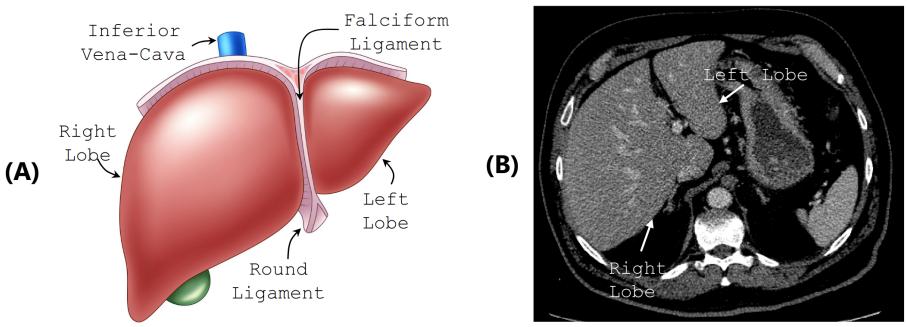


Figure 1.1: General liver anatomy on (A) physical model and (B) CT slice.

1.1.2 Vasculature

The liver receives about 25% of the cardiac output, via two main inflows: the hepatic artery (25%) and the portal vein (75%) [27]. These blood sources mix within the hepatic sinusoids before draining through hepatic veins into the inferior vena cava (IVC).

Venous System Couinaud's classification [6] defines liver segments based on portal and hepatic vein bifurcations. The portal vein divides into right and left branches at

the hilum, creating functional subdivisions. Each subdivision is further segmented into sectors and segments (I–VIII) as shown in Figure 1.2.

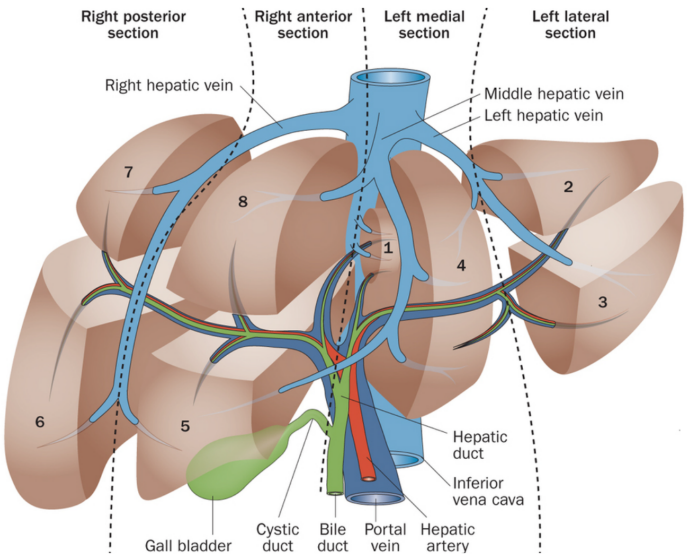


Figure 1.2: Couinaud classification of liver segments based on vascular anatomy.

Arterial and Biliary Systems The hepatic artery typically branches from the celiac axis, forming the proper hepatic artery which runs alongside the portal vein and bile ducts in the Glissonean pedicle [27]. The biliary tree closely follows portal vein branching and is essential for bile secretion.

1.1.3 Function

The liver performs numerous essential functions including metabolism regulation, bile production, detoxification, blood clotting factor synthesis, hormone regulation, and storage of vitamins and glucose [11]. Albumin production and bilirubin elimination are especially critical in hepatic pathology evaluation.

1.2 Importance of Preoperative Visualization

1. Why imaging is important

Imaging is essential for diagnosing liver diseases and planning surgery because it is **non-invasive**. It helps doctors clearly **locate tumors inside the liver**.

2. Common types of imaging

- **CT Scan (Computed Tomography)** Computed Tomography (CT), commonly referred to as a CT scan, is a medical imaging method that uses X-rays taken from multiple angles and computer processing to generate cross-sectional images of the body. It is especially effective for examining bones, detecting internal injuries, and evaluating conditions affecting the chest, abdomen, and pelvis.

- **MRI (Magnetic Resonance Imaging)** Magnetic Resonance Imaging (MRI) is a non-invasive medical imaging technique that uses strong magnetic fields and radiofrequency waves to produce high-resolution images of soft tissues in the human body. Unlike CT, MRI does not use ionizing radiation. It is particularly useful for visualizing the brain, spinal cord, joints, muscles, and internal organs such as the liver or kidneys.

3. What imaging tells us

- **Liver anatomy and blood vessels:** Imaging shows the shape of the liver and the position of its main vessels (portal vein, hepatic artery, hepatic veins), which is important to see how close tumors are to these structures.
- **Tumor details:** Imaging helps locate tumors, count them, measure their size, and analyze their blood flow. Malignant tumors usually have specific contrast patterns (bright in arterial phase, then fading in later phases).
- **Other liver problems:** It can also show signs of fibrosis or cirrhosis, such as irregular liver shape, enlarged or shrunken liver parts, or uneven liver texture.
- **Signs of portal hypertension:** Doctors look for an enlarged portal vein (over 12 mm), a big spleen (more than 381 cc), or unusual blood vessels (shunts), which can affect whether surgery is safe.

1.3 Current Challenges in 2D-based Planning

While two-dimensional (2D) imaging modalities like CT and MRI remain the foundation for diagnosing and assessing liver disease, relying solely on 2D views for complex surgical planning presents significant limitations and challenges [21].

The primary challenge lies in limited spatial understanding. Standard 2D displays (axial, coronal, sagittal slices) require surgeons to mentally integrate these separate views to build a three-dimensional understanding of the liver's intricate internal anatomy, including the complex branching patterns of the portal veins, hepatic arteries, hepatic veins, and bile ducts, and their precise spatial relationship to tumors [38]. This mental reconstruction process is inherently difficult, places a high cognitive load on the surgeon, is heavily dependent on individual experience and spatial reasoning skills, and is prone to error or misinterpretation, especially when dealing with anatomical variations or complex tumor configurations [38].

The lack of true depth perception in 2D images makes it challenging to accurately judge the proximity of a lesion to critical vascular structures or to precisely define the 3D trajectory of a resection plane relative to segmental boundaries [29]. Structures can be obscured by others in certain 2D projections, hindering a complete assessment [23]. This limited spatial comprehension can directly impact the accuracy and confidence of surgical planning, potentially leading to suboptimal resection strategies, difficulty in achieving clear margins, or increased risk of inadvertent vascular injury during the procedure [14].

Comparative studies consistently show that surgeons using 3D visualization tools achieve significantly higher accuracy in localizing tumors and defining appropriate resection plans compared to those relying on 2D images alone [38].

Chapter 2

State of The Art on Medical Image Segmentation with Deep Learning

2.1 Medical Image Segmentation

2.1.1 Definition and Importance

Image segmentation is the process during which the pixels/voxels[1] shaping a particular region of interest (ROI) are identified. Image segmentation continues to be one of the hottest research topics in the field of computer vision. In medical imaging, the role of segmentation extends beyond anatomical structure delineation to include ROI localization (e.g., tumor position in the liver). With technological advancement in image acquisition and the ever-increasing number of acquired images, medical image segmentation has gained significant popularity in assisting medical doctors in clinical diagnosis and treatment planning. Therefore, accurate, reliable, and repeatable segmentation algorithms can play an important role in the fast automation process of analyzing a large number of cases, and the improvement of the overall standard routine of care.

This chapter introduces detailed literature review of the different deep learning models proposed for the liver, tumor, and vessel segmentation.

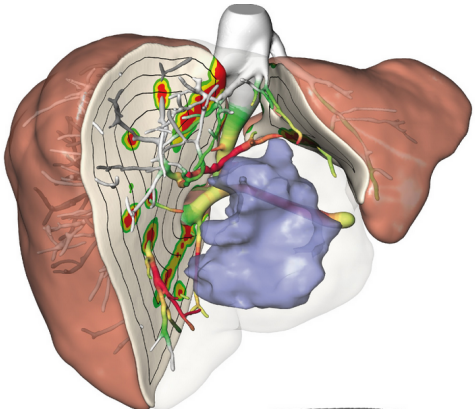


Figure 2.1: 3D anatomical of the liver with segmented tumor (purple), vascular structures (red, green, yellow), and parenchymal regions.

[1]Pixel is the smallest 2D constituent of an image. Voxel is the smallest volume element in a 3D image.

2.2 Liver and Tumor Segmentation

Liver tumor segmentation is a challenging task to accomplish for various reasons such as: the tumor size and position can significantly vary between different patients. The following is a detailed review of the recent (2015–2024) deep learning models that perform liver tumor segmentation.

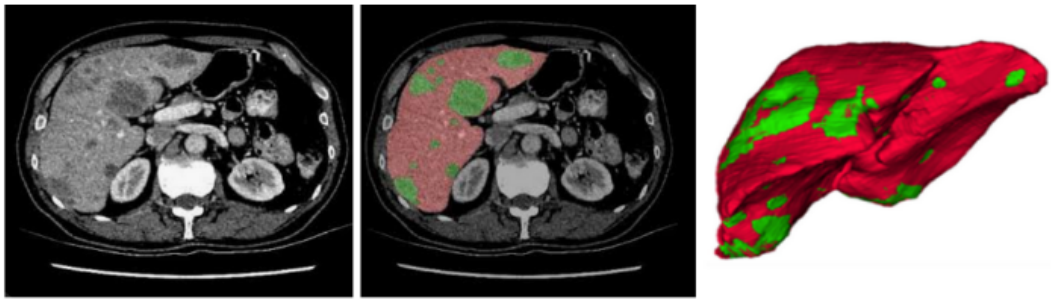


Figure 2.2: Multimodal visualization of liver and tumor segmentation. (a, b) Axial CT slices and (c) 3D reconstructed volume.

2.2.1 Literature Review

The evolution of deep learning approaches for liver tumor segmentation can be categorized into three primary methodological frameworks: cascaded architectures, end-to-end integrated models, and hybrid attention-based networks. Each approach addresses the fundamental challenge of accurately delineating tumor boundaries while managing the inherent complexity of medical imaging data.

Cascaded architectures emerged as the foundational approach for liver tumor segmentation, establishing a two-stage process that first segments the liver region before focusing on tumor detection. Christ et al. introduced this paradigm using fully connected CNNs trained on the IRCAD dataset, achieving 94% Dice coefficient for liver segmentation and 82% for tumor segmentation through the application of 3D conditional random fields for post-processing refinement. This methodology demonstrated that hierarchical processing could effectively manage the multi-scale nature of the segmentation task. Yuan et al. extended this concept with hierarchical convolution-deconvolution networks on the LiTS dataset, implementing a three-stage approach where successive networks progressively refine segmentation quality, ultimately achieving 65.7% Dice coefficient for tumor segmentation.

End-to-end integrated approaches subsequently emerged to address the limitations of cascaded methods, particularly the error propagation inherent in sequential processing. Vorontsov et al. proposed simultaneous training of liver and tumor segmentation networks, where both models receive identical input while sharing latent representations from the final convolutional layers. This approach achieved 95% Dice coefficient for liver segmentation and 66% for tumor segmentation on the LiTS dataset. The integration of contextual information through shared representations demonstrated superior performance compared to traditional cascaded methods while maintaining computational efficiency.

Recent developments have focused on attention mechanisms and hybrid architectures to enhance feature extraction and spatial awareness. Li et al. introduced the

H-DenseUNet, combining 2D DenseUNet for intra-slice feature extraction with 3D components for inter-slice processing, achieving 96% liver segmentation and 72.2% tumor segmentation accuracy. This hybrid approach addresses the volumetric nature of medical imaging while maintaining computational tractability. Advanced attention mechanisms have been implemented by Jiang et al., incorporating both soft and hard attention blocks within fully convolutional networks, resulting in 96% liver segmentation and 67% tumor segmentation performance on the IRCAD dataset.

The relationship between tumor size and segmentation accuracy represents a critical performance factor across all methodological approaches. Vorontsov et al. demonstrated this challenge through evaluation on liver metastases from colorectal cancer, achieving 14%, 53%, and 68% Dice coefficients for tumors smaller than 10mm, between 10-20mm, and larger than 20mm, respectively. This size-dependent performance limitation remains consistent across different architectural approaches and represents a fundamental challenge for clinical application.

Contemporary research has achieved notable performance improvements through advanced architectural innovations. Li et al. developed the Eres-UNet incorporating highly efficient channel attention modules, achieving 96% liver segmentation and 91% tumor segmentation accuracy. However, this evaluation was conducted on a subset of the LiTS dataset with small tumors removed, highlighting the ongoing challenge of comprehensive evaluation standards.

The literature reveals persistent challenges in standardized evaluation and dataset availability. While the LiTS and IRCAD datasets serve as common benchmarks, many studies employ private datasets or modified evaluation protocols, limiting direct performance comparisons.

2.3 Liver Vessel Segmentation

Proper vascular function remains fundamental to tissue health throughout the human body, with vessel abnormalities manifesting as significant pathological conditions including atherosclerosis and cerebrovascular accidents. Medical imaging and computational analysis have substantially advanced vessel delineation capabilities, enabling enhanced diagnostic and therapeutic applications. Vessel segmentation from medical images serves as an essential preprocessing step for surgical planning and vascular disease diagnosis, with methodological approaches varying according to imaging modality, contrast quality, automation level, and segmentation technique employed.

Traditional vessel segmentation approaches relied on manual delineation across sequential image slices, requiring substantial expertise while proving both labor-intensive and time-consuming. The emergence of deep learning methodologies for medical image segmentation has facilitated the development of automated vessel segmentation approaches that address these limitations. Classical vessel segmentation methods have been comprehensively reviewed by Kirbas and Quek [22], while Moccia et al. [26] provide extensive analysis of blood vessel segmentation algorithms across various imaging modalities and evaluation frameworks.

The following section presents a focused examination of liver vessel segmentation from contrast-enhanced computed tomography using deep learning approaches, emphasizing recent methodological developments and performance characteristics within this specialized domain.

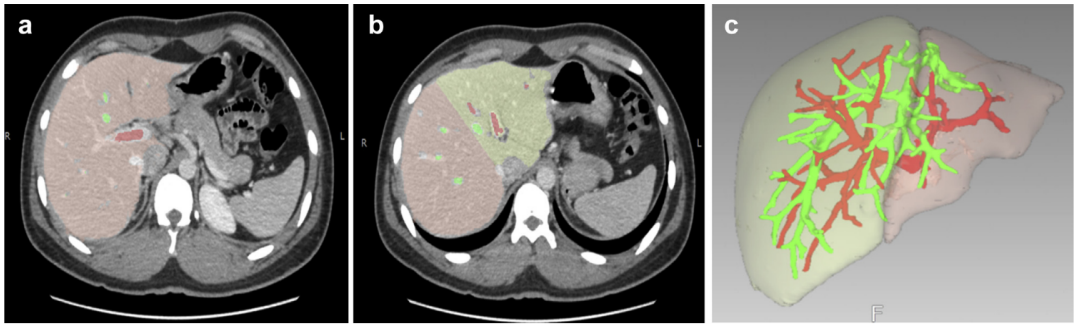


Figure 2.3: Multimodal visualization of liver vessel segmentation. (a, b) Axial CT slices and (c) 3D reconstructed liver model.

2.3.1 Literature Review

Jin et al. propose a dual-stage transformer-based framework for liver vessel segmentation that leverages global contextual modeling and local spatial refinement [19]. In the first stage, a coarse segmentation of the liver vessels is generated using a Vision Transformer (ViT) that captures global features from the liver region cropped using a pretrained liver segmentation model. The second stage refines the coarse predictions using a hybrid CNN-transformer module that integrates fine-grained spatial details. This framework is evaluated on the IRCAD dataset, with 16 cases used for training and 4 for testing. The method achieves an average Dice score of 92%, outperforming several baseline CNN-based models and demonstrating enhanced performance in delineating thin and low-contrast vessels. Notably, their ablation study shows that combining transformer-based attention with traditional convolutional layers improves both recall and vessel continuity.

2.4 Mixed Reality in Surgical Planning

Mixed Reality technology represents a paradigm shift in surgical planning and intraoperative assistance by seamlessly integrating physical and digital environments. This technology enables real-time interaction with three-dimensional holograms that are spatially anchored within the real-world environment, providing enhanced depth perception and spatial awareness capabilities that are particularly valuable for complex surgical procedures.

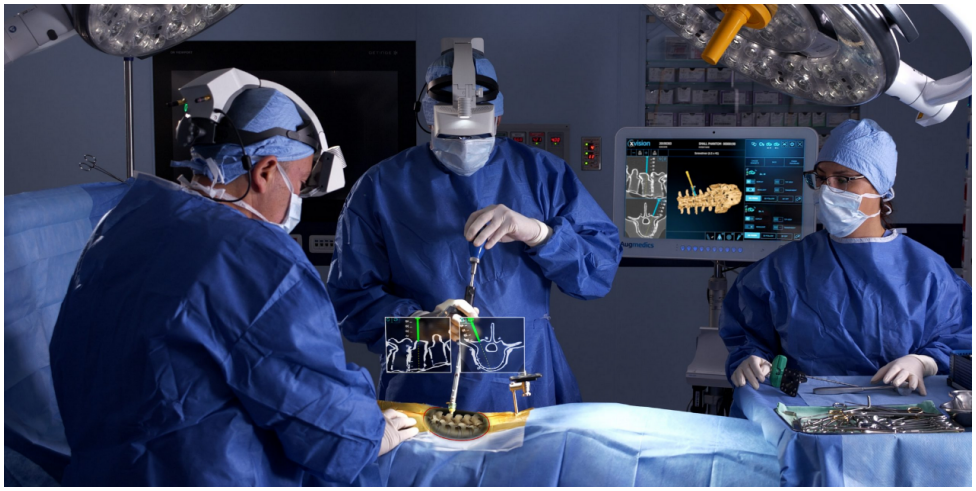


Figure 2.4: Mixed Reality applied in surgery for preoperative planning and intraoperative assistance.

Recent developments between 2022 and 2024 have demonstrated the practical application of MR devices, including Microsoft HoloLens and Meta Quest 3, across various surgical specialties. The CarnaLife Holo system developed by MedApp S.A. in Poland exemplifies this advancement by converting medical imaging data into interactive three-dimensional holograms for both preoperative planning and intraoperative guidance. Preliminary evaluation of this system in pediatric oncological surgery revealed that MR implementation did not significantly extend surgical procedure duration or hospitalization time, establishing its viability as a promising surgical tool [1].

Figure 2.5: Visual comparison of XR headsets. On the left: Meta Quest 3, On the right: Microsoft HoloLens 2.

Surgical navigation applications have shown particularly compelling efficiency improvements. An augmented reality-mixed reality navigation system for high tibial osteotomy demonstrated substantial reductions in preoperative planning time, decreasing from conventional method durations of 30.5 to 75.5 minutes to just 4 minutes. Similarly, intraoperative time requirements were reduced from 31.5 minutes or 10.5 minutes using conventional approaches to 8.5 minutes with the MR system. Although this system exhibited slightly lower accuracy compared to traditional methods, the significant efficiency gains position it as a valuable alternative for appropriate clinical scenarios [36].

The application of augmented reality in surgical consent and anatomical understanding has emerged as another significant benefit area. Comprehensive reviews of AR applications from 2019 to 2023 have highlighted substantial improvements in mental comprehension of complex medical concepts and enhanced visualization of anatomical structures. These capabilities prove particularly valuable for the surgical consent process, where patient understanding of proposed procedures is essential [37]. Additionally, HoloLens technology has gained increasing adoption in medical education applications, including anatomy instruction and medical imaging training [13].

Despite these promising developments, current MR devices present several inherent limitations that must be addressed for optimal surgical implementation. The field of view constraint represents a fundamental challenge, as MR headsets like the HoloLens provide a more restricted visual field compared to natural human vision, potentially limiting immersion and comprehensive situational awareness during surgical procedures.

Latency issues pose another critical concern, particularly for surgical applications where temporal precision is paramount. Real-time tracking, registration of holograms with the physical environment, and rendering processes can introduce perceptible delays that may compromise surgical accuracy and workflow efficiency.

Depth sensor accuracy represents an additional technical limitation, as these sensors can be affected by environmental noise, reducing the precision of environmental mapping capabilities. The fixed focus plane limitation, such as the two-meter focus distance of the HoloLens, can particularly impact the precision required for close-proximity surgical procedures where millimeter-level accuracy is essential [13].

The novelty bias phenomenon must also be considered when evaluating MR technology benefits. Initial participant enthusiasm for new technology may lead to overestimation of perceived benefits, potentially skewing early adoption assessments and requiring careful evaluation protocols to distinguish genuine utility from technological novelty appeal [37].

The consistent identification of latency and field of view as fundamental limitations across different MR systems suggests these constraints are inherent to current technology generations. Consequently, project design must accommodate these limitations while future development efforts should focus on mitigation strategies and leverage anticipated advancements in subsequent hardware generations to address these foundational challenges.

Chapter 3

Methodology Part I: Segmentation and 3D Reconstruction

3.1 Dataset and Preprocessing

3.1.1 Dataset

We used three publicly available CT datasets commonly applied in abdominal imaging research:

LiTS (Liver Tumor Segmentation Challenge): This dataset contains CT scans focused on liver and tumor segmentation. It is widely used to benchmark algorithms on liver lesion detection and volumetry [3].

Figure 3.1: Example from the LiTS dataset showing liver and tumor segmentation on axial CT slices.

IRCAD: The IRCAD dataset consists of high-resolution, manually annotated CT images of the liver, including tumors and hepatic vasculature. It is often used for detailed anatomical analysis and vessel-tumor interaction studies [7].

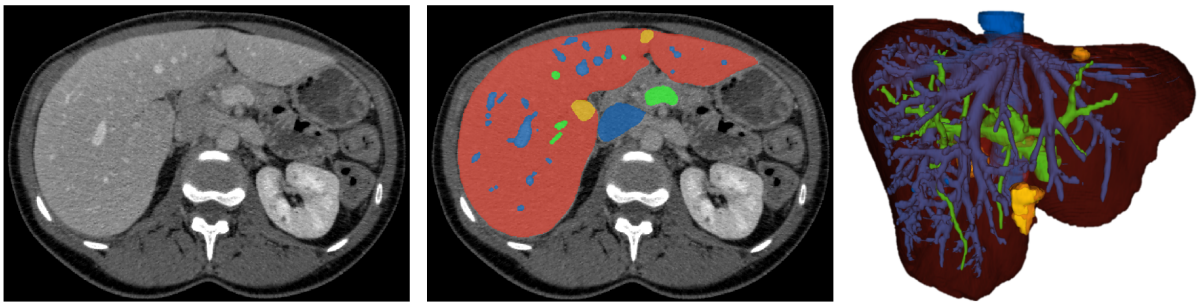


Figure 3.2: Example from the IRCAD dataset showing annotated liver, vessels, and tumor regions.

MDHV (Medical Decathlon Hepatic Vessel): Part of the Medical Segmentation Decathlon challenge, this dataset provides CT scans with annotations specifically targeting hepatic vessel segmentation, useful for vascular topology learning [33].

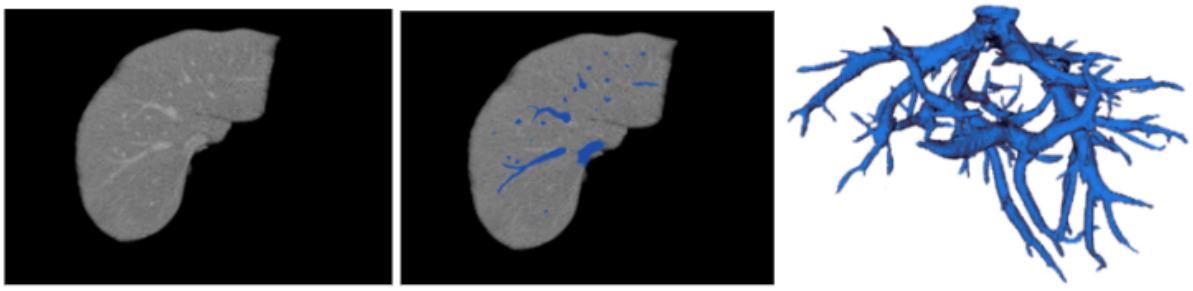


Figure 3.3: Example from the MDHV dataset: (Left) raw CT image, (Middle) vessel mask overlaid on CT, (Right) 3D reconstruction of segmented hepatic vessels.

Summary Table of Dataset Characteristics

Properties	LiTS	IRCAD	MDHV
Use Case	Liver, Tumor	Liver, Tumor, Vessels	Vessels
Dataset Size	201 (131/70)	20	443 (303/140)
Available Annotations	Train: Yes, Test: No	Yes	Train: Yes, Test: No
In-plane Resolution (mm)	0.76 (0.56, 1.00)	0.74 (0.56, 0.87)	0.80 (0.56, 0.97)
Slice Thickness (mm)	1.0 (0.7, 5.0)	1.6 (1.0, 4.0)	5.0 (0.8, 8.0)
Number of Slices	432 (74, 987)	99 (41, 187)	49 (24, 181)

Table 3.1: Characteristics of publicly available CT datasets. In-plane resolution, slice thickness, and number of slices are expressed as median (min, max).

3.1.2 Preprocessing

Prior to model training, a comprehensive preprocessing pipeline was applied to ensure data uniformity and enhance model performance. The preprocessing steps are described below.

- **Resampling:** CT images often exhibit anisotropic voxel spacing — for example, in-plane resolution might be 0.7×0.7 mm while the slice thickness is 5 mm. To address this:
  - A resampling strategy was implemented to standardize all scans to isotropic spacing (typically $1 \times 1 \times 1$ mm^3).
  - A threshold value (`ANISO_THRESHOLD = 3`) was defined to determine whether a volume was anisotropic (e.g., the z-spacing is three times the x/y spacing).

  Isotropic Volumes: Volumes with nearly equal spacing in all directions were interpolated directly using trilinear or bicubic interpolation to standardize resolution.

  Anisotropic Volumes: For volumes with significant spacing disparities, slice-wise processing was considered or higher-order interpolation was avoided to prevent spatial distortion.

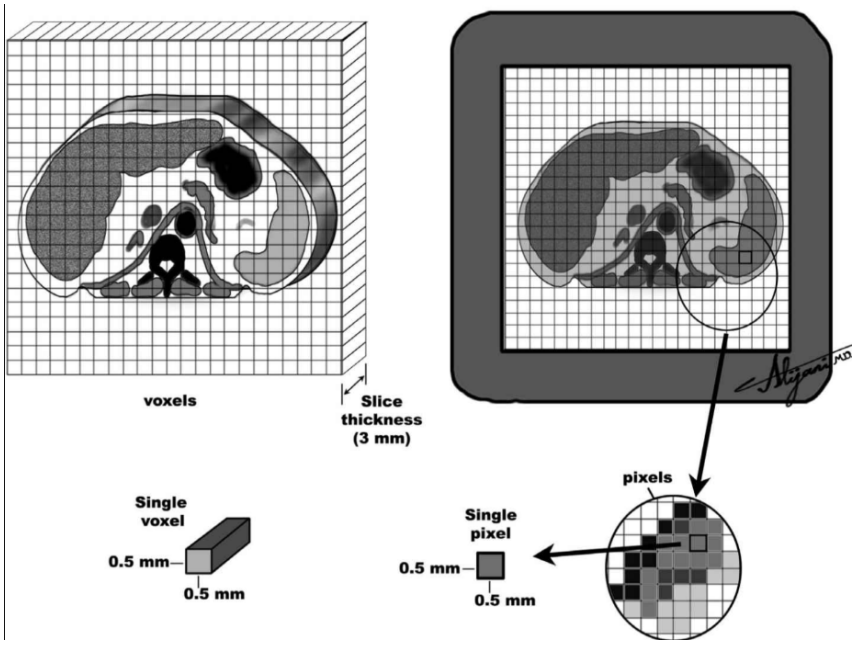


Figure 3.4: Comparison between anisotropic (left) and isotropically resampled (right) CT volumes.

- **Intensity Windowing and Normalization:** CT scan intensities are measured in Hounsfield Units (HU), a radiodensity scale. To focus on soft tissues, intensities were clipped to the range $[-100, 400]$ HU, a standard liver window.

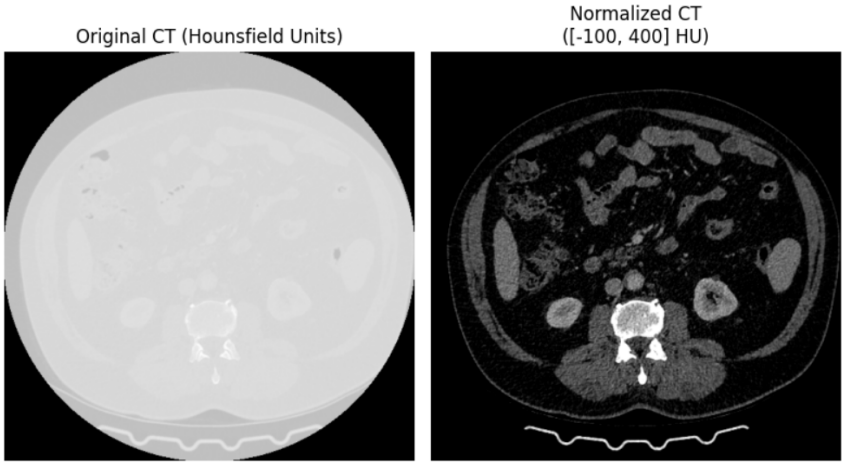


Figure 3.5: Effect of intensity windowing on CT data. Left: Original CT image in raw Hounsfield Units. Right: CT image clipped to $[-100, 400]$ HU for soft tissue visualization.

  - This step discards irrelevant intensity values (e.g., bone, air) and emphasizes liver and tumor structures.

- **Cropping (Bounding Box Extraction):** To optimize training efficiency and focus the model on relevant anatomical structures:

  - A tight 3D bounding box was computed around the liver (and tumor) region by identifying non-zero voxels in the segmentation masks.
  - A few slices above and below the liver were retained to preserve anatomical context.
  - Cropped sub-volumes reduced memory usage and increased the model's focus on pathological regions, particularly for detecting small lesions.

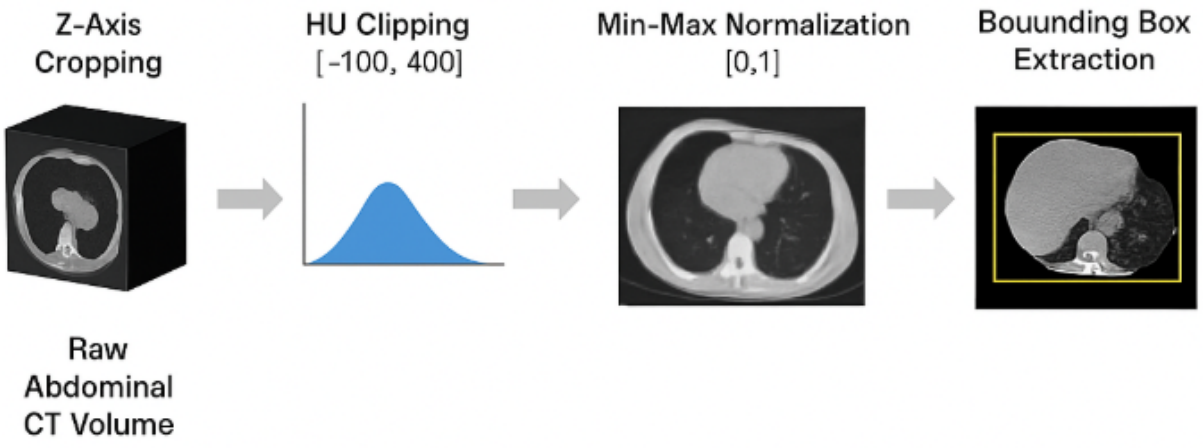


Figure 3.6: Preprocessing pipeline for liver CT scans including cropping, HU clipping, normalization, bounding box extraction.

3.2 Evaluation Metrics

3.2.1 Dice Loss

Dice Loss is a popular metric for evaluating segmentation performance, especially in medical imaging. It measures the overlap between the predicted segmentation and the ground truth [2], defined as:

$$\text{Dice Loss} = 1 - \frac{2|P \cap G|}{|P| + |G|}$$

where P represents the predicted segmentation set and G the ground truth set.

3.2.2 IoU Loss

Intersection over Union (IoU), also known as the Jaccard Index, is another common metric for evaluating segmentation performance. It quantifies the similarity between the predicted segmentation and the ground truth by comparing their overlap relative to their union [2]:

$$\text{IoU Loss} = 1 - \frac{|P \cap G|}{|P \cup G|}$$

where P is the predicted segmentation set and G is the ground truth set.

3.3 Segmentation Models Implemented

We implemented and evaluated multiple deep learning architectures for medical image segmentation.

Since the IRCAD dataset is relatively small, with only 20 patients, we propose a two-stage system to improve segmentation performance. The first stage focuses on liver and tumor segmentation, followed by a second stage for hepatic vessel segmentation.

3.3.1 Liver Tumor Segmentation

2D Approach

For the 2D approach, we employed the traditional 2D U-Net architecture, which has been widely used for image segmentation tasks in medical imaging.

- **2D U-Net:** The 2D U-Net architecture is a well-established convolutional neural network designed specifically for image segmentation. It follows a symmetric encoder-decoder structure with skip connections [30], as illustrated in Figure 3.7.

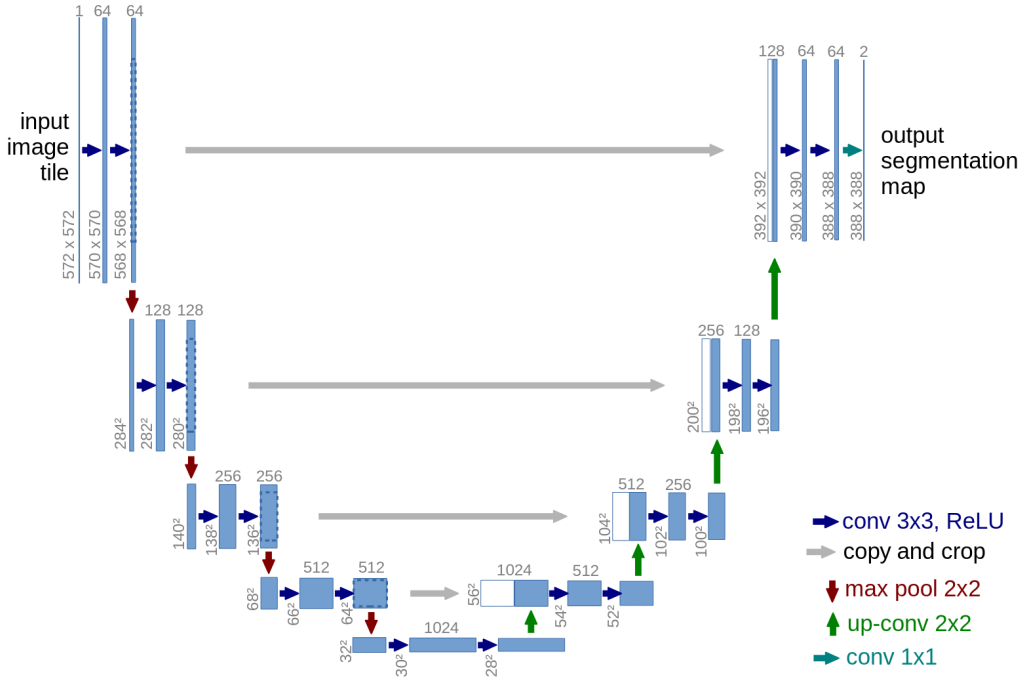
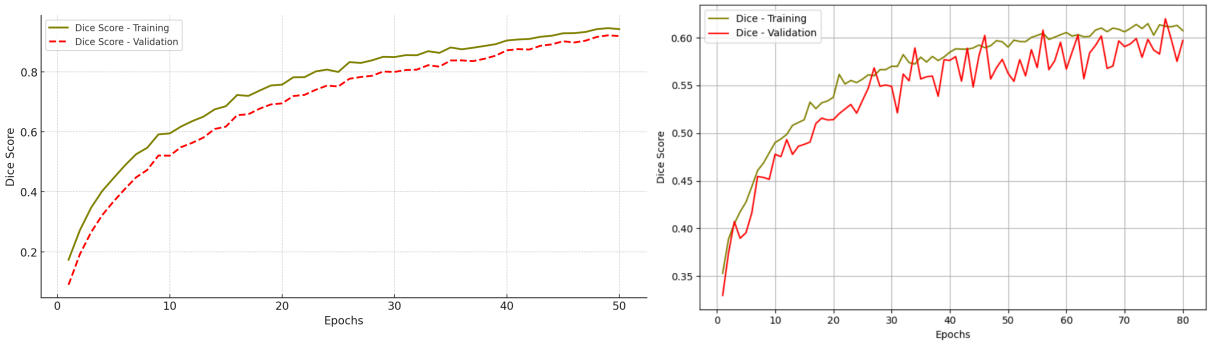


Figure 3.7: Architecture of the 2D U-Net.

Configuration of Training:

Parameter	Liver/Tumor Segmentation
Input Channels	1 (Grayscale CT)
Output Classes	3 (Background, Liver, Tumor)
Image Size	512×512
Normalization	Yes
Architecture	U-Net
Batch Size	8
Epochs	50
Optimizer	Adam
Learning Rate	1×10^{-4}
Final Dice Score (Liver)	0.931
Final Dice Score (Tumor)	0.613

Table 3.2: Training configuration and results for the 2D U-Net model used in liver and tumor segmentation.



(a) Dice score evolution (training and validation) for liver segmentation

(b) Dice score evolution training and validation) for tumor segmentation (

Figure 3.8: Training and validation Dice score curves over 50 epochs. (a) Liver segmentation. (b) Tumor segmentation.

2. 2.5D Approach

The 2.5D approach extends the 2D model by incorporating neighboring slices to provide context along the depth dimension.

Models: Attention U-Net, Dense U-Net

- **Attention U-Net:** Introduces attention gates that help the model focus on relevant structures such as the liver and tumors [28].

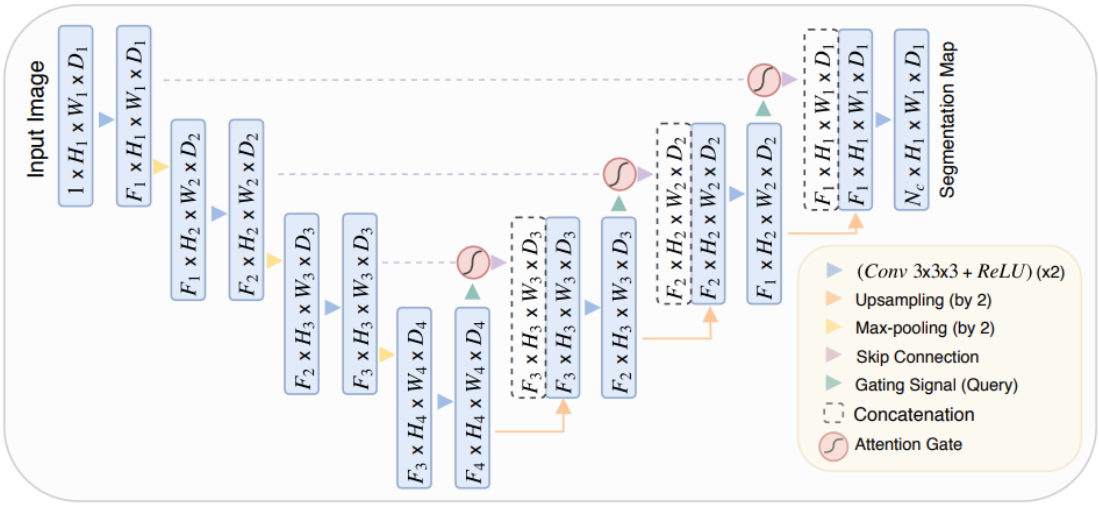


Figure 3.9: Architecture of the Attention U-Net [28].

- **Dense U-Net:** Employs dense connections to improve feature reuse and gradient propagation [4].

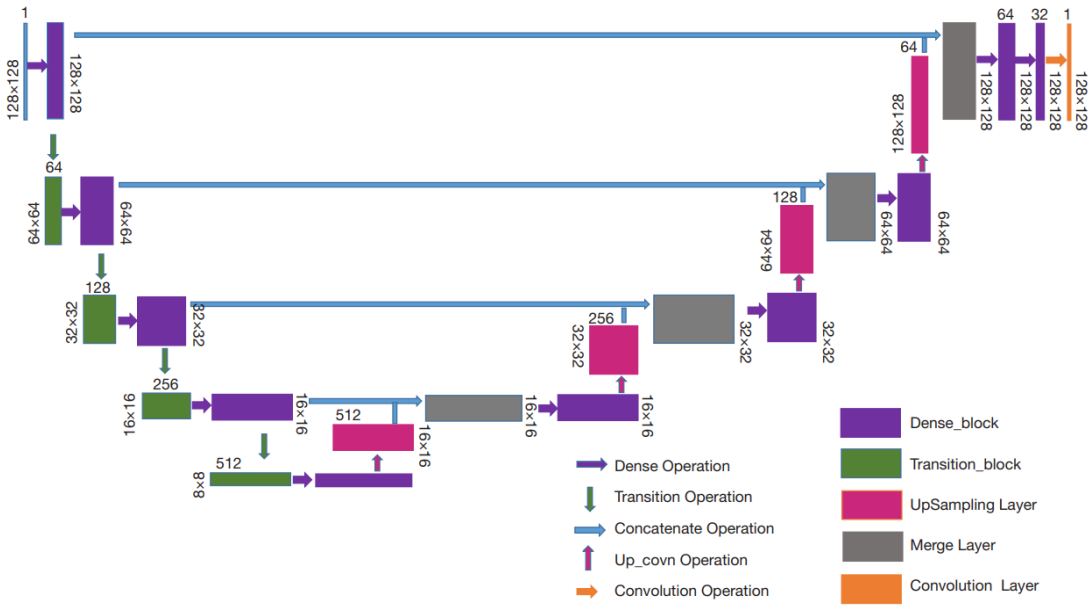


Figure 3.10: Architecture of Dense U-Net.

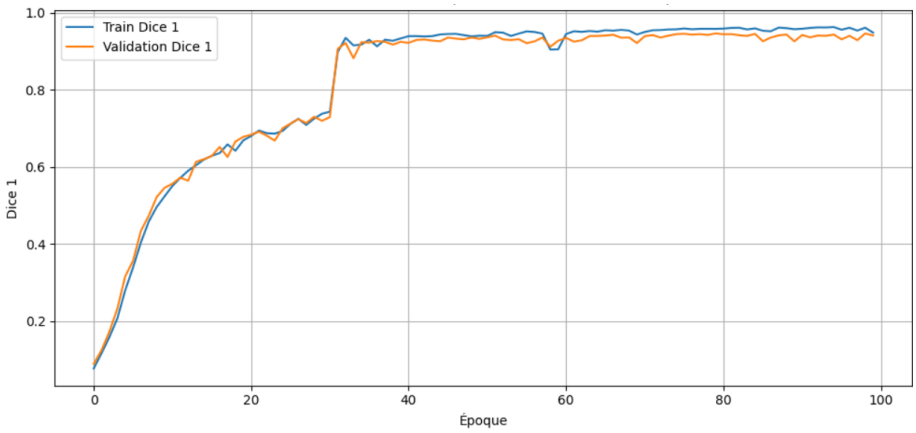
Configuration of Training:

Parameters	Attention U-Net	Dense U-Net
Depth	4	4
Dropout	0.2	–
Epochs	50	50
Batch size	8	8
Batch normalization	True	True
Loss	Dice loss	Dice loss
Optimizer	Adam	Adam
Momentum	0.99	0.99
Learning rate	0.0001	0.0001

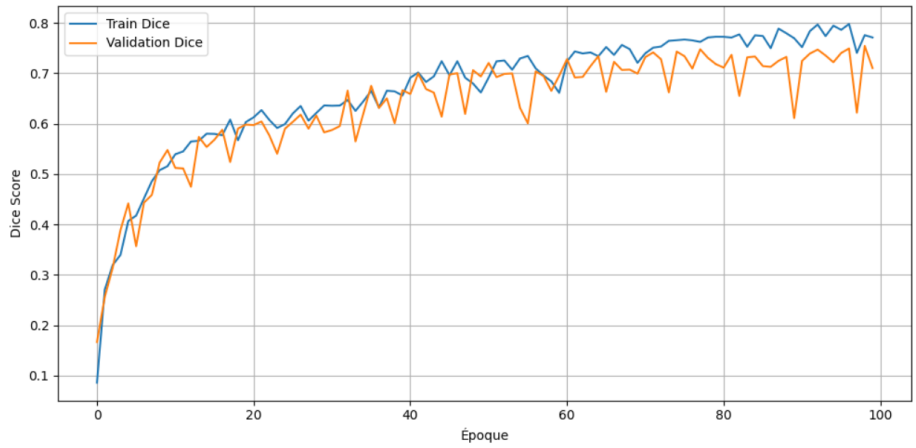
Results:

Method	Att U-Net	Dense U-Net
Dice (liver)	0.951	0.949
Dice (tumor)	0.65	0.600

Table 3.3: Dice scores for liver and tumor segmentation (best results highlighted).



(a) Dice score evolution for liver segmentation using Attention U-Net (2.5D)



(b) Dice score evolution for tumor segmentation using Attention U-Net (2.5D)

Figure 3.11: Training and validation Dice score curves for liver (a) and tumor (b) segmentation using the best-performing 2.5D model: Attention U-Net.

3. 3D Approach

The 3D approach handles full volumetric data, enabling better spatial context extraction in all directions.

Models: 3D U-Net and 3D ResUNet

- **3D U-Net:** A direct 3D extension of the classic U-Net [5].

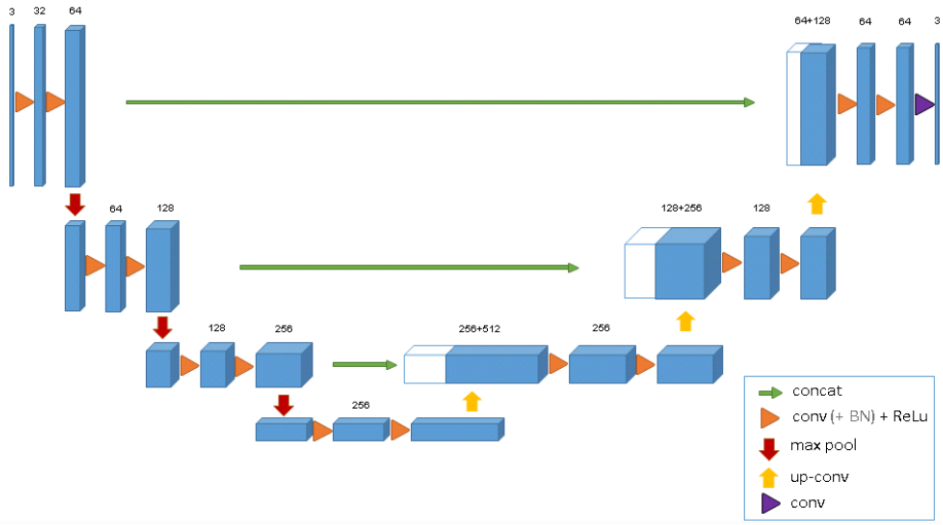


Figure 3.12: Architecture of the 3D U-Net [5].

- **3D ResUNet:** Incorporates residual connections to improve training depth and stability [39].

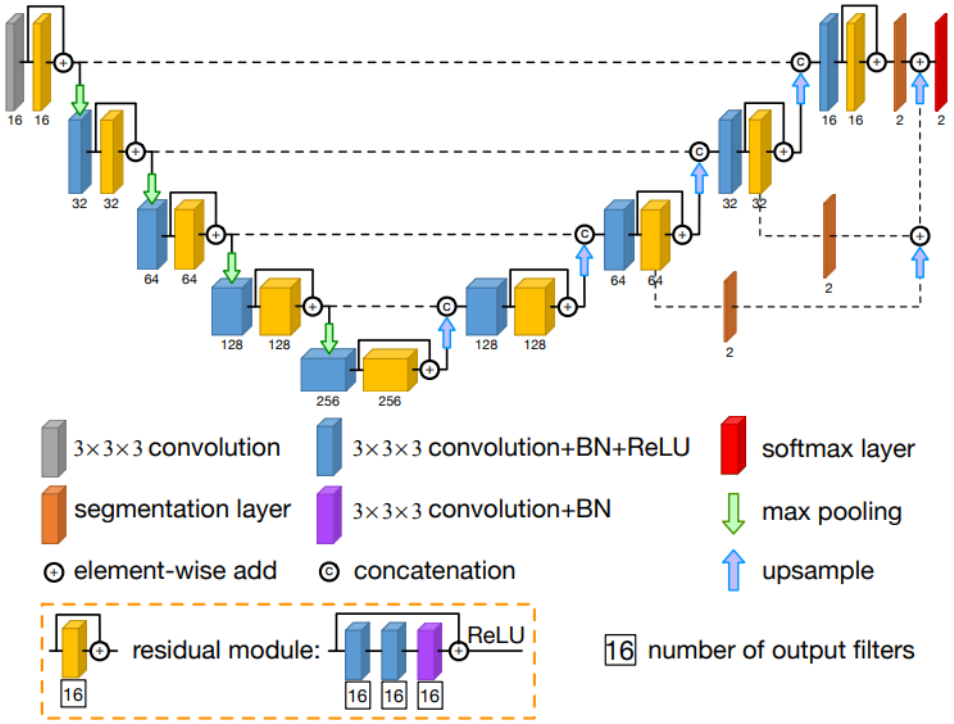


Figure 3.13: Architecture of the 3D Residual U-Net [39].

Input Strategies for 3D Models:

- **Patch-based:** The 3D volume is divided into smaller patches.

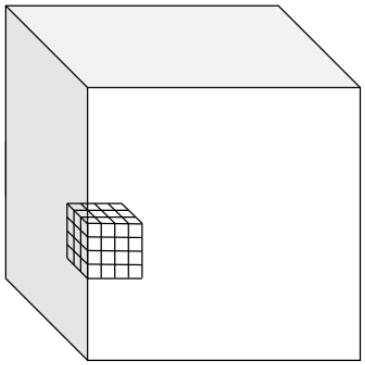


Figure 3.14: Patch-based approach.

- **Slab-based:** Small contiguous blocks of slices (slabs) are processed.

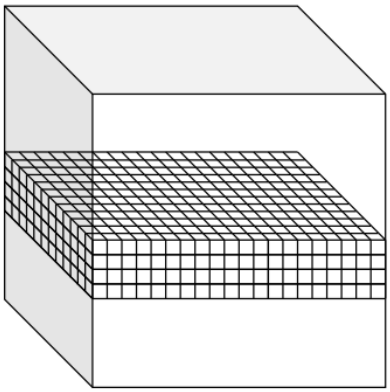


Figure 3.15: Slab-based approach.

- **Full-volume:** The entire 3D CT volume is input at once.

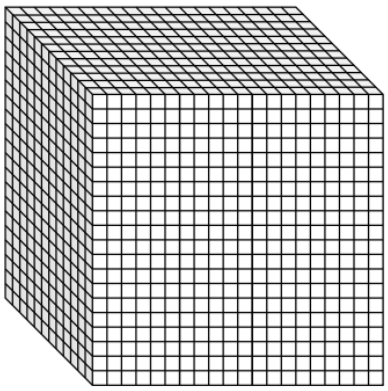


Figure 3.16: Full-volume approach.

Configuration of Training:

Parameter	Value
Train/Val Split	80% / 20%
Batch Size	4
Normalization	Yes
Model Architecture	3D UNet, 3D ResUnet
Epochs	50 (with early stopping after 100 epochs)
Learning Rate	3×10^{-4}
Optimizer	Adam
Loss Function	BCE + Dice (`BCEDiceLoss`)
Metrics	Dice, IoU
Checkpoint Saving	Best model saved based on validation loss

Dice Scores for Liver Segmentation (3D Models)

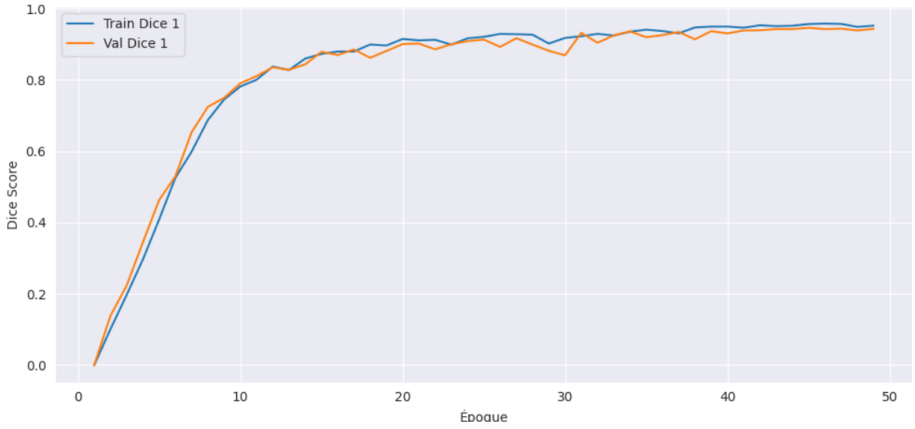
Set-up	3D U-Net	3D ResU-Net
Full 3D	0.951	0.963
Slabs-based	0.947	0.953
Box-based	0.920	0.943

Table 3.4: Dice scores for liver segmentation using different 3D input strategies.

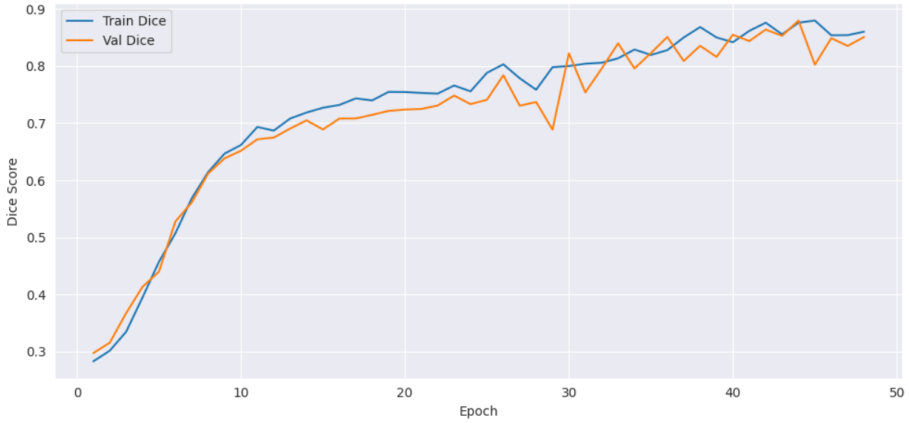
Dice Scores for Tumor Segmentation (3D Models)

Set-up	3D U-Net	3D ResU-Net
Full 3D	0.734	0.88
Slabs-based	0.700	0.863
Box-based	0.551	0.764

Table 3.5: Dice scores for tumor segmentation using different 3D input strategies.



(a) Dice score evolution for liver segmentation using 3D ResU-Net



(b) Dice score evolution for tumor segmentation using 3D ResU-Net

Figure 3.17: Training and validation Dice score curves for liver (a) and tumor (b) segmentation using the best-performing model: 3D ResU-Net (full volume setup).

3.3.2 Hepatic Vessel Segmentation via Filter-enhanced and 3D U-Net

For the hepatic vessel segmentation task, we retained the 3D U-Net architecture, which is well suited for volumetric medical image processing. Instead of modifying the model itself, we enhanced its input by applying vesselness filters to better emphasize vascular structures within the CT scans.

Hepatic vessels are particularly challenging to segment due to their small caliber, branching geometry, and poor contrast in CT images. Standard intensity-based segmentation methods often fail to capture their complex topology. Therefore, we explored several vesselness filters that leverage second-order intensity information to enhance tubular structures before feeding the volumes into the 3D U-Net.

These filters are typically based on the Hessian matrix $\mathbf{H}(f)$ of the image function $f(x_1, x_2, x_3)$, which encodes second-order partial derivatives:

$$\mathbf{H}(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 \partial x_3} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} \end{bmatrix} \tag{3.1}$$

To handle the discrete nature of digital images, f is approximated by smoothing the input image I with a Gaussian kernel of standard deviation σ, enabling multi-scale analysis.

Let $\lambda_1, \lambda_2, \lambda_3$ be the eigenvalues of $\mathbf{H}(f)$ ordered as $|\lambda_1| \leq |\lambda_2| \leq |\lambda_3|$. Vessel-like structures exhibit the pattern:

$$|\lambda_1| \approx 0, \quad \lambda_2 \approx \lambda_3 \ll 0 \tag{3.2}$$

We evaluated the following vesselness filters:

Frangi filter [10]: Uses ratios of eigenvalues to enhance tubular structures. Three derived measures are:

$$R_b = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}}, \quad R_a = \frac{|\lambda_2|}{|\lambda_3|}, \quad S = \sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2} \tag{3.3}$$

Combined in the vesselness function:

$$F = \left(1 - e^{-\frac{R_a^2}{2\alpha^2}}\right) \cdot e^{-\frac{R_b^2}{2\beta^2}} \cdot \left(1 - e^{-\frac{S^2}{2c^2}}\right) \tag{3.4}$$

This response is applied only when $\lambda_2, \lambda_3 < 0$.

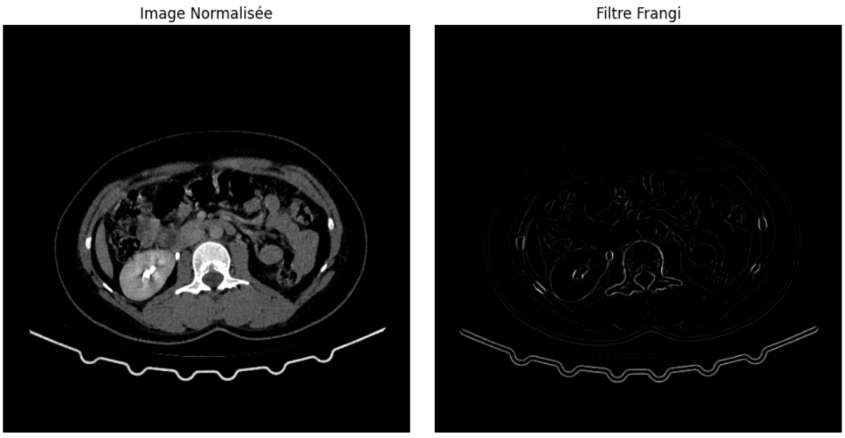


Figure 3.18: Effect of the Frangi filter on a CT scan slice.

Sato filter [31]: Uses an asymmetric formulation based on the sign of λ_1:

$$F = \begin{cases} \lambda_c \cdot \exp\left(-\frac{\lambda_1^2}{2(\alpha_1 \lambda_c)^2}\right) & \text{if } \lambda_1 \leq 0, \lambda_c \neq 0 \\ \lambda_c \cdot \exp\left(-\frac{\lambda_1^2}{2(\alpha_2 \lambda_c)^2}\right) & \text{if } \lambda_1 > 0, \lambda_c \neq 0 \\ 0 & \text{if } \lambda_c = 0 \end{cases} \tag{3.5}$$

with $\lambda_c = \min(-\lambda_2, -\lambda_3)$.

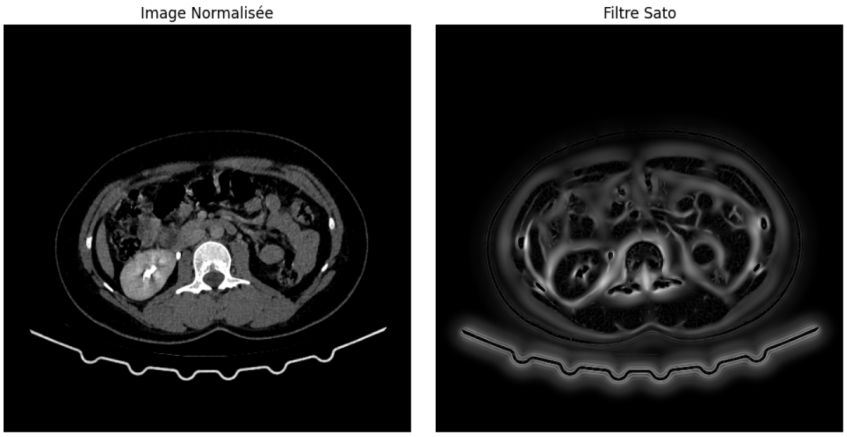


Figure 3.19: Effect of the Sato filter on a CT scan slice.

Jerman filter [18]: Introduces a regularized eigenvalue λ_ρ for bifurcation robustness:

$$\lambda_\rho = \begin{cases} \lambda_3 & \text{if } \lambda_3 > \tau \cdot \max_x \lambda_3(x) \\ \tau \cdot \max_x \lambda_3(x) & \text{if } 0 < \lambda_3 \leq \tau \cdot \max_x \lambda_3(x) \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

Then the vesselness function is defined as:

$$F = \begin{cases} 0 & \text{if } \lambda_2 \leq 0 \text{ or } \lambda_\rho \leq 0 \\ 1 & \text{if } \lambda_2 > \lambda_\rho/2 > 0 \\ \frac{\lambda_2^2(\lambda_\rho - \lambda_2)^3}{(\lambda_2 + \lambda_\rho)^3} & \text{otherwise} \end{cases} \tag{3.7}$$

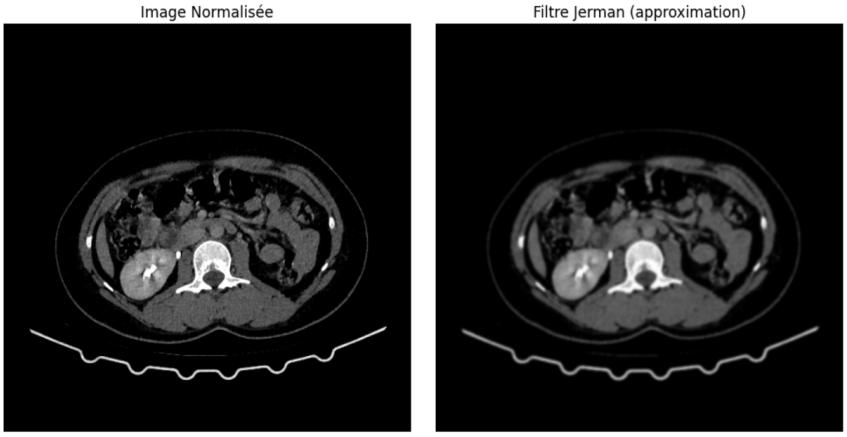


Figure 3.20: Effect of the Jerman filter on a CT scan slice.

Zhang filter [40]: Builds on Jerman's function by incorporating tissue-specific enhancement through clustering. The vesselness is further modulated by an exponential term over the squared sum of eigenvalues:

$$F = (\text{JermanResponse}) \cdot \left(1 - \exp\left(-\frac{3(\lambda_1^2 + \lambda_2^2 + \lambda_\rho^2)}{2\lambda}\right)\right) \tag{3.8}$$

where λ is a tunable contrast control parameter.

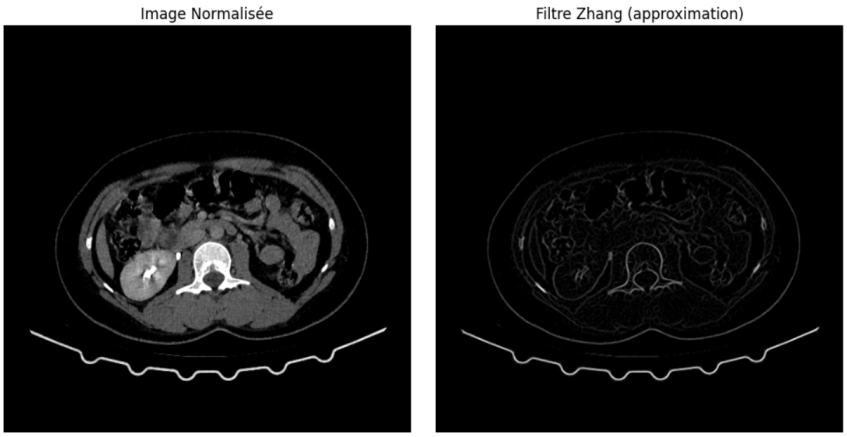


Figure 3.21: Effect of the Zhang filter on a CT scan slice.

Training Configuration for Vascular Segmentation

Parameter	**Value**
Train/Val Split	80% / 20%
Image Size	512×512
Number of Classes	2 (background, vessel)
Batch Size	8
Filter Type	Frangi, jerman, sato
Normalization	Yes
Model Architecture	3D UNet
Input Channels	2 (Raw CT + Enhanced)
Output Channels	1
Feature Maps	[64, 128, 256, 512]
Epochs	50
Learning Rate	1×10^{-4}
Optimizer	Adam
Loss Function	Dice + Cross-Entropy (0.5 / 0.5)

Table 3.6: Configuration used for vascular segmentation training with preprocessing and model setup.

Results:

Filter	Dice Score
Frangi	0.5
Jerman	0.481
Zhang	0.561

Table 3.7: Dice scores of the filters.

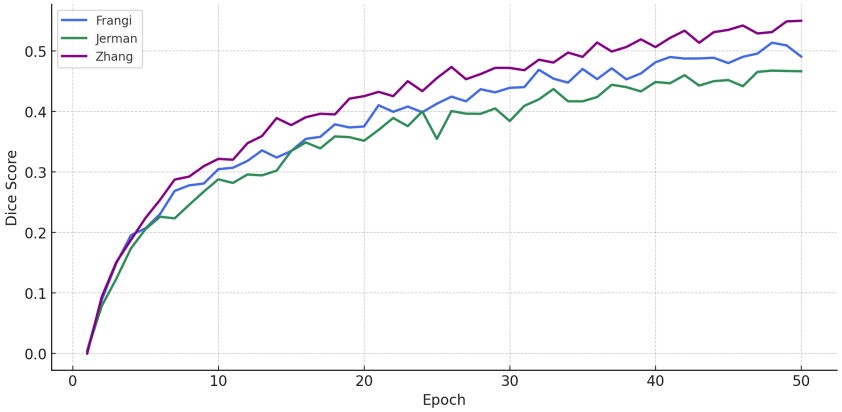


Figure 3.22: Dice score evolution across epochs for the three vessel enhancement filters (Frangi, Jerman, Zhang).

3.4 Discussion

3.4.1 Interpretation of Results

This subsection discusses the performance of the segmentation models on liver, tumor and vessels tasks.

Liver and Tumor Segmentation

Across all experiments, liver segmentation showed consistently high performance across all models, with Dice scores above 0.94 in most cases. The liver, being a large and relatively homogeneous organ, is easier to segment accurately, especially with volumetric models.

Tumor segmentation, however, proved more challenging due to their small size, irregular shape, and intensity similarity with healthy tissue. The best tumor segmentation results were obtained using the **3D ResUNet with full-volume input**, achieving a Dice score of 0.880. This confirms that both architectural depth (residual connections) and full spatial context are crucial for accurate tumor detection.

Impact of Model Dimensionality and Input Strategy:

- **2D U-Net:** Adequate for liver segmentation (Dice = 0.931), but limited in tumor detection (Dice = 0.613) due to lack of inter-slice context.

- **2.5D Models:** Slight improvements by including neighboring slices, but still inferior to volumetric approaches.
- **3D Patch-based Models:** Improved tumor segmentation (up to 0.764 Dice for ResUNet), but suffered from loss of global anatomical context.
- **3D Slab-based Models:** Balanced trade-off between context and memory usage. 3D ResUNet with slab input achieved 0.863 Dice for tumor.
- **3D Full-volume Models:** Best overall results. 3D ResUNet reached **0.961** for liver and **0.880** for tumor, highlighting the benefit of full spatial continuity and residual learning.

Interpretation

2D and 2.5D Approaches

The 2D U-Net model show satisfactory segmentation results for the liver, achieving a Dice score of 0.931. However, performance declined for tumor segmentation, with a Dice score of 0.613. This discrepancy highlights the limitation of slice-wise 2D segmentation in capturing 3D spatial continuity.

The 2.5D approaches, including Attention U-Net and Dense U-Net, offered only marginal improvements or parity.

3D Volumetric Context and Architectures

The 3D U-Net and ResUNet models leveraged full volumetric context. ResUNet outperformed U-Net due to residual connections that improved gradient flow and boundary learning. Full-volume ResUNet achieved Dice scores of **0.961** for liver and **0.880** for tumors, the best in the study.

Approach	Model	Liver Dice	Tumor Dice
2D	U-Net	0.931	0.613
2.5D	Att U-Net	0.951	0.650
	Dense U-Net	0.949	0.600
3D Patch	3D U-Net	0.920	0.551
	3D ResUNet	0.943	0.764
3D Slab	3D U-Net	0.947	0.700
	3D ResUNet	0.953	0.863
3D Full	3D U-Net	0.951	0.734
	3D ResUNet	**0.963**	**0.880**

Table 3.8: Comparison of Dice scores for liver and tumor segmentation across all approaches.

Liver Vessel Segmentation

Across all experiments, liver segmentation showed consistently good performance, with Dice scores above 0.5 in most cases.

The best results were obtained using the **3D U-Net combined with the Zhang filter**, achieving a Dice score of 0.561. These results suggest that while vessel enhancement contributes positively, it represents only an initial step in segmentation pipeline.

3.4.2 Challenges Encountered

Automatic 3D segmentation of hepatic structures presents several challenges:

- **Anatomical Complexity:** The liver varies greatly between patients. Ambiguous boundaries with adjacent organs (spleen, heart, stomach) and low tissue contrast complicate segmentation [25].
- **Tumor Variability:** Tumors vary in shape, size, and intensity. Small or low-contrast lesions are difficult to identify, even for experts [8].
- **Vascular Complexity:** Vessels are thin, branching, and have variable diameters. Maintaining topological continuity is challenging, especially with low-contrast CT data [12].
- **Dataset Limitations:** Public datasets (e.g., 3D-IRCADb1) are small, prone to inconsistent annotations, and do not always include all hepatic structures [20].
- **Computational Demands:** Full-resolution 3D volumes require high GPU memory. Training on patches reduces context, which may impair learning [25].

3.5 3D Reconstruction

3.5.1 Marching Cubes Algorithm

The Marching Cubes algorithm, introduced by Lorensen and Cline in 1987 [24], is a widely used method for extracting isosurfaces from 3D scalar fields, such as those obtained from CT or MRI scans. It is especially popular in medical imaging and scientific visualization for reconstructing 3D surface models from volumetric data.

Overview

Given a 3D scalar volume $V(x, y, z)$, the algorithm generates a surface mesh corresponding to a specified isovalue τ. This isosurface represents all points in the volume where the scalar value equals τ, i.e.,

$$\mathcal{S} = \{(x, y, z) \mid V(x, y, z) = \tau\} \tag{3.9}$$

The method proceeds by traversing the volume in a grid of *cubes*, where each cube is defined by 8 neighboring voxels. For each cube, the algorithm determines which corners lie above or below the isovalue and uses this information to generate a set of triangles that approximate the surface within the cube.

Algorithm Steps

The Marching Cubes algorithm can be summarized in the following steps:

1. **Cube Construction:** Divide the scalar volume into small cubes formed by 8 adjacent voxel values.
2. **Classification:** For each cube, compare the scalar value at each of the 8 corners to the isovalue τ. Label each corner as either inside (value $< \tau$) or outside (value $> \tau$).
3. **Index Computation:** Use the inside/outside labels to construct an 8-bit index representing the cube's configuration (total of $2^8 = 256$ possible cases).
4. **Triangle Lookup:** Use a precomputed lookup table (edge and triangle tables) to determine which edges of the cube are intersected by the surface, and how to connect those intersections into triangle(s).
5. **Vertex Interpolation:** For each intersected edge, linearly interpolate the surface intersection point using the scalar values at the edge's endpoints:

$$\mathbf{p} = \mathbf{p}_1 + \frac{\tau - V_1}{V_2 - V_1}(\mathbf{p}_2 - \mathbf{p}_1) \tag{3.10}$$

   where $\mathbf{p}_1$ and $\mathbf{p}_2$ are the voxel coordinates, and V_1, V_2 are the scalar values.
6. **Mesh Generation:** Create triangle(s) inside the cube using the interpolated points and add them to the output mesh.

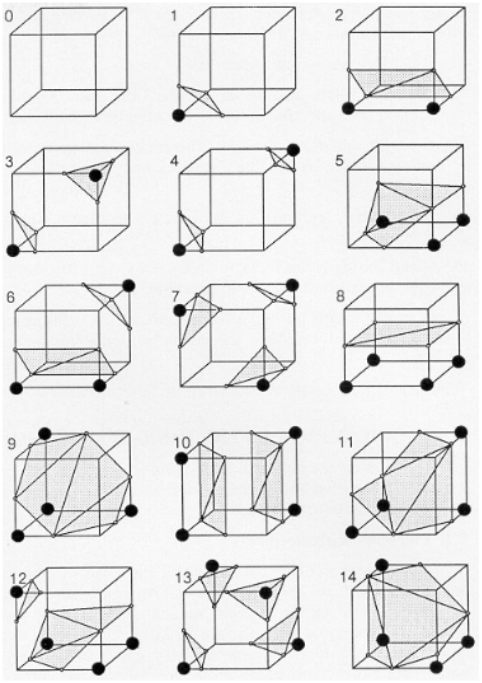


Figure 3.23: triangulated cubes

Advantages and Limitations

Marching Cubes is efficient and simple to implement, and it generates high-quality triangle meshes for smooth surfaces. However, it has limitations:

- It cannot accurately represent sharp features (e.g., corners or edges).
- It may produce non-manifold meshes or ambiguities in some configurations.
- It generates a dense mesh even in flat regions, leading to redundant triangles.

3.6 Flying Edges Algorithm

The Flying Edges algorithm, introduced by Lawlor et al. [32], is a high-performance isosurface extraction method that improves upon the classical Marching Cubes algorithm by leveraging modern CPU architectures and memory access patterns. It is particularly efficient for large volumetric datasets commonly encountered in medical imaging and scientific computing.

Motivation

Although Marching Cubes is widely used, it has several performance limitations:

- Redundant computation of edge intersections across neighboring cubes.
- Inefficient memory usage and branching.
- Poor cache utilization, especially on modern CPUs.

Flying Edges addresses these issues through parallel processing, edge-based traversal, and minimized memory access, while maintaining the same surface quality as Marching Cubes.

Key Concepts and Workflow

Flying Edges rethinks the traversal pattern of the volume by operating along the primary axes (usually the x-axis), rather than marching through full cubes. The algorithm performs three main passes:

1. **Edge Classification (Pass 1):** The scalar values are evaluated along the primary edges (e.g., x-axis), and edges are classified as active (cross the isovalue) or inactive. This pass computes the number of intersections and prepares prefix sums for memory allocation.
2. **Vertex Generation (Pass 2):** Interpolation is performed on the active edges to compute surface vertices, similar to Marching Cubes. These vertices are stored in a compact data structure, reusing shared edges to avoid duplication.
3. **Triangle Construction (Pass 3):** Using the previously computed edge information and a lookup table (analogous to Marching Cubes), the algorithm constructs triangle primitives across cells using the interpolated vertices.

Advantages Over Marching Cubes

Flying Edges provides several important advantages:

- **Thread-parallelism:** The algorithm is designed to be easily parallelized across multiple CPU cores.
- **Reduced memory usage:** Shared edge data is reused, and unnecessary computations are skipped.
- **Improved performance:** Cache-friendly data access and reduced branching allow faster execution on modern CPUs.
- **Same mesh quality:** The output triangle mesh is topologically and geometrically equivalent to that of Marching Cubes.

Comparison with Marching Cubes

Feature	**Marching Cubes**	**Flying Edges**
Traversal Strategy	Cube-by-cube	Axis-aligned edge-by-edge
Parallel Execution	Limited	Highly parallel
Memory Efficiency	Moderate	High (reuse of edge data)
Performance	Slower on large data	Significantly faster
Output Quality	High	High (same as MC)

Table 3.9: Comparison of Marching Cubes and Flying Edges

Chapter 4

Methodology Part II: Mixed Reality and Interaction System

4.1 Definition of Mixed Reality (MR)

Definition

Mixed Reality (MR) is a type of technology where **real and virtual elements are combined** in a way that they can **interact with each other in real time**.

You still see the real world around you (like with Augmented Reality), but **virtual objects appear to exist inside that real world**—and you can interact with them just like physical objects [16].

More Detailed Explanation

The virtuality continuum

The virtuality continuum represents the full spectrum of technological possibilities between the entirely physical world or real environment and the fully digital world or virtual environment. It includes all current technologies that alter reality with computer-generated graphics as well as those yet to be developed.

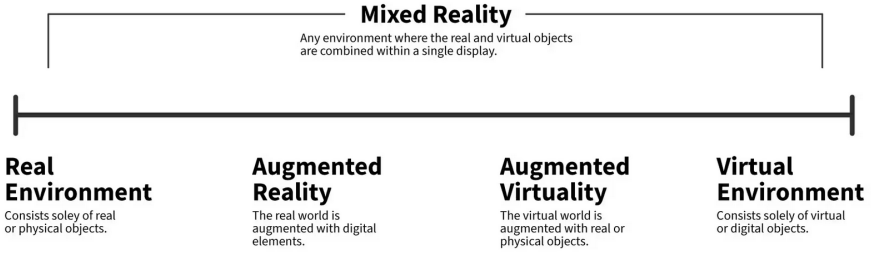


Figure 4.1: The virtuality continuum [17]

In a continuum, adjacent parts are almost indistinguishable, but the extremes are very different. Therefore, the exact limits of the various terms are not a hundred percent clear. The term mixed reality covers any environment where the real and virtual objects are combined within a single display. According to this framework, mixed reality covers most of the continuum except for the endpoints. The researchers Paul Milgram and Fumio

Kishino first introduced the virtuality continuum or reality-virtuality continuum concept in 1994 [17].

Example:

Imagine you're wearing a Mixed Reality headset like the *Microsoft HoloLens* or *Meta Quest 3* (in MR mode):

- You look at your desk, and you see a **virtual 3D brain model** floating above it.
- As you move closer, the brain **stays in place**, and you can walk around it to view it from every angle.
- You can **touch it with your hands**, and it rotates or opens up to show its parts.
- If you push it, it might **fall on the floor (virtually)** and bounce, just like a real object.

This is not just a video effect—it's an **interactive simulation** where real and virtual worlds are **fused together**.

How It Works

Mixed Reality uses a combination of technologies:

- **Cameras and sensors** to scan your environment (walls, tables, lighting).
- **Spatial mapping** to understand where virtual objects can be placed.
- **Head tracking** to know where you're looking or moving.
- **Hand tracking or controllers** for interactions.
- **3D rendering engines** (like Unity or Unreal Engine) to create the visuals.

These technologies work together to make it feel like **virtual objects are really there**, and not just "floating" on top of your camera view like in AR.

4.2 VR Headsets

4.2.1 definition

A VR headset is a head-mounted device that provides immersive virtual experiences. Also known as a head-mounted display, it typically includes a pair of lenses that users look through, a screen (or screens) inside the device, and a mechanism to secure it to the head [15].

4.2.2 Evolution of VR Headsets

Year	Milestone	Description
1832	The Stereoscope – Sir Charles Wheatstone	Used angled mirrors to view two images side-by-side, creating a 3D effect; foundation of stereoscopic photography.
Late 1800s–Early 1900s	Stereoscopic Viewers	Popular devices for viewing stereoscopic images of faraway places; early examples of virtual-like experiences.
1968	Sword of Damocles – Ivan Sutherland	First-ever head-mounted display (HMD); overlaid basic wireframe graphics onto the real world.
1980s	VPL Research – Jaron Lanier	Developed early commercial VR headsets and coined the term "Virtual Reality"; introduced gloves and goggles for interactive simulation.
1990s	Virtuality Arcade Systems	Public VR arcade machines with HMDs and motion-tracked gameplay; showcased early interactive 3D environments.
2012	Oculus Rift Kickstarter – Palmer Luckey	Launched the first modern consumer-grade VR headset; sparked mass interest and major industry investment.
2016–2020	Consumer Headsets (HTC Vive, PSVR, Oculus)	Introduced room-scale tracking, motion controllers, and high-resolution displays; brought VR to gamers and developers.
2023–2024	Standalone Headsets (Apple Vision Pro, Meta Quest)	Modern VR headsets with wireless freedom, eye/hand tracking, haptics, and spatial computing, enabling fully immersive interaction.

Table 4.1: Evolution of VR Headsets[15]

4.2.3 Components of a VR headset

A virtual reality headset includes the following components [15]:

Display: One or two screens that display stereoscopic images. Stereoscopic images show slightly different versions of the same image for each eye to create a three-dimensional effect, thus giving the illusion of depth and space.

Lenses: Lenses focus the pictures for each eye, creating a convincing 3D virtual environment. They also help to enlarge the image, filling the user's field of vision for a more immersive experience.

Tracking Sensors: VR headsets can include various sensors such as gyroscopes (to track orientation), accelerometers (for movement), and cameras (to detect the user's hand movements and surroundings). These track the user's head movements and adjust the image accordingly to ensure that the virtual environment aligns with physical movements.

Input Devices: Many headsets work with handheld controllers or gloves that track hand and finger movements. Some also support voice commands, eye tracking, and even full-body tracking for a more interactive experience.

Spatial Audio: Headsets often include built-in headphones or earphones that provide spatial audio to mimic how people hear sound in the real world. Audio enhances immersion, as the user can experience sounds as if they were coming from specific directions and distances within the virtual environment.

Comfort Features: Good headsets must be comfortable, especially for extended use. Adjustable straps, padding and balanced weight distribution minimize discomfort.

Connectivity: VR headsets may connect to a computer or a gaming console or operate as standalone devices. Tethered headsets require a physical connection (often via HDMI or USB) to a PC or console, while standalone headsets are self-contained units with their own onboard processing power

4.3 System Architecture

4.3.1 unity Engine

definition

Unity is a real-time development platform and game engine used to create 2D, 3D, Virtual Reality (VR), and Augmented Reality (AR) applications. It provides all the tools needed to build interactive experiences—from games to simulations to XR applications.

It's especially popular because it is:

- Easy to learn for beginners
- Very powerful and flexible for professionals
- Supports many platforms like Windows, Android, iOS, Web, VR/AR headsets, etc.

Unity is a cross-platform engine, which means you can write your code once and deploy it to many platforms. It uses C# as its main programming language and offers a powerful visual editor for building scenes, importing 3D models, managing lighting, adding animations, and handling physics.

Main Components of Unity Engine [35]

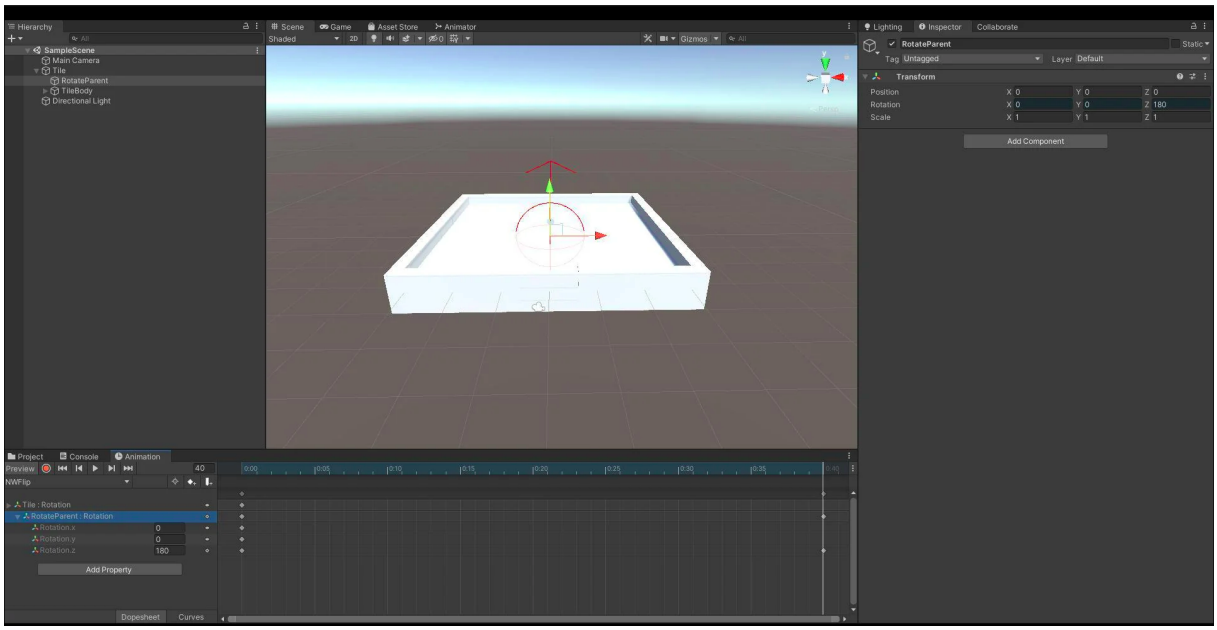


Figure 4.2: Main components of the Unity interface

Scene View

The Scene View is your design workspace—it's where you visually build and arrange your game or XR environment. You can:

- Drag and drop 3D models, cameras, lights, and other objects into the scene.
- Move, rotate, or scale objects using the transform tools.
- Use different views (top, side, perspective) to place objects accurately in 3D space.
- In XR apps, layout virtual objects in the real-world context (e.g., place a 3D organ on a table).

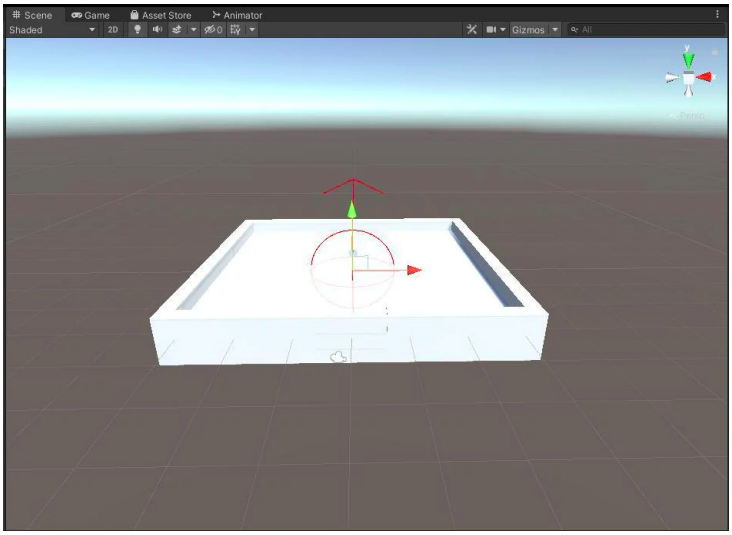


Figure 4.3: Unity Scene View

Game View

The Game View shows what the user will actually see when running the game or app. It is a preview mode that renders the scene through the active camera.

- Test how your scene looks and behaves.
- Switch screen resolutions or aspect ratios to simulate different devices.
- Simulate VR/AR views using XR tools.

Hierarchy

The Hierarchy window in Unity displays all the GameObjects in the currently loaded Scene, such as models, cameras, lights, and prefabs, organized in a tree structure. It reflects the structure and relationships of these objects, including parent-child hierarchies, allowing users to view, group, and organize elements within a Scene. It also supports multi-Scene editing, sorting options (e.g., alphabetical or transform order), and override indicators for prefab modifications. This window plays a central role in managing the contents and structure of a Scene.

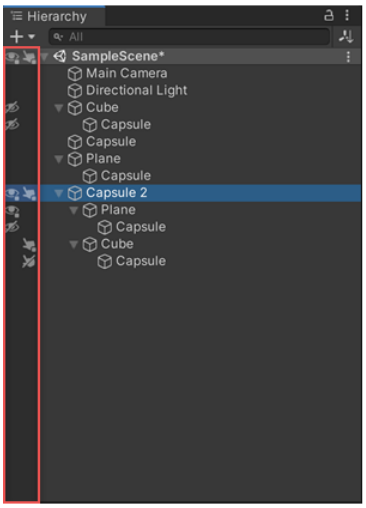


Figure 4.4: The Hierarchy window in Unity showing a structured view of all GameObjects in the active Scene, organized in a parent-child hierarchy.

Inspector

The Inspector window displays the properties and components of the selected GameObject.

- Modify position, rotation, scale, materials, animations, and more.
- Add or remove components like Rigidbody, Colliders, or custom scripts.

Project Panel

The Project window in Unity is a key interface that displays and organizes all the files and assets in a project, allowing developers to browse, manage, and preview their content. It consists of a left panel showing the folder hierarchy and a right panel displaying the selected folder's contents with icons representing different asset types. Users can switch between One Column and Two Column layouts, with the latter offering visual previews. A Favorites section allows easy access to frequently used items or saved searches. The

toolbar at the top includes tools like the Create menu, search bar, search filters by type or label, and options to manage hidden packages, making the Project window essential for efficient asset navigation and organization

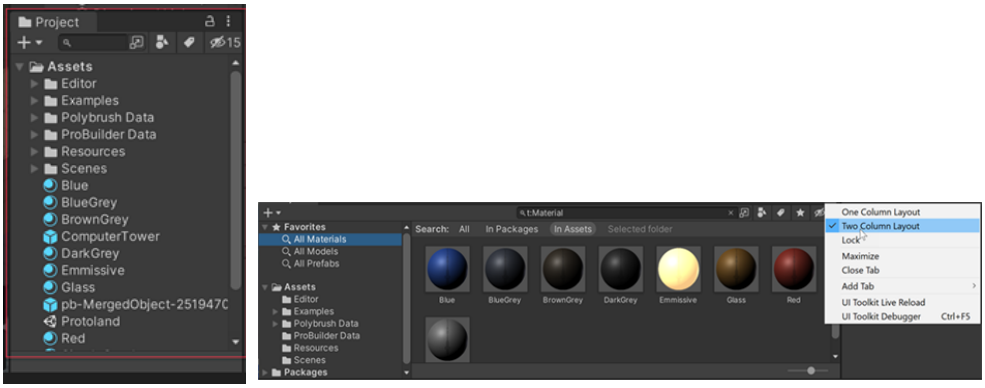


Figure 4.5: Unity Project window in One Column Layout (left) and Two Column Layout (right)

Scripting (C#)

Scripting is how you define logic and interactivity in your Unity project.

- Unity uses C# as its scripting language.
- Scripts are attached to GameObjects as components.
- Control animations, input, UI, physics, networking, and more through scripting.

Debugging Workflow During Development One major advantage of using Unity with the Meta XR All-in-One SDK is the ability to test and debug applications directly in the Unity Editor, even without building to the Meta Quest device. This is made possible through Unity's support for in-editor XR simulation and hand tracking emulation, which allows rapid iteration and real-time debugging using Unity's native debugger, console, and scene view.

However, when testing features that require reading files directly from the Meta Quest's internal storage—such as runtime `.obj` model loading in our case—a full APK build and deployment to the device becomes necessary. This introduces a debugging challenge, as Unity's editor debugger is no longer available once the app runs on the headset.

To address this, we used the Android Debug Bridge (ADB), a versatile command-line tool provided by the Android SDK. ADB allows developers to:

- Log and filter real-time application output using `adb logcat`,
- Inspect or push/pull files on the device file system,
- Monitor performance and application state during runtime,
- Run shell commands remotely from the PC to control the device.

This ADB-based workflow enabled us to capture logs, diagnose import errors, and monitor interactions directly on the Meta Quest 3, even during runtime testing outside the Unity environment. It proved essential during the runtime model loading stage, especially for resolving issues related to file paths, model formatting, or Android permissions.

Using ADB for On-Device Debugging During early development stages, Unity's ability to run the application directly within the Editor without compiling to APK proved extremely valuable. It allowed us to test core features like model loading logic, interaction setup, and camera positioning while leveraging Unity's native debugging tools. However, when we transitioned to testing runtime file access on the actual Meta Quest 3 device, a full APK build and install became mandatory—which disabled Unity Editor debugging.

To overcome this limitation, we used **Android Debug Bridge (ADB)**, the command-line tool bundled with the Android SDK. ADB enabled us to monitor real-time logs, test APK installations, and issue commands directly to the headset. For instance, running `adb logcat` allowed us to capture Unity errors and warnings during model import, interaction triggers, and hand tracking validation.

ADB also supports wireless debugging. After connecting the headset via USB and identifying the IP address using:

```
adb shell ip route
```

we enabled TCP/IP mode with:

```
adb tcpip 5555
adb connect <device-ip>:5555
```

This made it possible to test while the headset was untethered[9].

Moreover, to install updated APKs without removing the previous one, we used:

```
adb install -r myApp.apk
```

We also addressed common ADB issues, such as unrecognized devices (solved by authorizing USB debugging inside the headset and using verified USB cables) and ambiguous device targeting when both USB and Wi-Fi were active. In multi-device setups, the `-s <device-id>` flag helped us route commands to the correct device.

Altogether, ADB was indispensable for bridging the gap between Unity's simulation and real-world device behavior during runtime testing, especially for APKs accessing Quest internal storage.

4.4 Post-processing and Preparation for Mixed Reality Integration

Following the 3D segmentation process, the resulting model consists of multiple anatomical structures—such as the liver, blood vessels, and tumors—already assembled into a coherent 3D volume. However, before integrating this model into a Mixed Reality (MR) application, several preprocessing steps are required to localize, enrich, and optimize each part of the model for real-time interaction and visualization.

To perform this post-processing, we used the open-source 3D content creation software **Blender**, which offers a powerful suite of tools for coordinate management, material assignment, optimization, and animation.

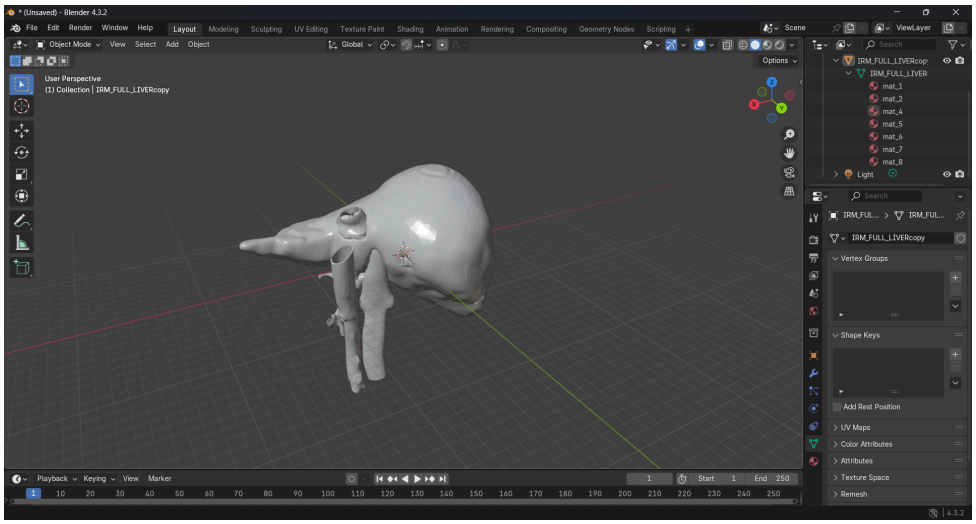


Figure 4.6: Initial 3D model composed of multiple anatomical structures (liver, vessels, tumors).

4.4.1 Preprocessing Steps

1. **Spatial Reference Assignment:** Although the anatomical structures are spatially coherent and correctly positioned in the 3D reconstruction, each part initially lacks an explicit local coordinate system. The first step consists of defining and assigning a local reference frame to each structure (e.g., origin, orientation). These frames are then anchored within a unified global reference system, which is necessary for subsequent transformations, animations, and correct alignment within the MR environment.

2. **Material and Color Assignment:** To improve visual clarity and facilitate structure identification, each anatomical part is assigned a specific material and color. For example, the liver may appear in semi-transparent red, blood vessels in blue, and tumors in yellow or orange. This color coding greatly enhances interpretability during MR visualization by clinicians.

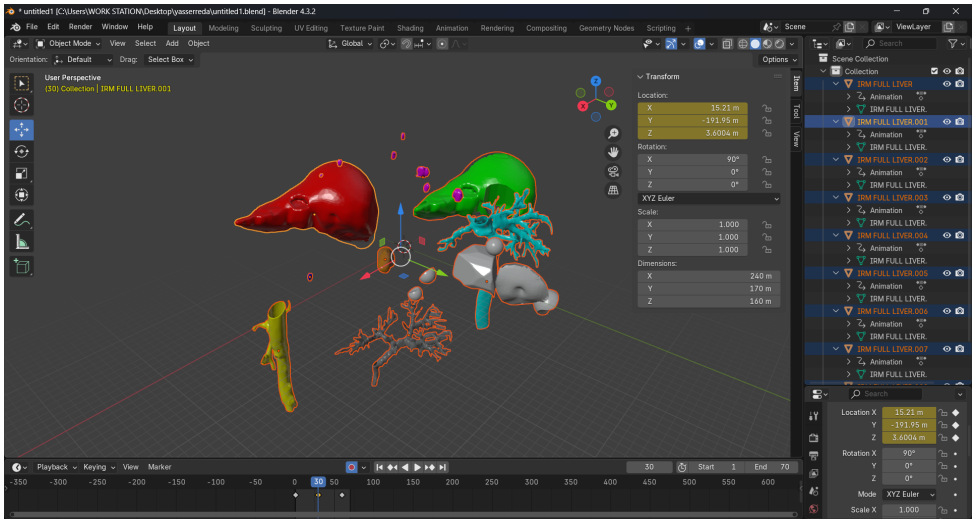


Figure 4.7: Final 3D model after preprocessing: materials assigned.

Animation for Mixed Reality Applications

A key requirement for Mixed Reality integration is the ability to animate components of the model. While the 3D model remains static in its initial form, animations allow structures to move, rotate, or change visibility over time. These animated behaviors are essential for simulating surgical procedures, educational sequences, or user-guided exploration.

Each animation frame encodes a transformation—position, rotation, and optionally scale—for a given structure relative to its reference frame. These animations are embedded in the final exported model and can be dynamically triggered within the MR application. They enable features such as:

- Simulating organ motion or virtual dissection.
- Triggering structure separation or highlighting via user interaction (e.g., hand gestures).
- Stepwise educational sequences that gradually reveal or explain anatomical content.

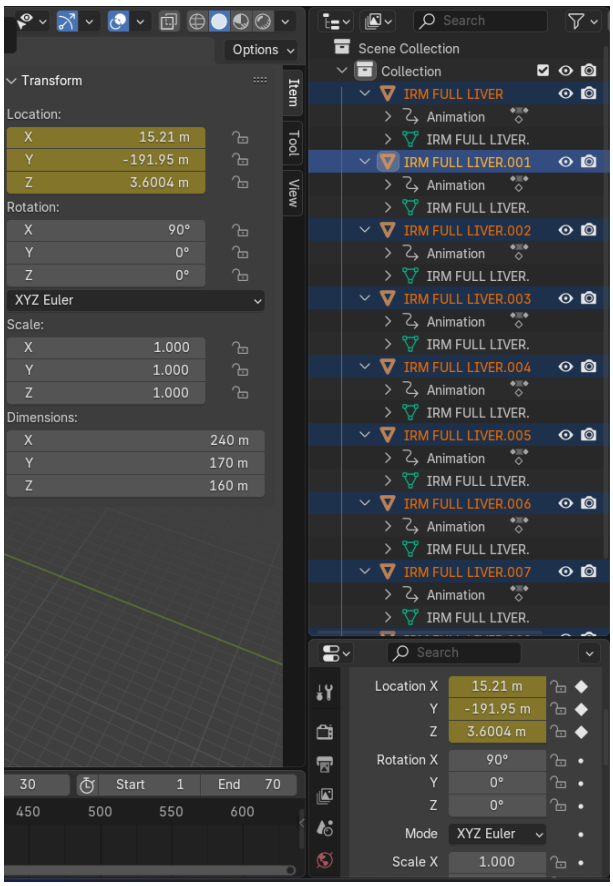


Figure 4.8: Example of an animation keyframe applied to anatomical structures in Blender.

Together, these preprocessing and animation steps serve as a crucial bridge between raw segmentation output and an interactive, immersive MR application. The result is a medically accurate, visually intuitive, and performance-optimized model tailored for clinical training, patient education, and preoperative planning.

.

4.5 Mixed reality application developmental

4.5.1 XR Foundation Setup: Meta All-in-One SDK Integration

To establish the core Mixed Reality (MR) infrastructure on Meta Quest 3, the application was developed using the **Meta XR All-in-One SDK**. This official SDK is designed to streamline XR development on Meta devices by providing modular prefabs, interaction patterns, and system components — referred to as *building blocks*.

Meta SDK Building Blocks

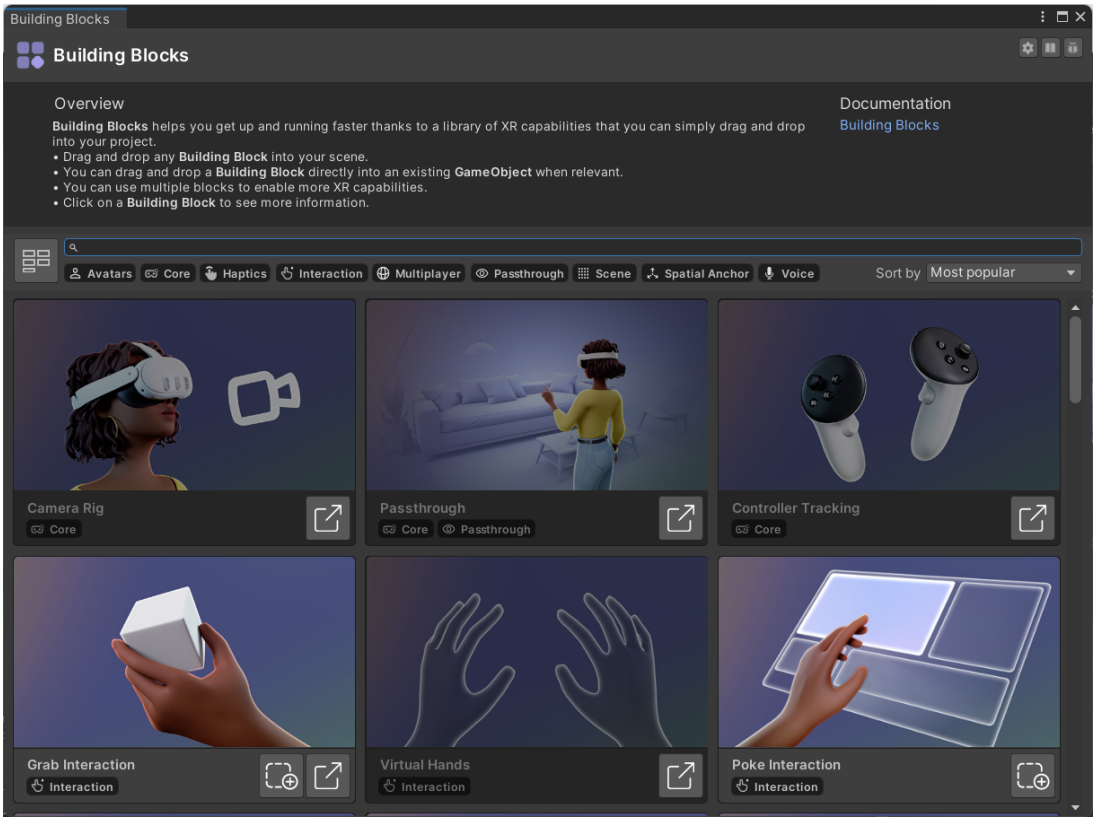


Figure 4.9: Main building blocks from Meta All-in-One SDK integrated into the MR application.

The SDK offers a range of plug-and-play modules that handle common XR functionality. The following building blocks were utilized during this stage:

- **Camera Rig (OVRCameraRig)**: A central prefab that handles stereoscopic rendering, head tracking, spatial anchoring, and serves as the base for all other head-mounted features.
- **Passthrough Layer**: Allows blending of the physical environment into the virtual scene using the Meta Quest's external cameras. It enables full passthrough or masked occlusion effects.
- **Hand Tracking and Interactors**: Includes tracking support for hands and fingers, as well as built-in components to detect grabbing, pinching, or poking gestures.

- **Interaction Prefabs**: Reusable prefabs that implement typical interaction logic (e.g., grab interactables, grabbables, physics manipulators), which are essential for later stages of the application.
- **System UI and Input Blocks**: Although not used at this stage, the SDK provides system-ready UI elements such as virtual keyboards, text input fields, and UI canvases optimized for MR.

These building blocks are designed for drag-and-drop use within Unity and are highly customizable. They significantly reduce implementation time while ensuring stability and consistency with Meta's system-level interaction patterns.

Camera Rig and Passthrough Setup

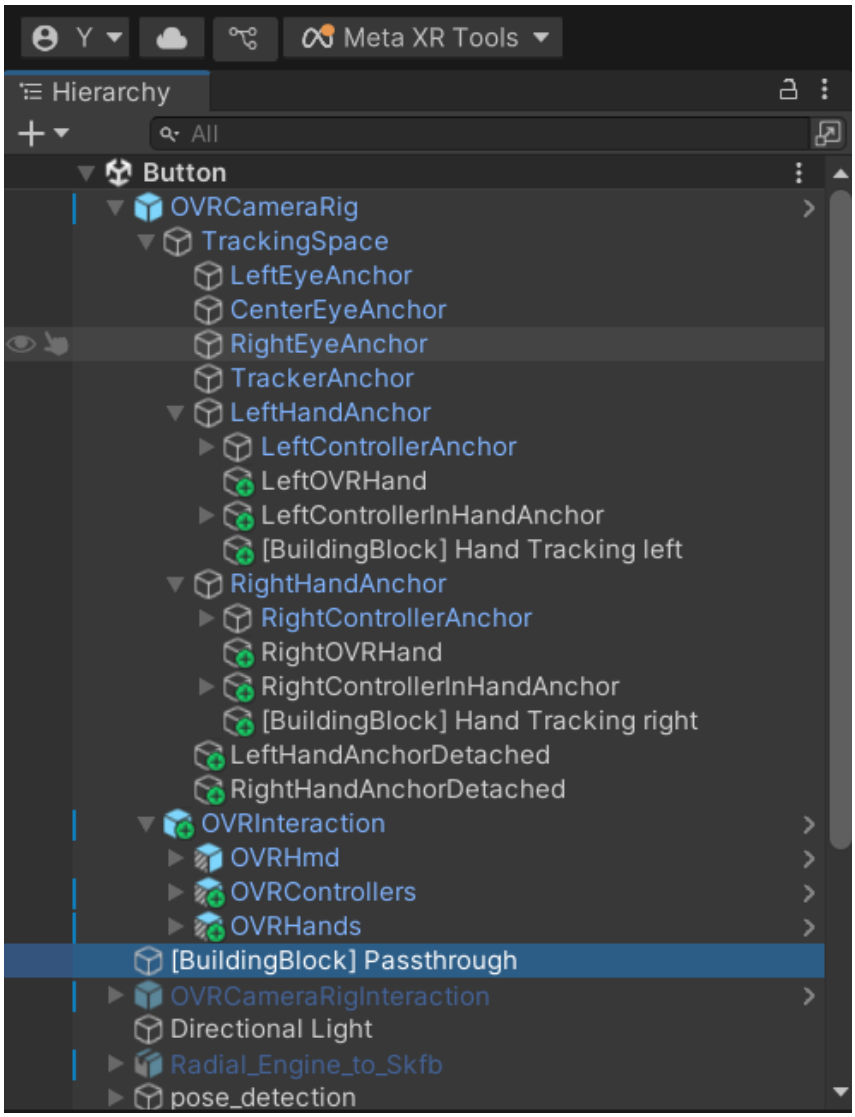


Figure 4.10: OVRCameraRig and scene hierarchy after SDK integration.

Using the SDK's building blocks, the scene was initialized with the `OVRCameraRig` prefab, which manages all head and eye tracking, stereoscopic rendering, and spatial anchoring. To enable **passthrough**, the `PassthroughLayer` component was configured and attached

to the rig. This allows the user to see the real world behind or around virtual content, establishing the foundation for Mixed Reality experiences.

4.5.2 Stage 1: Runtime 3D Model Upload on Meta Quest 3

At this early stage of development, the Mixed Reality APK built with Unity targets a critical foundational capability: loading and displaying a 3D model (`.obj` format) at runtime directly from the internal storage of the Meta Quest 3 headset. This mechanism enables fast model iteration without requiring the application to be rebuilt each time a new model is used.

Functionality Overview

When launched, the APK automatically searches for a 3D model located in the directory:

`/Android/media/com.DefaultCompany.load_obj/MyModels/`

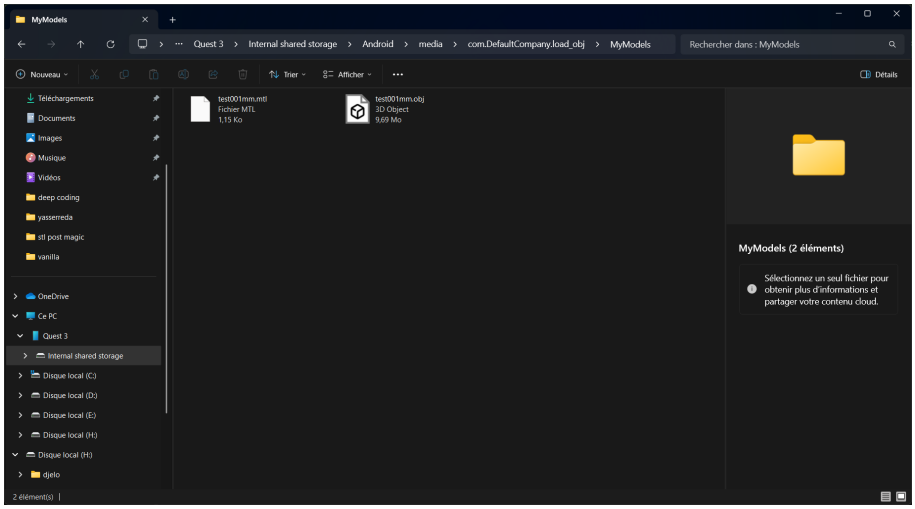


Figure 4.11: Model folder on Meta Quest 3 internal storage. The application automatically searches this path for 3D models in OBJ format at runtime.

This folder must be populated beforehand via USB or file transfer tools. The system scans this path for the first valid `.obj` file and dynamically loads it into the Unity scene at runtime. The model is rendered and placed at a predefined position in the 3D environment, visible through the headset.

At this stage, the model is static and not yet interactive. However, this system provides the backbone for future manipulation and animation capabilities.

Technical Architecture

The APK is implemented in Unity (version 2022.3 or higher) using the following key packages and libraries:

- **AsImpL**: an open-source runtime importer for `.obj` files, used to load models from the device's storage during runtime.

- **Oculus Integration SDK**: provides native support for Meta Quest hand tracking and controller input.
- **XR Interaction Toolkit (Unity)**: a cross-platform input system used to support grabbing and object manipulation.
- **Unity Netcode for GameObjects**: a networking framework that will later enable model synchronization in multi-user MR experiences.

Implementation Workflow

The runtime model upload system is implemented as a Unity C# MonoBehaviour script attached to a scene GameObject. Its behavior follows these steps:

1. At startup, the script determines the appropriate file path based on the platform:
   - On Meta Quest 3: `/Android/media/com.DefaultCompany.load_obj/MyModels/`
   - In the Unity Editor (for testing): `Application.persistentDataPath`
2. The script then enumerates all files in this directory and filters for those with the `.obj` extension.
3. If a valid model is found, the `LoaderObj` component from AsImpL is instantiated and used to import the file.
4. The model is scaled, positioned, and rotated based on predefined import options.
5. Once loaded, the GameObject is added to the scene with basic configuration.

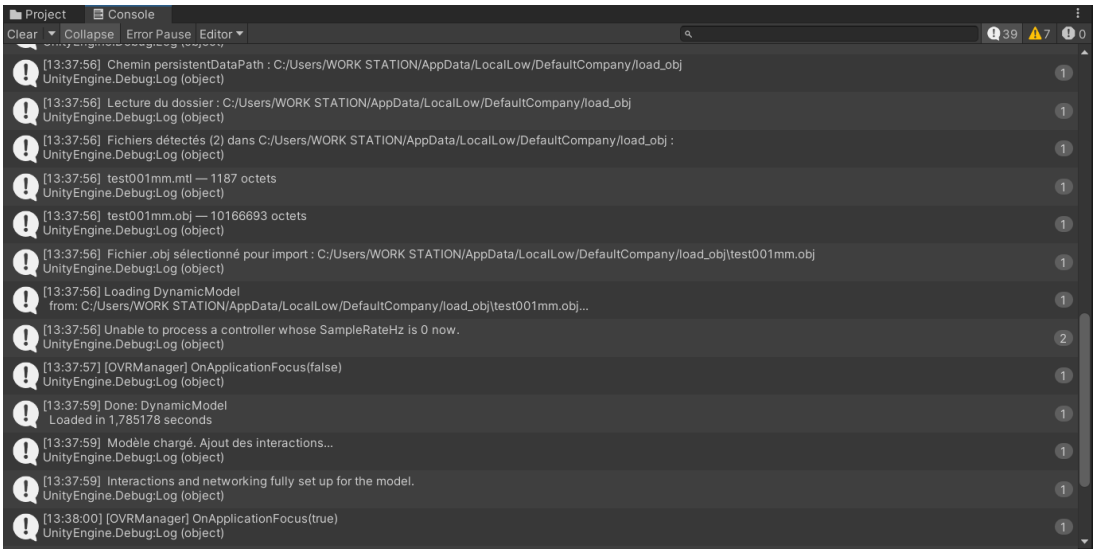


Figure 4.12: Unity log output confirming the model loading pipeline: directory access, model detection, and component assignment.

Runtime Scene Integration

Once the selected 3D model is successfully imported, it is instantiated **twice** in the scene, resulting in two initially identical GameObjects:

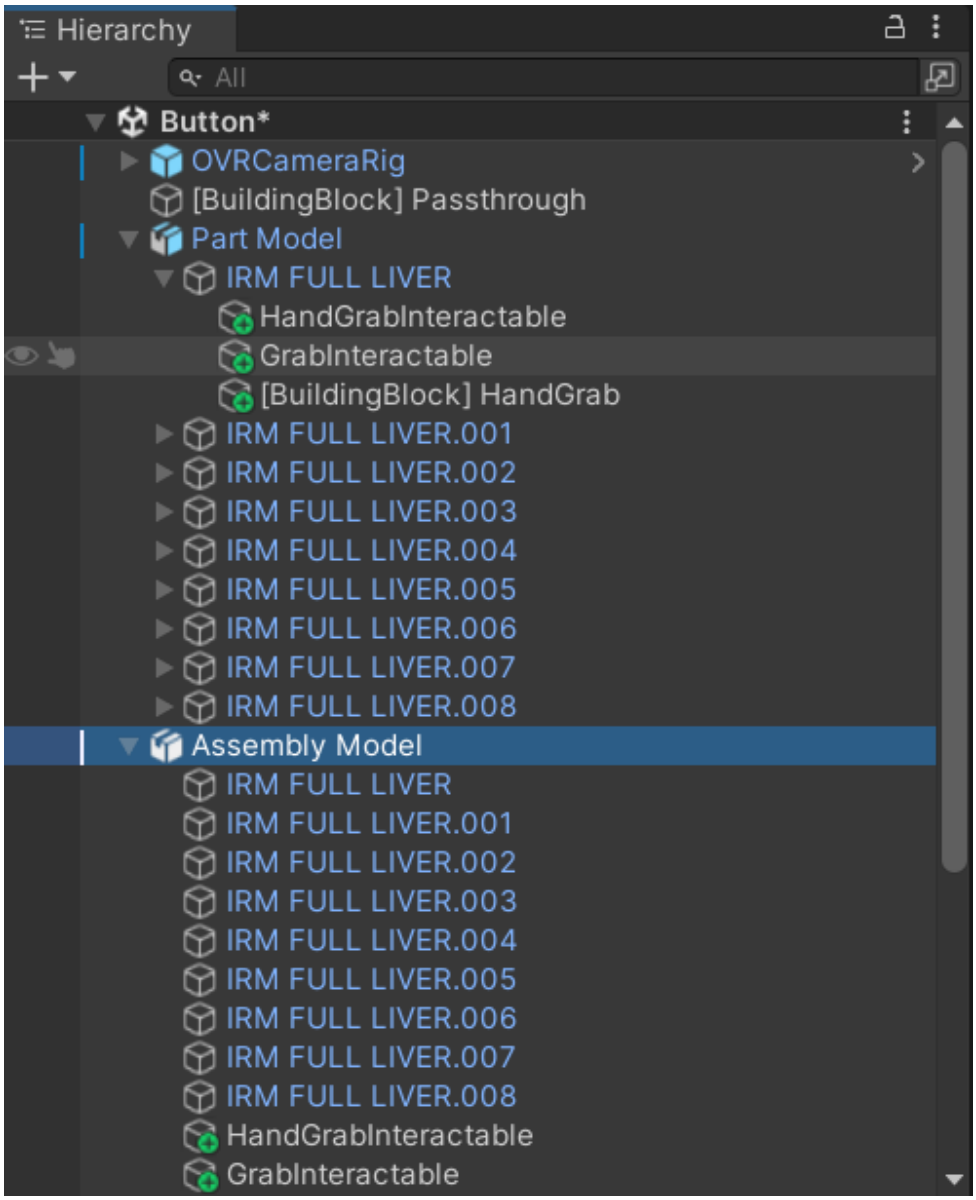


Figure 4.13: At runtime, the uploaded 3D model is instantiated twice: once as `PartModel` and once as `AssemblyModel`. Initially identical, they are configured differently in later stages—`PartModel` will support per-component interaction, while `AssemblyModel` remains a manipulable whole.

- **PartModel**: This GameObject contains a full copy of the imported model and will later support per-part interaction. Each individual component (e.g., anatomical substructure) may become grabbable or selectable in future stages.
- **AssemblyModel**: Also based on the same model, this object is treated as a complete unit. It will be used for global interactions such as full-model manipulation, inspection, or positioning within the MR space.

These GameObjects are instantiated independently, enabling flexible interaction modes while maintaining a consistent reference geometry.

Advantages of This Approach

This runtime-based model upload system introduces multiple benefits:

- **Decoupling content from build**: New models can be tested on-device without recompiling the APK.
- **Scalability**: Supports future features such as dynamic model libraries or real-time cloud-synced content.
- **Clinical relevance**: Facilitates rapid testing and review of anatomical models in surgical planning contexts.

Current Limitations

- Only the first model in the directory is loaded (single-model support).
- The system assumes correctly formatted and preprocessed `.obj` files with appropriate pivot and scale.
- No interaction or animation is yet available; these will be addressed in subsequent stages.

4.5.3 Stage 2: Enabling Grab Interaction on Children of `PartModel`

In this stage, interaction is added to the children of the `PartModel` GameObject. The parent object remains untouched — all logic and components are applied to its individual sub-objects (referred to as "parts").

Each child corresponds to a single mesh representing an anatomical component. These sub-objects are enhanced with components that allow them to be grabbed, moved, and reset independently in Mixed Reality.

Component Configuration Per Child

For each part (child GameObject) under `PartModel`, the following components are added in Unity:

- **Mesh Filter**: Contains the geometry data of the part.
- **Mesh Renderer**: Renders the part using the assigned material.
- **Rigidbody**: Enables physics. It is typically set to `isKinematic = true` to support grabbing without falling.
- **Box Collider**: Allows collision and physical interaction. Adjusted to tightly fit the part's geometry.
- **Grabbable (Script)**: Enables the part to be grabbed using the Meta Interaction SDK. This is part of the [BuildingBlock] Grab system.

- **Grab Free Transformer (Script)**: Handles how the part behaves when grabbed and released, e.g., allowing free movement in space.
- **Reset Pose (Script)**: Provides functionality to return the part to its original position and rotation when we reset or switching between models.

Meta Building Block Integration

The grab system used here relies on Meta's high-level XR interaction building blocks. Specifically:

- **[BuildingBlock] Grab Interaction**: A ready-made prefab or script bundle from the Meta SDK that manages detection of user hands, gesture recognition, grip logic, and object attachment.

Each child receives this component to become individually grabbable in Mixed Reality, using either hand tracking or controller input.

Example Setup

Figure 4.14 shows the typical component configuration applied to each child GameObject:

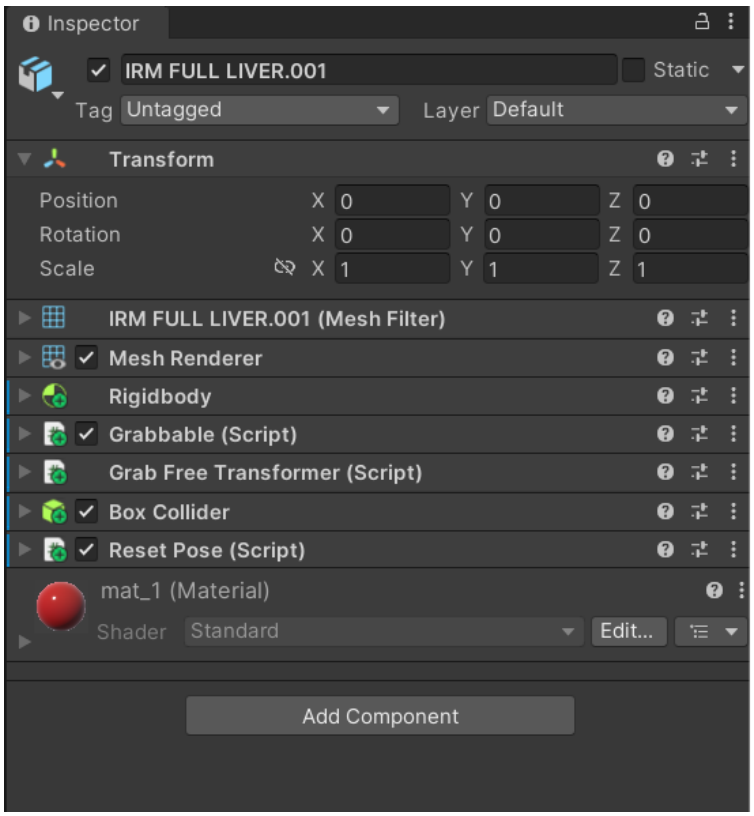


Figure 4.14: Unity Inspector showing all components required for enabling grab interaction on a child of `PartModel`.

Resulting Interaction

Once configured, each part becomes individually manipulable in MR. The user can:

- Grab a part using hand tracking or controller input.
- Move, rotate, or inspect it in isolation.
- Return it to its original pose using the Reset function.

This stage enables immersive interaction with sub-anatomies or modular components, opening the path for future features like snapping, guided disassembly, or procedural learning sequences.

4.5.4 Stage 3: Enabling Grab Interaction on the Assembled Model (`AssemblyModel`)

Following the implementation of part-based interaction on the `PartModel`, the next development stage focuses on the `AssemblyModel`. Unlike the previous configuration, interaction in this case is applied only to the parent GameObject, treating the model as a single unified structure.

Interaction Strategy

The goal is to allow users to grab, move, and manipulate the entire model at once without affecting or interacting with individual sub-parts. This is particularly useful for scenarios involving global repositioning, inspection, or comparative visualization of the full anatomy.

To achieve this, all interaction components are attached directly to the root GameObject `AssemblyModel`, while the children (the individual mesh parts) remain unmodified.

Component Configuration on `AssemblyModel`

The following components are added to the parent object only:

- **Rigidbody**: Enables physics-based manipulation. Set as `isKinematic = true` for controlled behavior.
- **Box Collider** or **Mesh Collider**: Defines the bounds for physical interactions with the model as a whole.
- **Grabbable (Script)**: Makes the model responsive to XR-based grab events.
- **Grab Free Transformer (Script)**: Enables smooth movement and release behavior during interactions.
- **Reset Pose (Script)**: Allows the user to return the model to its original position and orientation.

Applying the Building Block

The same Meta SDK prefab used in the part-level interaction is applied here, but to the parent model instead of individual parts:

- **[BuildingBlock] Grab Interaction**: Attached to the `AssemblyModel` GameObject to support full-model manipulation via hand tracking or controllers.

Resulting Behavior

With this setup, the user can:

- Grab and move the entire model as a single unit.
- Rotate or reposition it spatially within the MR scene.
- Reset its pose to a default anchored location if needed.

Visual Example

The diagram below illustrates the structure of the `AssemblyModel`, where only the parent GameObject contains the interaction components. All children remain passive renderable parts.

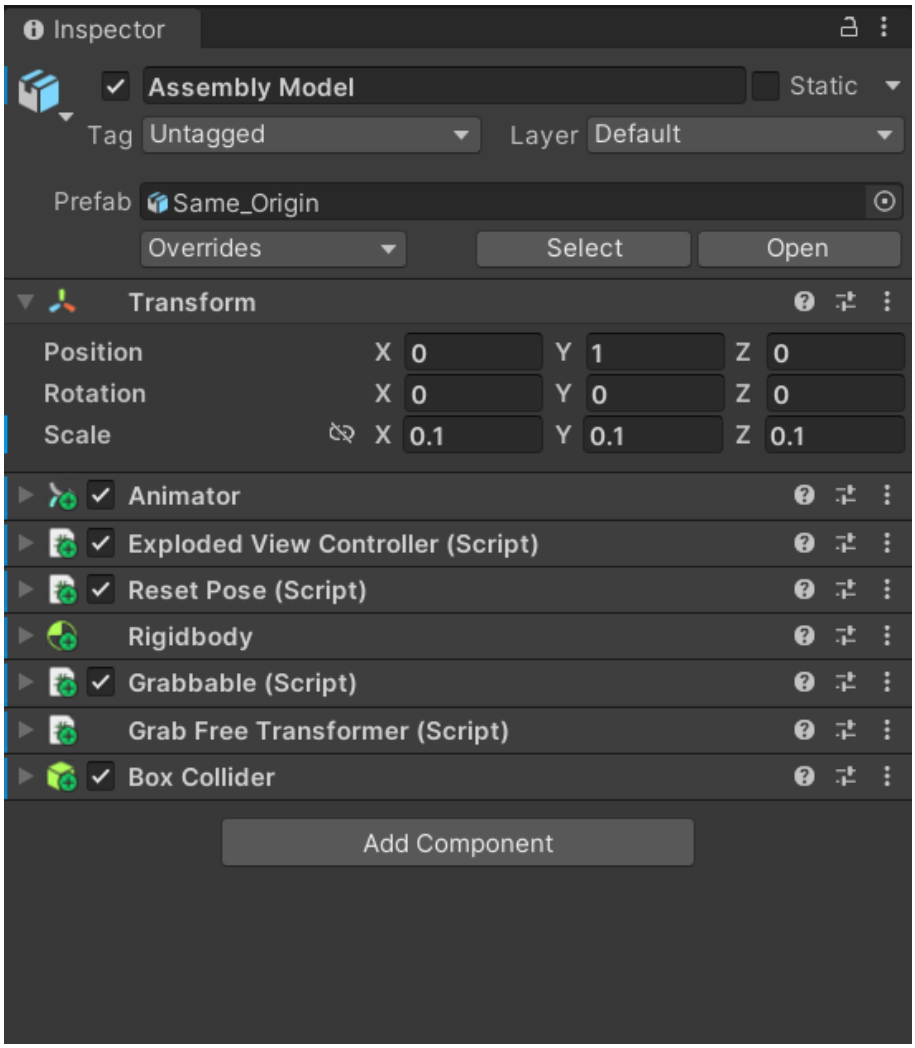
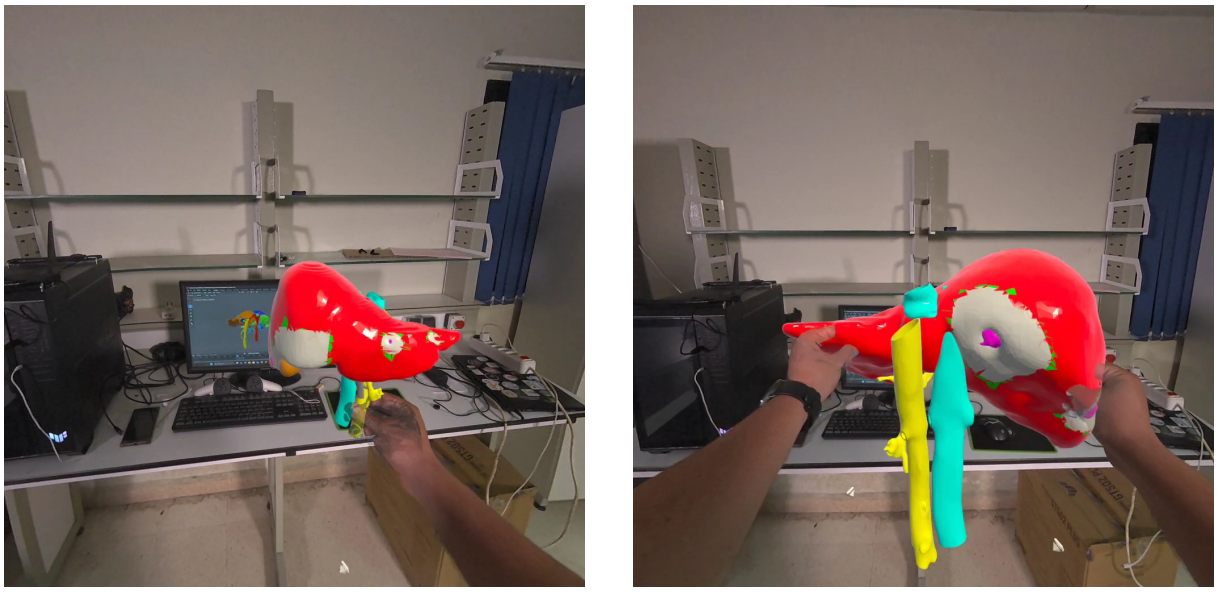


Figure 4.15: Interaction applied at the `AssemblyModel` level. All components such as Rigidbody, Collider, and Grab Interaction are attached to the parent. Children are used only for rendering and are not individually grabbable.

Comparison with `PartModel`

Feature	`PartModel`	`AssemblyModel`
Target	Children (per part)	Parent (whole model)
Grab Behavior	Individual parts	Entire model
Interaction Level	Fine-grained	Global
BuildingBlock Applied To	Each child	Parent object only
Collider Type	Per-part Box Collider	Whole-model Box/Mesh Collider



(a) Grabbing the whole model (b) Scaling the model with two hands

Figure 4.16: Screenshots from Meta Quest showing manipulation of the model: grabbing (left) and scaling (right).

4.5.5 Stage 4: Adding Hand Pose Recognition

In this stage, user interaction is extended through the integration of hand pose recognition using Meta's Interaction SDK. This enables the application to recognize specific hand gestures (e.g., thumbs up) and associate them with triggers, selections, or interaction events in the MR environment.

Pose Definitions

Several custom hand poses are defined and organized under a dedicated GameObject group named `pose_detection`. Each pose is represented as a child GameObject:

- `Left_Hand_Pose`
- `Right_Hand_Pose`
- `thumps_up`
- `thumps_up_left`

- `thumps_up_right`

These GameObjects are empty containers that serve as anchors for pose tracking and recognition components.

Component Setup for Each Pose

Each hand pose GameObject is equipped with a standard set of components from the Meta SDK to enable recognition and event binding:

- **Hand Ref (Script)**: Specifies whether the pose applies to the left or right hand.
- **Shape Recognizer Active State**: To define the hand gesture for triggering the exploded view, we used the `Shape Recognizer Active State` component referencing a left-hand pose and the custom shape `ExplodedView`, which is defined by configuring curl, flexion, and abduction parameters for each finger.

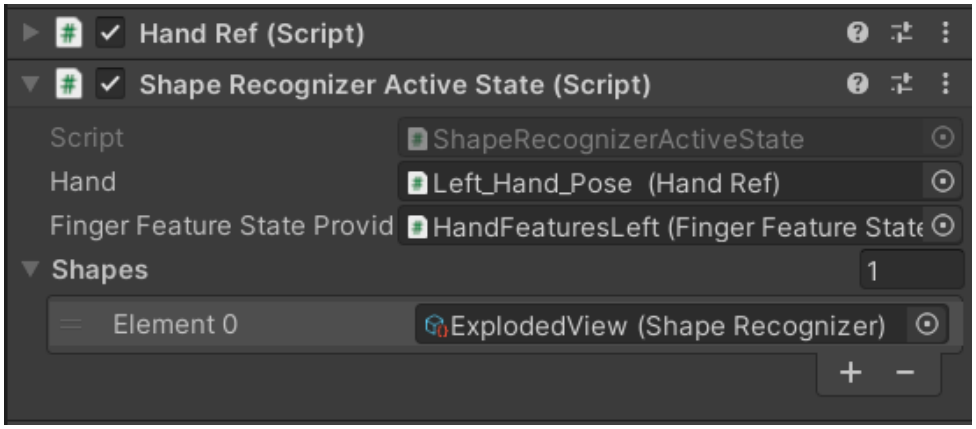


Figure 4.17: Shape Recognizer Active State linked to the `Left_Hand_Pose` and the `ExplodedView` shape.

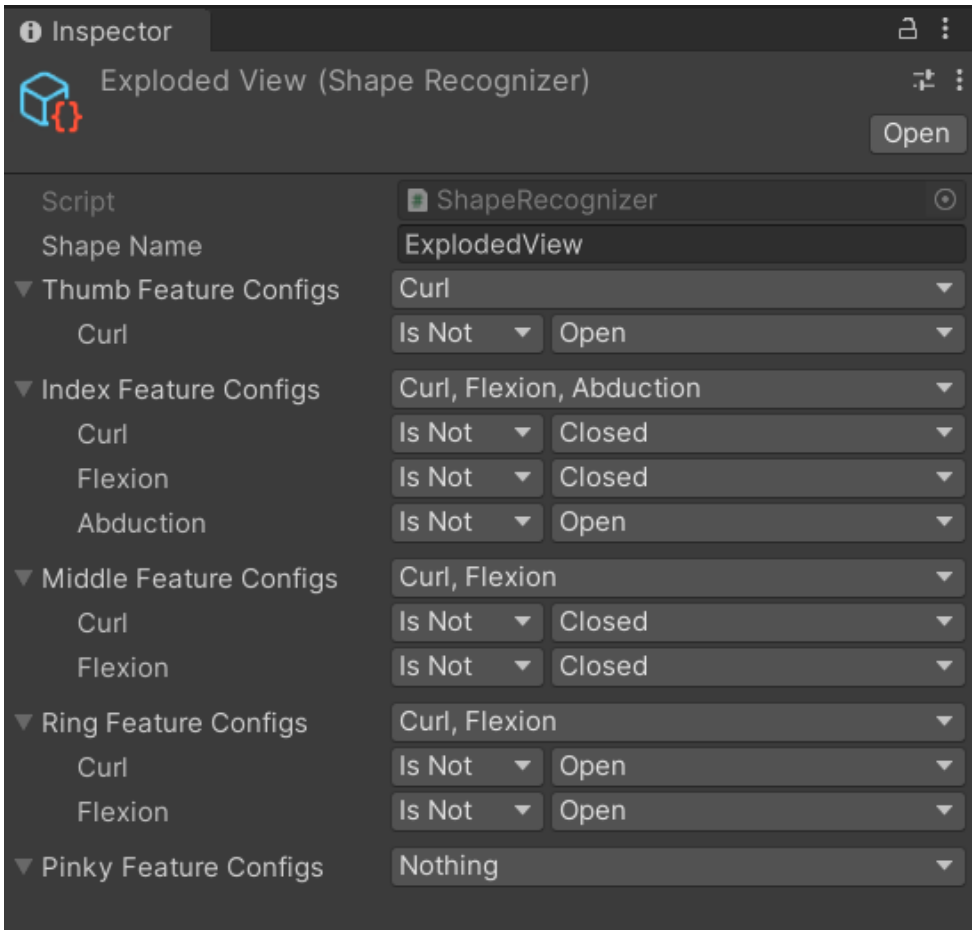


Figure 4.18: Detailed configuration of the `ExplodedView` shape recognizer using per-finger feature states.

- **Transform Recognizer Active State**: Tracks spatial configuration (pose) relative to the camera or hand.

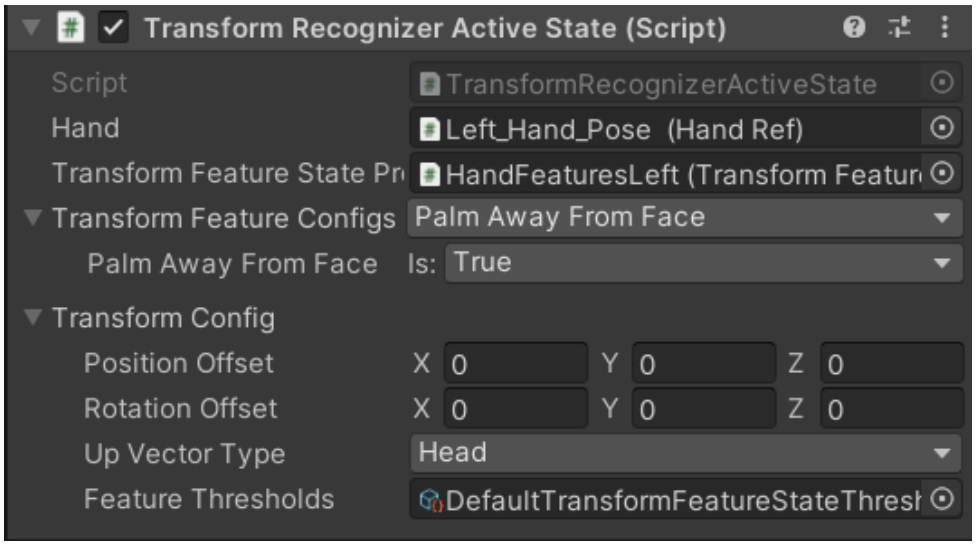


Figure 4.19: Inspector view of the `Transform Recognizer Active State` component used to detect spatial hand poses based on relative transforms.

- **Active State Group**: Groups multiple recognizers together it become true in our state when the two (shape **and** transform)are true .
- **Active State Selector**: Evaluates active states and determines whether the full pose condition is met.
- **Selector Unity Event Wrapper**: Allows UnityEvents to be invoked when the pose is detected.

Visual Reference

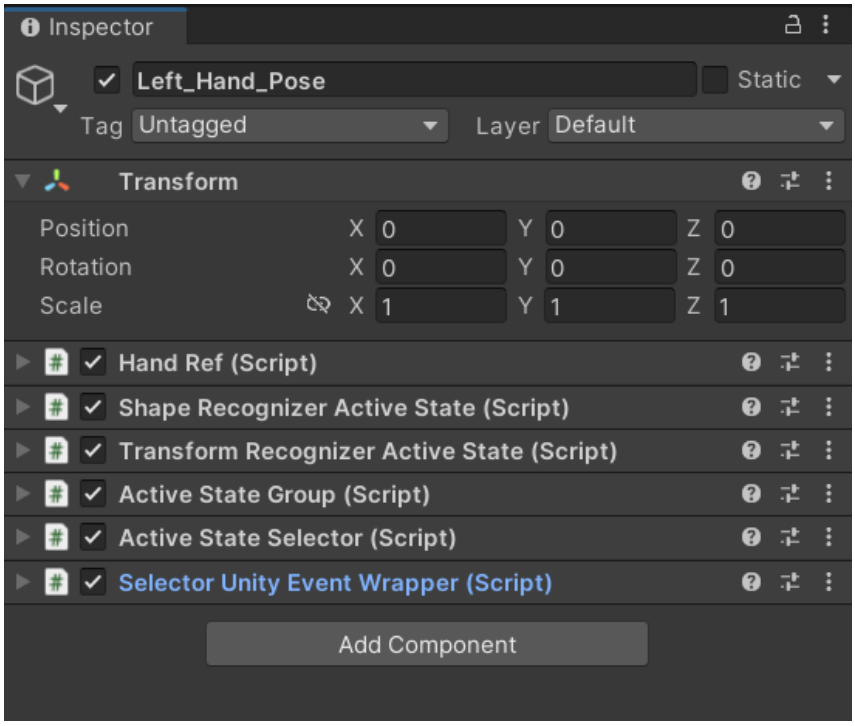
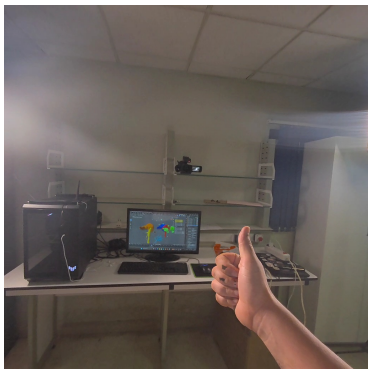


Figure 4.20: Inspector view of a single hand pose GameObject, configured with all required components for shape and transform recognition.

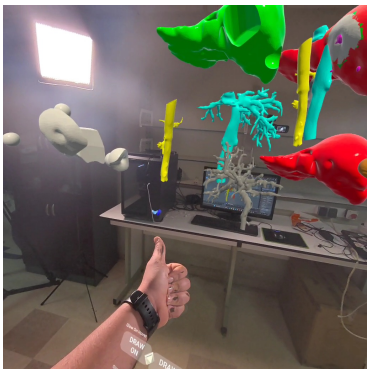
Runtime Behavior

Once active in the scene, the pose detection system continuously monitors the user's hand input. When a predefined pose is recognized:

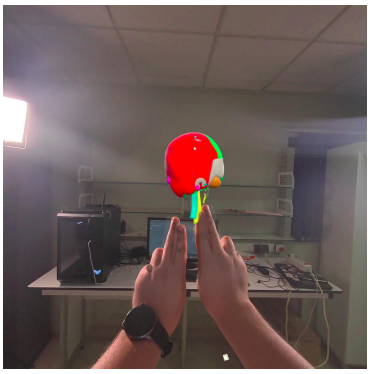
- The corresponding `Active State Selector` becomes `true`.
- The `Selector Unity Event Wrapper` emits an event.
- This event can be used to trigger logic such as toggling object visibility, snapping parts, changing modes, etc.



(a) `ThumbsUpright` pose recognized



(b) `ThumbsUpLeft` pose recognition.



(c) `ExplodedView` gesture detected.

Figure 4.21: Screenshots from the Meta Quest 3 showing real-time recognition of three hand poses used in the application.

4.5.6 Stage 5: Switching Interaction Modes Using Hand Poses

In this stage, specific hand gestures are used to switch between different interaction modes in the application — namely, toggling between the `PartModel` and `AssemblyModel`. Two hand poses are defined: `thumbs_up_left` and `thumbs_up_right`, each assigned to a different functional state.

Pose Recognition for Mode Switching

Each pose is implemented using the same Meta SDK pattern:

- A `Shape Recognizer Active State` tied to a hand reference (left or right).
- A `Selector Unity Event Wrapper` to listen for pose activation.
- A UnityEvent that activates the relevant GameObject and resets the other.

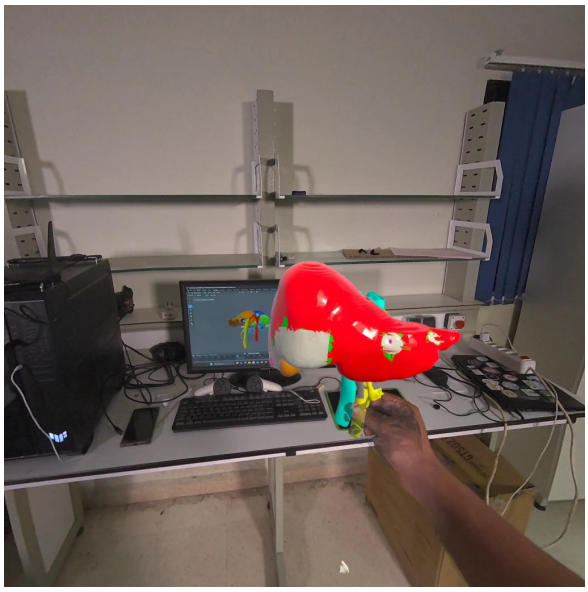
`thumbs_up_right` is used to activate and reset the `PartModel`, enabling per-part interaction. Conversely, `thumbs_up_left` is linked to the activation of the `AssemblyModel`, which treats the model as a unified whole.

This switching mechanism ensures only one model is active and interactive at a time, avoiding conflicts in physics or user experience.

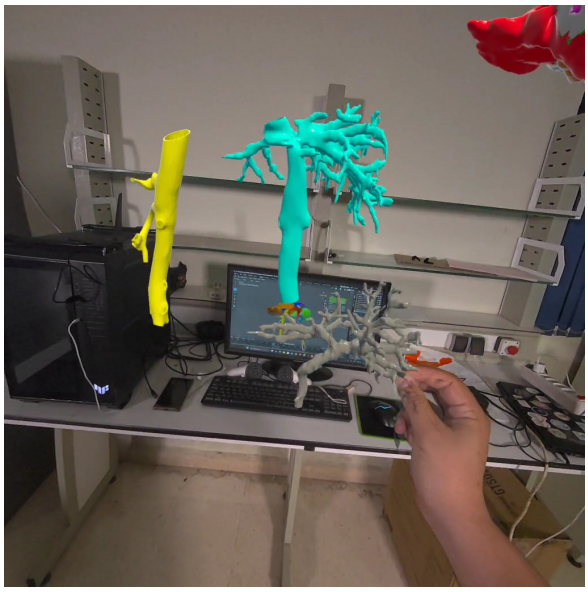
Runtime Behavior

- When the user performs the `thumbs_up_right` gesture, the system enables part-level manipulation and disables the assembled view.
- When the user performs the `thumbs_up_left` gesture, the whole model becomes active and movable, and part-based interaction is hidden.

This logic is especially useful in educational or surgical planning contexts where the user might toggle between detailed anatomical exploration and a global model view with a simple, intuitive gesture.



(a) Grabbing the full model



(b) Grabbing a single part

Figure 4.22: Meta Quest screenshots comparing whole-model interaction (left) with per-part interaction (right).

4.5.7 Stage 6: Triggering Animations with Recognized Hand Poses

This stage builds on the hand pose recognition system by linking gestures to animations through Unity's Animator Controller. The goal is to allow gestures — such as `ExplodedView` — to trigger part separation or movement animations in Mixed Reality.

Animator Setup

An Animator Controller is assigned to the animated model (typically `AssemblyModel`). It contains a state machine where custom animation clips (e.g., `Scene`) are linked to states and controlled via runtime parameters such as `AnimationPosition`.

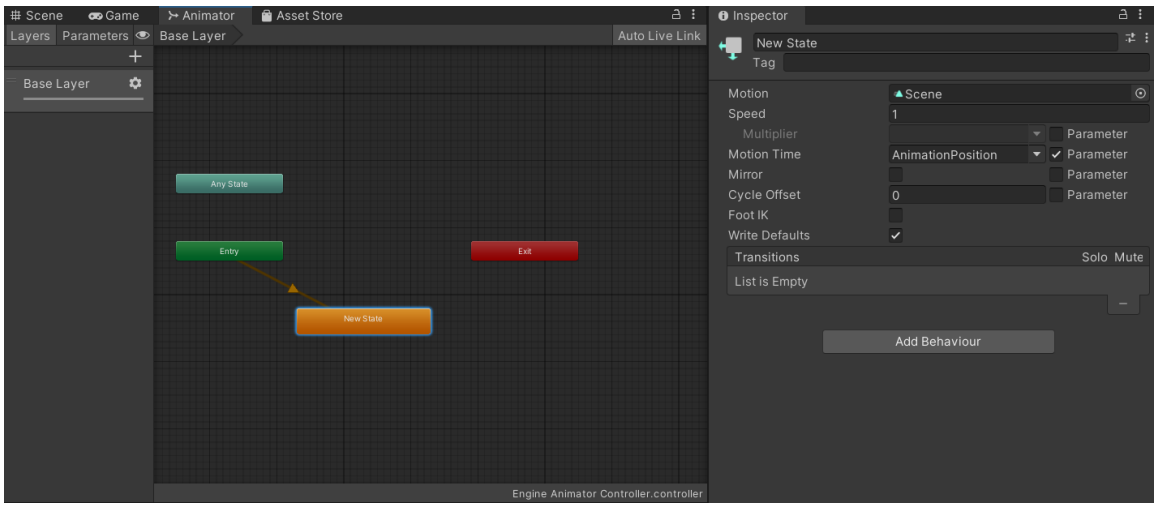


Figure 4.23: Animator Controller with a custom clip triggered by a gesture using parameter-based control.

Gesture-Driven Triggering

The custom pose `ExplodedView` is recognized through the `Shape Recognizer Active State`. When matched:

- The `Selector Unity Event Wrapper` emits an event.
- This event sets the Animator parameter (e.g., `AnimationPosition` = 1).
- The Animator transitions to the `Scene` clip and plays the explosion animation.

Resulting Workflow

1. User performs the `ExplodedView` hand pose.
2. Pose is matched and event is triggered.
3. Animator parameter is updated.
4. The model plays an animation separating its parts or changing layout.

This gesture-to-animation system introduces a powerful, immersive control method for surgical planning, simulation, or learning sequences — allowing intuitive visual transitions controlled entirely through hand poses.

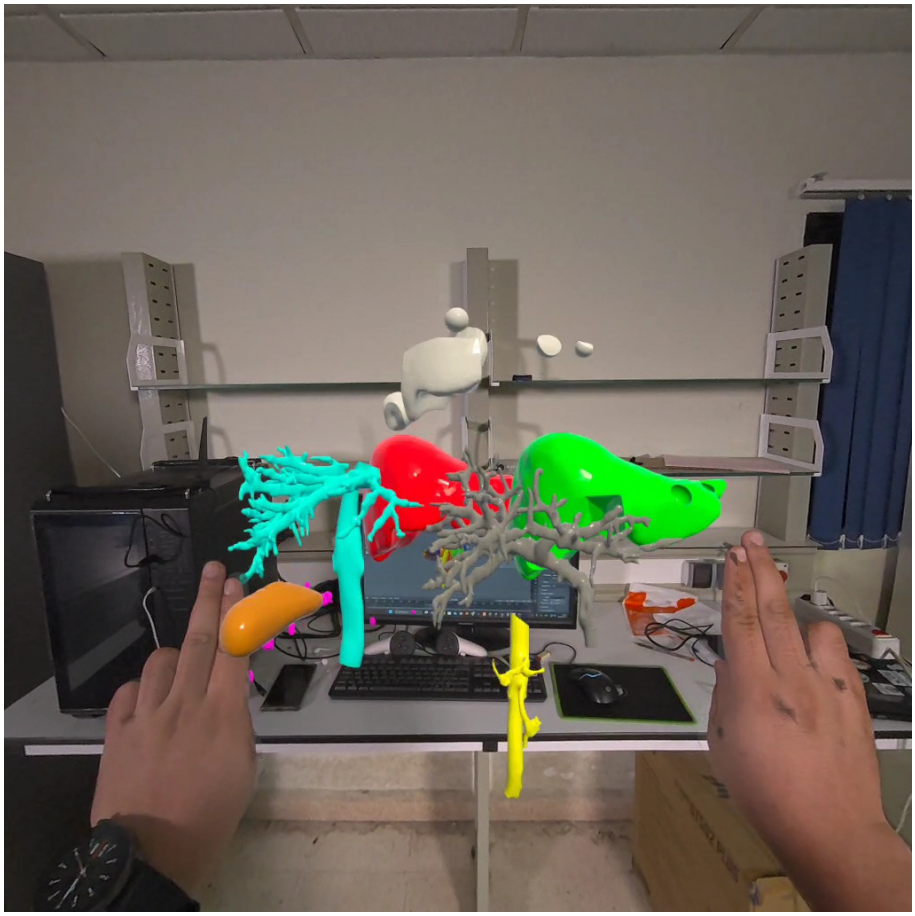


Figure 4.24: The animation triggered by the `ExplodedView` pose, showing part separation in Mixed Reality.

4.5.8 Stage 7: Adding View Mode Controls via UI Buttons

To support clinical exploration and surgical planning, this stage introduces a user interface element in the form of a button-based table. Each button represents a distinct "view mode" that selectively reveals or hides components of the anatomical model. This allows users to visually explore tumors, vessels, and regions of interest with precision and clarity.

User Interface Structure

The interface is implemented as a virtual panel in the MR environment, visible in-world or attached to the user's hand. It contains three primary buttons, each associated with a UnityEvent that controls specific visibility logic in the model:

- **Deep View**: Deactivates the outer liver mesh to expose tumors and vasculature embedded within the `AssemblyModel`. This is used for internal inspection without disassembly.

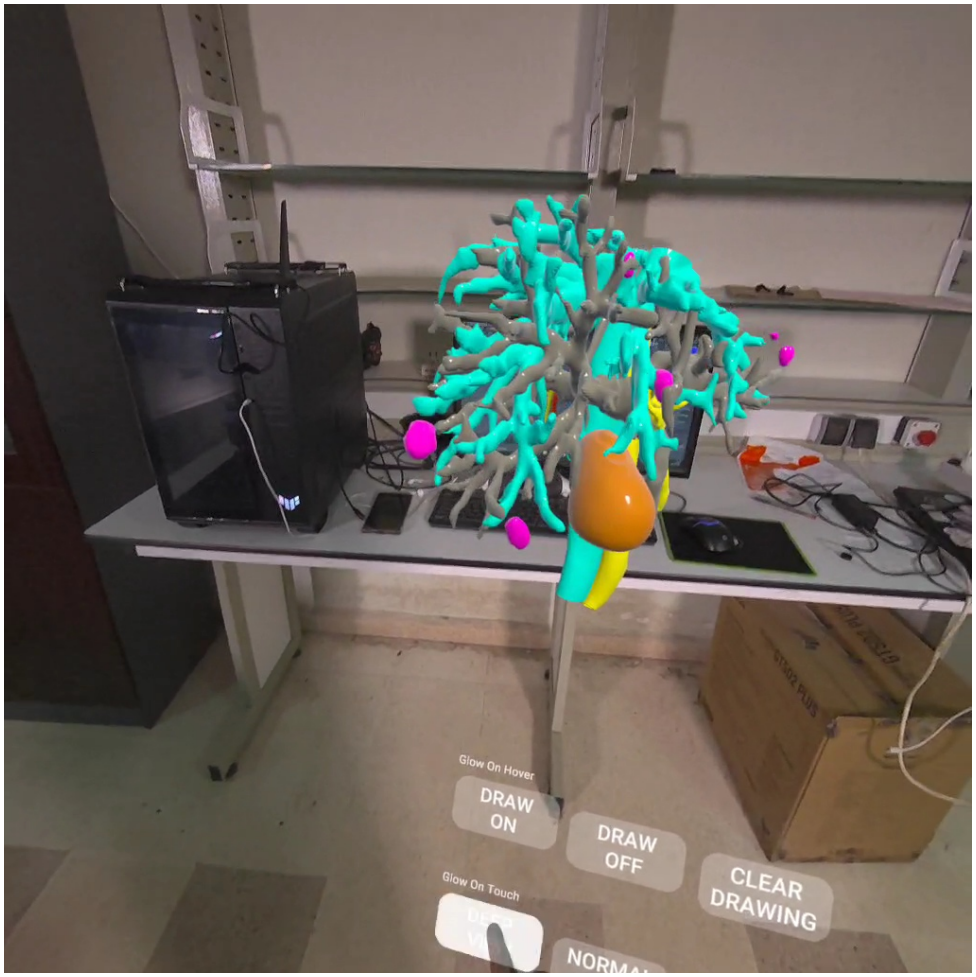


Figure 4.25: Deep View mode: The outer liver body is hidden to expose internal tumors and vessels.

- **Inside View**: Hides the healthy regions of the liver while keeping only the part that the surgeon plans to resect. This focused view emphasizes the surgical target area and highlights potential risk zones.

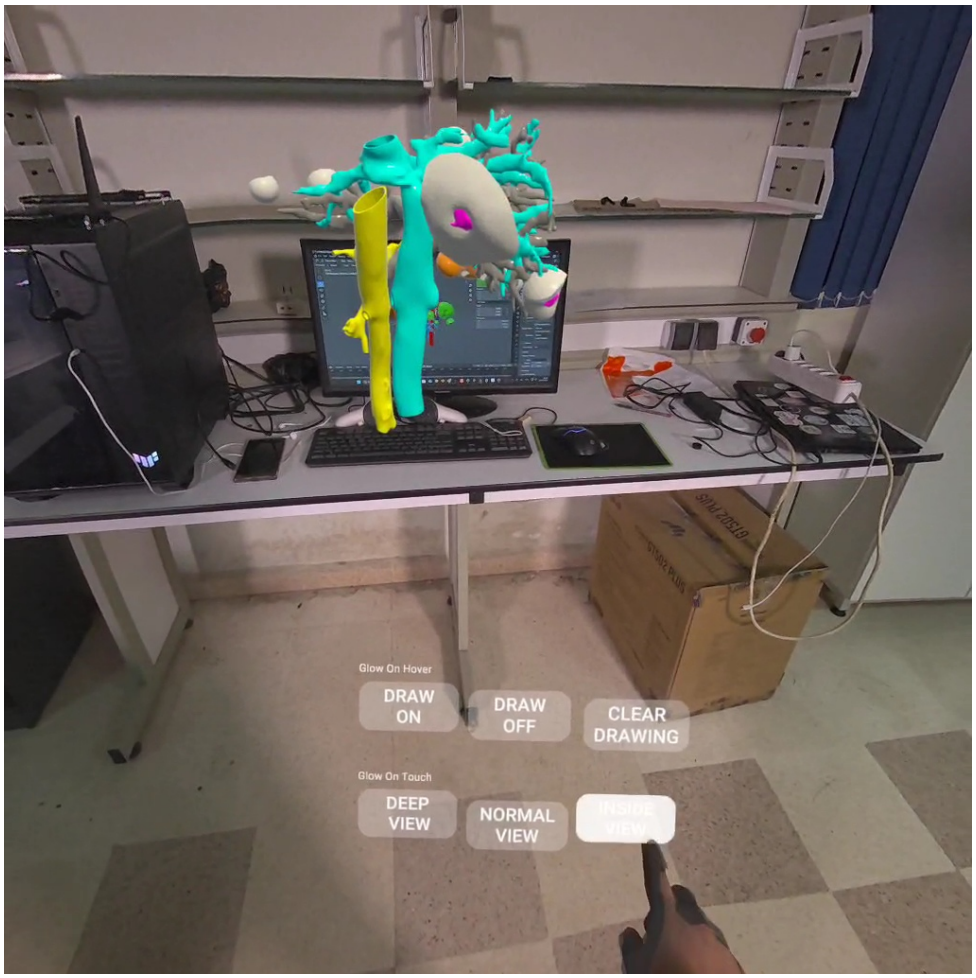


Figure 4.26: Inside View mode: The liver is mostly hidden except for the resection zone — the part intended to be surgically removed.

- **Normal View**: Restores all components of the liver model to their original state, re-enabling the full anatomical surface.

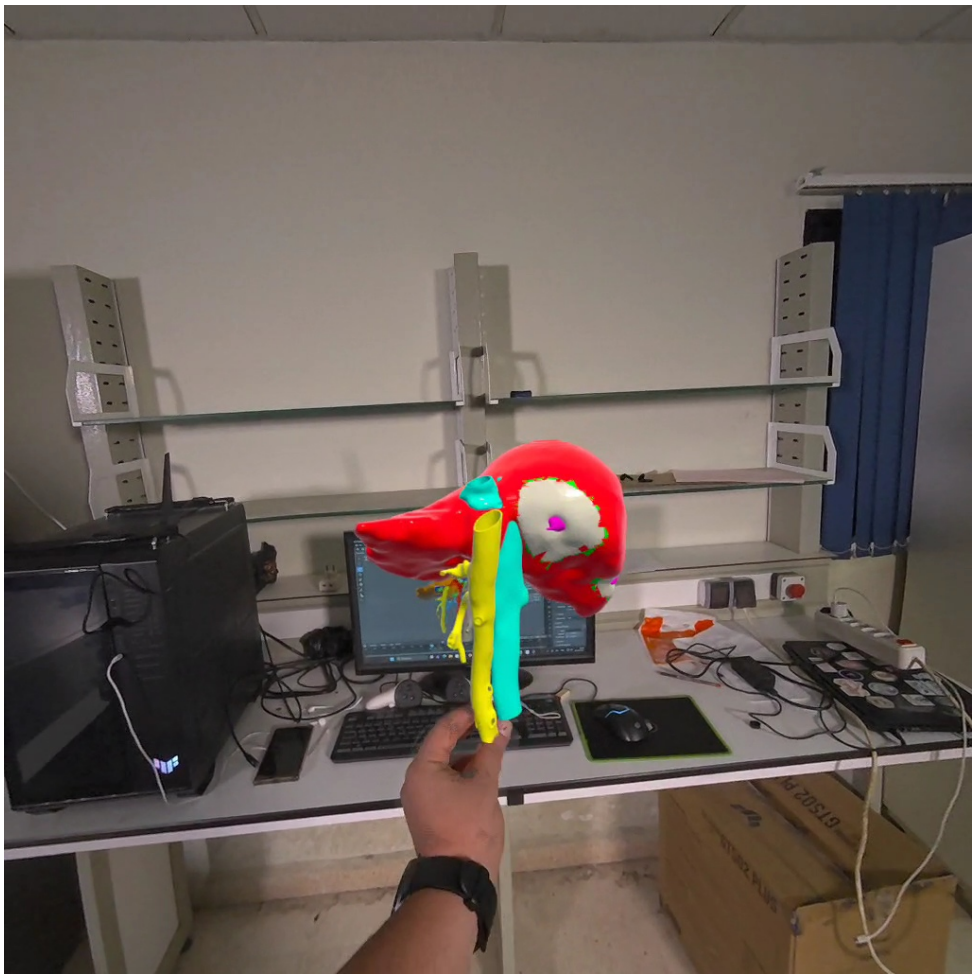


Figure 4.27: Normal View mode: The entire liver anatomy is visible in its original, preoperative form.

Each view is toggled programmatically via the button's UnityEvent trigger, which calls a visibility management script that shows or hides specific GameObjects.

Interaction Logic

The logic behind each view mode is modular. Specific liver parts are grouped and tagged in Unity, allowing batch activation or deactivation based on the selected view. This ensures consistency and simplifies logic for reset or switching between modes.

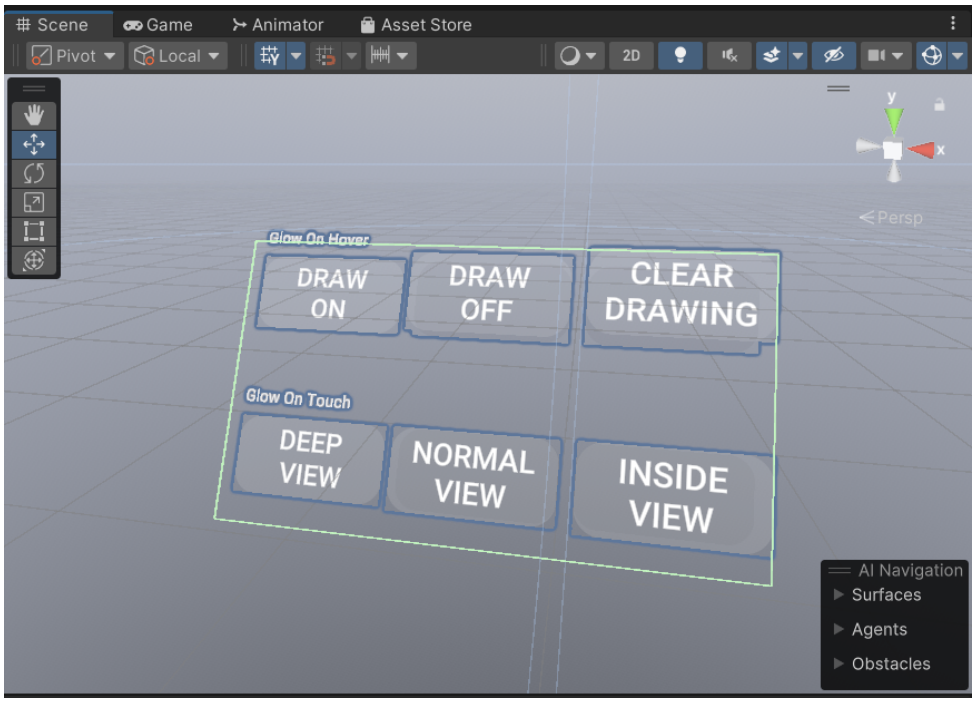


Figure 4.28: View mode panel in MR. Users can switch between internal, external, and surgical views of the liver using simple hand-based UI interaction.

Future UI Expansion

Three additional buttons have been defined in the same UI panel but are not yet fully implemented:

- **Activate Drawing**: Enables an MR drawing or markup tool for sketching on or around the liver.
- **Deactivate Drawing**: Disables the drawing tool to return to navigation mode.
- **Clear Drawing**: Removes all current annotations or sketches made during the session.

These functions are scheduled to be implemented in the next development stage, expanding the system's utility for education, planning, and intraoperative guidance.

4.5.9 Stage 8: Freehand Drawing and Annotation in Mixed Reality

This stage introduces an in-situ annotation feature, enabling users to draw freely in 3D space using natural hand gestures. The system leverages Meta Quest's hand tracking to detect index finger pinching, allowing clinicians, educators, or engineers to sketch in mid-air, annotate surgical zones, or highlight anatomical landmarks in real-time.

Drawing Interaction Logic

Drawing is initiated when the system detects a sustained pinch gesture between the user's thumb and index finger. Once the pinch strength crosses a predefined threshold, the

system begins rendering a 3D line at the index fingertip's position, continuously updating as the user moves their hand.

To avoid noisy inputs or jitter, a minimum distance threshold is enforced between each point added to the line. This ensures smooth strokes while maintaining fidelity with the user's motion.

When the pinch is released, the current stroke is considered finished. A new stroke begins upon the next pinch. Each line is rendered with a consistent width, customizable color, and is recorded as an independent object within the scene, allowing future interaction or deletion.

Multi-Stroke and Runtime Customization

All strokes are managed independently to allow for:

- **Multiple annotations** within a single session. - Runtime updates of **line thickness**, **color**, and **drawing precision** (distance threshold between points). - Clearing all strokes via a single interaction, such as pressing a UI button.

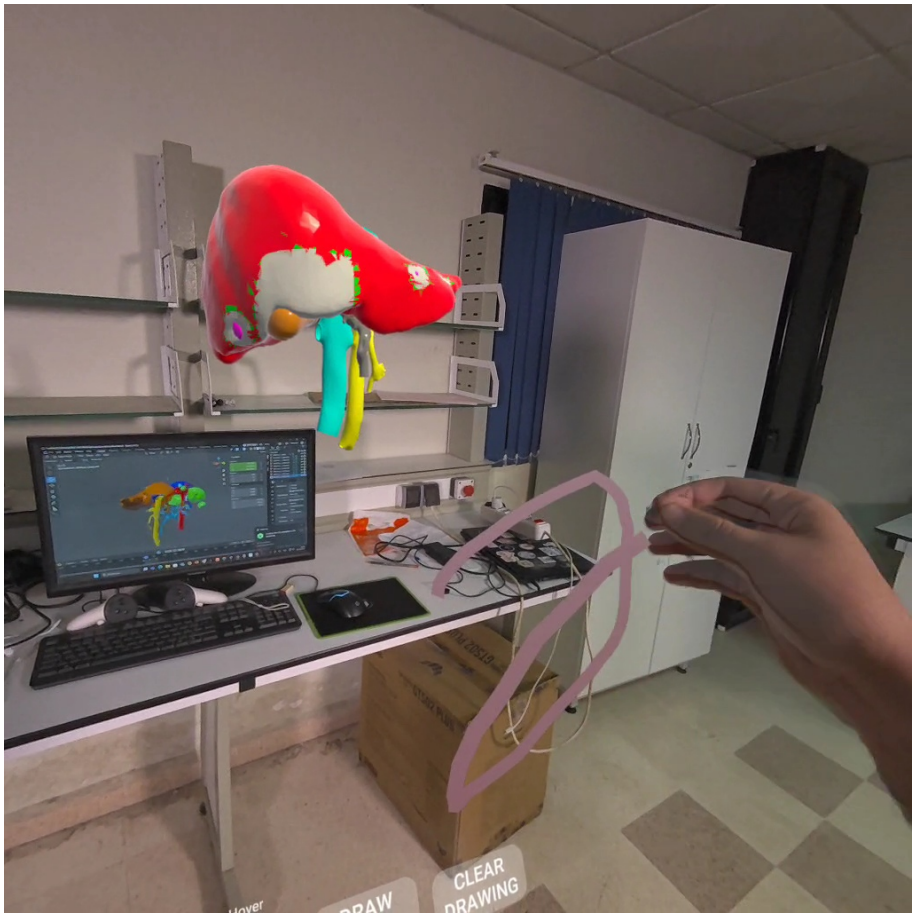


Figure 4.29: Drawing in MR using hand tracking and index pinch. The user traces contours or marks surgical regions directly in 3D space.

Drawing Tool Control Buttons

As prepared in Stage 7, three UI buttons are now fully functional to control the drawing tool:

- **Activate Drawing**: Enables line rendering and begins monitoring hand pinch input.
- **Deactivate Drawing**: Halts all drawing activity without erasing existing sketches.
- **Clear Drawing**: Deletes all current strokes from the scene to reset the canvas.

This feature significantly enhances the interactivity and educational value of the MR environment, empowering users to visually communicate decisions, risks, or observations within the surgical or training scenario.

4.5.10 Stage 9: Measurement Tool Integration with VR Controllers

In the final stage, the Mixed Reality application was equipped with a measurement tool to allow precise spatial analysis of anatomical structures or surgical targets. This feature is designed to simulate the use of a virtual measuring tape within the 3D environment using Meta Quest's hand controllers.

Controller-Based Measurement Workflow

Users initiate the measurement process by pressing a designated button on either the left or right Meta Quest controller. Upon activation:

- A virtual line is instantiated at the controller's current position.
- As the controller moves, the end point of the line dynamically updates to reflect its trajectory in 3D space.
- Real-time distance calculations are performed between the two endpoints of the line.
- The calculated distance is displayed in centimeters (`cm`) near the user's controller during the interaction.
- Once the user releases the button, the measurement line is fixed in space and the text label snaps to the center of the line for clarity.

Measurement Visualization and Usability

Each measurement line is visualized using a thin, colored 3D line rendered in world space. The distance label is implemented using a floating UI text element that always faces the user's view (billboard behavior). This ensures readability from all angles, even within complex anatomical scenes.

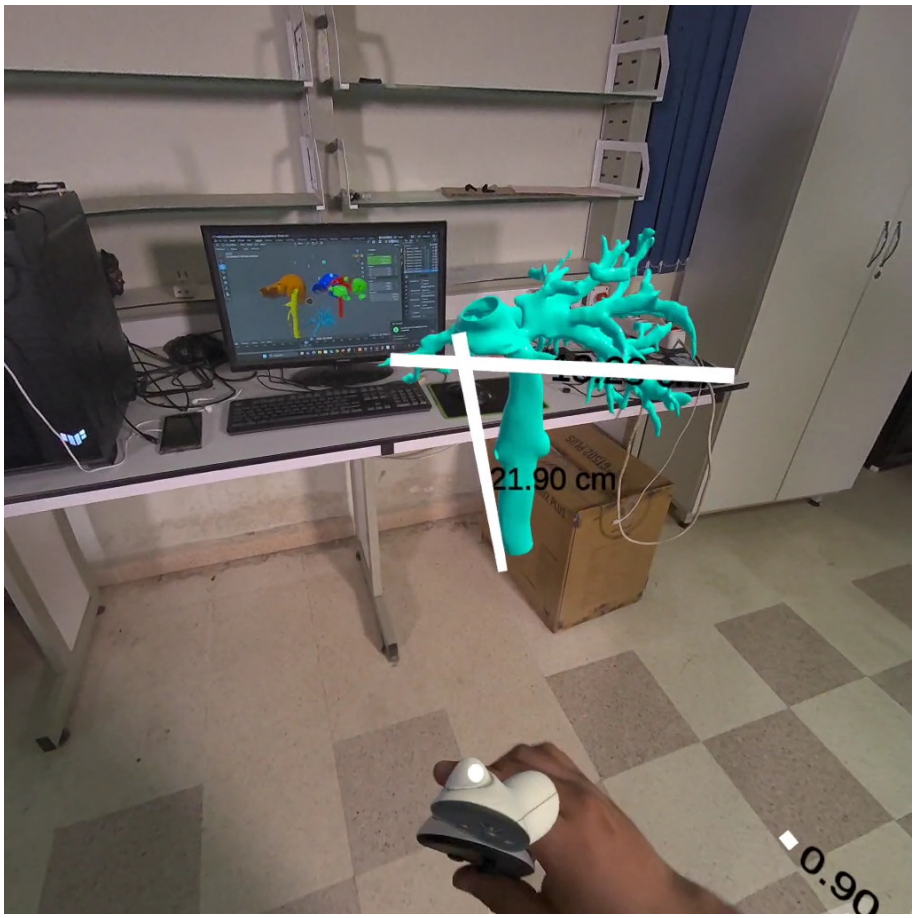


Figure 4.30: Measurement tool in action: a virtual line is drawn between two points, with the measured distance displayed in centimeters.

Educational and Clinical Impact

This functionality adds measurable value for:

- **Preoperative planning**, allowing users to measure tumor sizes or resection margins.
- **Training scenarios**, where learners can compare anatomical dimensions.
- **Spatial awareness**, improving depth perception and understanding of organ layout.

As with all other interactive elements, the measurement tool is modular and supports multiple annotations within a single session.

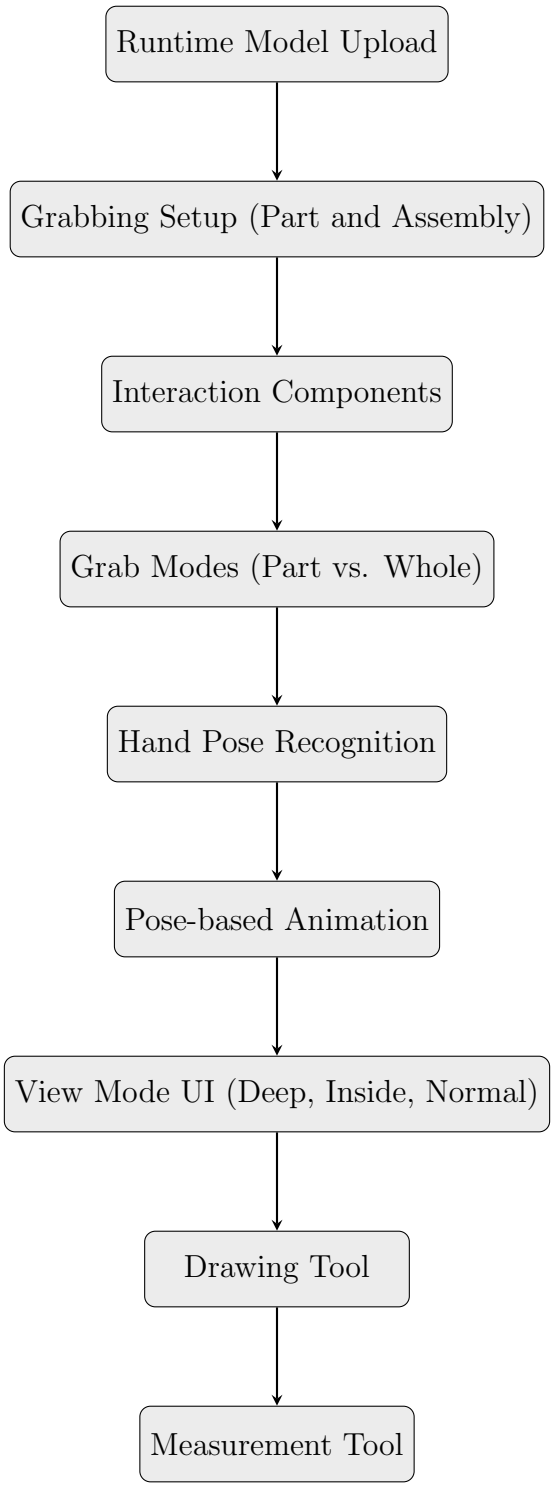


Figure 4.31: Development Pipeline: 9 Stages of Mixed Reality Interaction on Meta Quest3

4.5.11 Integrating Multiplayer into a VR APK with Unity

Overview

To enable multiplayer features in a VR application built with Unity, we leveraged the **Unity VR Multiplayer Template**. This template provides a complete framework that integrates networking, player avatars, and interaction systems using **Unity Netcode for GameObjects** and **Unity Cloud Multiplayer Services** such as Authentication, Lobby, Relay, and Vivox (voice chat). The process is compatible with XR platforms like Meta Quest and OpenXR-compliant devices.

Step-by-Step Integration Workflow

1. **Create a Project from the VR Multiplayer Template**
   In Unity Hub, choose *New Project ¿ VR Multiplayer*, and ensure that the option **Connect to Unity Cloud** is checked to automatically set up services.

2. **Configure XR Settings and Platform Build Target**
   In `File > Build Settings`, select the appropriate platform:

   - **Android** for Meta Quest – enables OpenXR settings with Meta Quest profile.
   - **PC** for desktop VR – enables Valve Index, Vive, or Oculus profiles.

3. **Set Up Multiplayer Services**

   - Go to `Window > Unity Services > Cloud Project Settings`.
   - Enable **Authentication**, **Lobby**, and **Relay**.
   - Optionally enable **Vivox** for in-game voice communication.

4. **Use the Sample Scene as a Base**
   The provided `SampleScene` includes:

   - XR Interaction Setup (Camera, Hands, Controllers)
   - Networked Player Avatars
   - Network Manager
   - Spatial UI (keyboard, menus)

   You can extend this scene or remove unused assets.

5. **Network Configuration**

   - The `NetworkManager` component handles connection setup.
   - Use the `XRI Network Game Manager` prefab to link to Unity Cloud services.
   - Instantiate player avatars with `XRI Network Player Avatar` prefab.

6. **Build the APK**

   - Connect your Meta Quest headset via USB and enable Developer Mode.
   - In Build Settings, choose **Android** and click `Build and Run`.
   - Test the APK on-device to verify multiplayer syncing.

Testing Multiplayer Locally

- Use **Multiplayer Play Mode** (MPPM) to simulate multiple players within the Unity Editor.
- Install from `Package Manager > Multiplayer Play Mode`.
- Ensure each virtual player has a unique identifier (`-playerArg:<UNIQUE_ID>`).

Notes and Recommendations

- Avoid using `ParrelSync` alongside Multiplayer Play Mode due to compatibility issues.
- Ensure all prefabs that are synchronized (avatars, objects, UI) include the required `NetworkBehaviour` components.
- For long-term deployment, consider using custom player authentication instead of anonymous login.

4.5.12 Conclusion

Using Unity's VR Multiplayer Template significantly simplifies the process of adding multiplayer support to a VR APK. With built-in components for networking, player representation, and UI, developers can rapidly prototype and deploy immersive collaborative VR experiences on devices like Meta Quest.

Conclusion and Perspectives

Summary of Contributions

This project developed and validated an innovative integrated system for liver surgical planning, combining artificial intelligence and collaborative mixed reality. The key contributions of the work are as follows:

- **Automated Segmentation and 3D Reconstruction:** We implemented an AI-based pipeline capable of segmenting key liver structures — including the liver parenchyma, tumors, and vascular networks. Our approach leverages state-of-the-art deep learning models widely used in medical image analysis. The resulting segmentations enabled high-quality 3D reconstructions, achieving competitive results compared to recent benchmarks in the literature.
- **Immersive Mixed Reality Visualization:** A mixed reality application was developed on the Meta Quest 3 headset, allowing fluid, real-time, and interactive rendering of 3D anatomical models. This significantly enhances spatial understanding when compared to conventional 2D planning approaches.
- **Intuitive Interactions and Clinical Tools:** Gesture-based interactions, combined with built-in measurement and annotation tools, provide a user-friendly and clinically relevant interface tailored for surgical planning tasks.
- **Local Collaboration Capability:** The system enables multiple users to engage in co-visualization and interaction in a shared MR environment, with minimal synchronization delay. This supports collaborative and multidisciplinary planning.

Clinical Potential of the Proposed System

The developed system demonstrates considerable potential for clinical application in liver surgery. By enabling surgeons to interact with accurate and immersive 3D anatomical representations, it aims to:

- Enhance the accuracy and confidence in surgical planning.
- Reduce intraoperative errors and complications.
- Improve overall patient outcomes through better preoperative assessment.
- Foster more effective communication among surgical team members.

To validate and implement this system clinically, a phased roadmap is proposed:

1. **Phase 1 – Pilot Trial:** A preliminary clinical study involving approximately 20 surgeons in a hospital setting will be conducted to evaluate usability, acceptability, and initial effects on planning confidence. Institutional Review Board (IRB) approval will be sought.

2. **Phase 2 – Workflow Integration:** Contingent on favorable pilot results, the system will be integrated into the hospital's preoperative planning workflow, with close coordination between surgical teams and radiologists.

3. **Phase 3 – Clinical Impact Study:** A comprehensive prospective study will measure clinical impact using metrics such as operative time reduction, improved resection margins, decreased postoperative complications, and long-term patient outcomes.

Future Work and Recommendations

Several research directions and enhancements are proposed to extend the system's capabilities:

- **Enhanced Remote Collaboration:** Extend the system beyond local collaboration to support real-time remote surgical planning over wide-area networks. This would allow geographically distributed teams to interact simultaneously with 3D models. Key challenges such as latency, synchronization, and network reliability must be addressed to ensure seamless experiences.

- **Haptic Feedback Integration:** Integrate haptic devices (e.g., haptic gloves) to provide tactile feedback when interacting with virtual anatomical models. Simulating the texture and stiffness of tissues would significantly improve immersion and realism in preoperative planning, especially when based on real patient data collected in clinical settings.

- **Extension to Other Organs:** Generalize the system to support the segmentation, reconstruction, and interactive planning of surgeries involving other organs such as the brain, heart, or kidneys. This would broaden its clinical applicability and usefulness across multiple surgical specialties.

- **Real Clinical Data Validation:** Enhance the system by validating and training it using real hospital data from diverse patient populations. This would improve model robustness, segmentation accuracy, and overall clinical relevance.

- **Headset-Free Mixed Reality Using Depth Cameras:** Develop a mixed reality experience without the use of a headset, leveraging depth-sensing cameras such as Kinect or stereo vision systems. This setup aims to deliver an immersive MR environment through spatial tracking and gesture recognition, enabling interaction with 3D anatomical models directly in the physical space—without requiring a head-mounted display.

These future directions aim to evolve the system into a more comprehensive and intelligent tool, aligning with the complex requirements of next-generation surgical planning and execution.

Bibliography

[1] *An attempt to evaluate the use of mixed reality in surgically treated pediatric oncology patients*. PubMed Central. Accessed: 2025-06-08. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC12064705/.

[2] Jeroen Bertels et al. *Optimizing the Dice Score and Jaccard Index for Medical Image Segmentation: Theory & Practice*. https://arxiv.org/pdf/1911.01685. Accessed: 2025-06-16. 2019.

[3] Patrick Bilic et al. "The liver tumor segmentation benchmark (LiTS)". In: *arXiv preprint arXiv:1901.04056*. 2019.

[4] Sijing Cai et al. "Dense-UNet: a novel multiphoton in vivo cellular image segmentation model based on a convolutional neural network". In: *Quantitative Imaging in Medicine and Surgery*. Accessed: 2025-06-14. AME Publishing Company. 2020. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC7276369/pdf/qims-10-06-1275.pdf.

[5] Özgun Çiçek et al. "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Accessed: 2025-06-14. Springer. 2016. URL: https://arxiv.org/pdf/1606.06650.

[6] Claude Couinaud. *Le foie: études anatomiques et chirurgicales*. French. Paris: Masson, 1957.

[7] 3D-IRCADb-01 database. *3D-IRCADb-01 - 3D Image Reconstruction for Comparison of Algorithm Database*. https://www.ircad.fr/research/3dircadb/. Accessed: 2025-06-18. 2009.

[8] "Detection of Liver Tumour Using Deep Learning Based Segmentation with Coot Extreme Learning Model". In: *ResearchGate* (2023). Accessed: 2025-05-22. URL: https://www.researchgate.net/publication/369083664_Detection_of_Liver_Tumour_Using_Deep_Learning_Based_Segmentation_with_Coot_Extreme_Learning_Model.

[9] Meta Developers. *Use ADB with Meta Quest Devices*. Accessed: 2025-06-12. 2025. URL: https://developers.meta.com/horizon/documentation/native/android/ts-adb/#mobile-android-debug-intro.

[10] Alejandro F. Frangi et al. "Multiscale vessel enhancement filtering". In: *Medical Image Computing and Computer-Assisted Intervention — MICCAI'98*. Ed. by William M. Wells, Alan Colchester, and Scott Delp. Vol. 1496. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1998, pp. 130–137. DOI: 10.1007/BFb0056195.

[11] John E. Hall. *Guyton and Hall Textbook of Medical Physiology*. 14th ed. Elsevier, 2020.

[12] "Hepatic vessel segmentation based on 3D swin-transformer with inductive biased multi-head self-attention". In: *PubMed Central* (2023). Accessed: 2025-05-22. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC10329304/.

[13] *HoloLens platform for healthcare professionals simulation training, teaching, and its urological applications: an up-to-date review*. ResearchGate. Accessed: 2025-06-08. URL: https://www.researchgate.net/publication/386567832_HoloLens_platform_for_healthcare_professionals_simulation_training_teaching_and_its_urological_applications_an_up-to-date_review.

[14] Poramate Horkaew et al. "Recent trends in surgical navigation systems using augmented and mixed reality: A review". In: *Computer Methods and Programs in Biomedicine* 232 (2023), p. 107350. DOI: 10.1016/j.cmpb.2023.107350.

[15] Interaction Design Foundation - IxDF. *What are VR Headsets?* Accessed: 2025-05-01. 2024. URL: https://www.interaction-design.org/literature/topics/vr-headsets.

[16] Interaction Design Foundation - IxDF. *What is Mixed Reality (MR)?* Accessed: 2025-05-01. 2022. URL: https://www.interaction-design.org/literature/topics/mixed-reality-mr.

[17] Interaction Design Foundation - IxDF. *What is Virtuality Continuum?* Accessed: 2025-05-01. 2022. URL: https://www.interaction-design.org/literature/topics/virtuality-continuum.

[18] Tim Jerman et al. "Enhancement of vascular structures in 3D and 2D angiographic images". In: *IEEE Transactions on Medical Imaging* 35.9 (2016), pp. 2107–2118. DOI: 10.1109/TMI.2016.2548501. URL: https://pubmed.ncbi.nlm.nih.gov/27076353/.

[19] Y. Jin et al. "Dual-Stage Transformer Framework for Liver Vessel Segmentation in CT Volumes". In: *Neurocomputing* 512 (Mar. 2025). Accessed: Jun. 18, 2025, pp. 105–117. DOI: 10.1016/j.neucom.2025.01.045.

[20] "JOURNAL OF SCIENTIFIC RESEARCH AND TECHNOLOGY (JSRT) VOLUME-1 ISSUE-5 AUGUST". In: *JSRT* (2023). Accessed: 2025-05-22. URL: https://jsrtjournal.com/index.php/JSRT/article/download/30/35/37.

[21] Ryan M. Juza and Eric M. Pauli. "Clinical and Surgical Anatomy of the Liver: A Review for Clinicians". In: *Clinical Anatomy* (2014). Accessed: 2025-05-03.

[22] C. Kirbas and F. Quek. "A review of vessel extraction techniques and algorithms". In: *ACM Computing Surveys* 36.2 (June 2004), pp. 81–121. DOI: 10.1145/1031120.1031121.

[23] Hermann Lang et al. "Impact of virtual tumor resection and computer-assisted risk analysis on operation planning and intraoperative strategy in major hepatic resection". In: *Archives of Surgery* 140.7 (2005), pp. 629–638. DOI: 10.1001/archsurg.140.7.629.

[24] William E. Lorensen and Harvey E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM SIGGRAPH Computer Graphics* 21.4 (1987). Accessed: 2025-06-18, pp. 163–169. URL: https://dl.acm.org/doi/10.1145/37402.37422.

[25] *Medical Image Segmentation Challenges*. Accessed: 2025-05-22. 2023. URL: https://d-nb.info/1342154339/34.

[26] S. Moccia et al. "Blood vessel segmentation algorithms — Review of methods, datasets and evaluation metrics". In: *Computer Methods and Programs in Biomedicine* 158 (May 2018), pp. 71–91. DOI: 10.1016/j.cmpb.2018.02.001.

[27] Frank H. Netter. *Atlas of Human Anatomy*. 6th ed. Elsevier Health Sciences, 2014.

[28] Ozan Oktay et al. "Attention U-Net: Learning Where to Look for the Pancreas". In: *International Conference on Medical Imaging with Deep Learning (MIDL)*. Accessed: 2025-06-08. arXiv preprint arXiv:1804.03999. 2018. URL: https://arxiv.org/pdf/1606.06650.

[29] Alexander Radtke and Hermann Lang. "Impact of 3D reconstruction on operation planning and intraoperative strategy in major hepatic resection". In: *Surgical Endoscopy* 32.5 (2018), pp. 2092–2101. DOI: 10.1007/s00464-017-5894-5.

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv preprint arXiv:1505.04597* (2015). Accessed: 2025-06-08. URL: https://arxiv.org/pdf/1505.04597.

[31] Yoshinobu Sato et al. "Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images". In: *Medical Image Analysis* 2.2 (June 1998), pp. 143–168. DOI: 10.1016/s1361-8415(98)80009-1.

[32] William Schroeder, Rob Maynard, and Berk Geveci. "Flying edges: A high-performance scalable isocontouring algorithm". In: *IEEE Symposium on Large Data Analysis and Visualization* (2015). Accessed: 2025-06-18, pp. 17–24. URL: https://ieeexplore.ieee.org/document/7347679.

[33] Alan Simpson et al. "The Medical Segmentation Decathlon". In: *arXiv preprint arXiv:1902.09063* (2019).

[34] Taylor & Francis. *Surgical Planning in Healthcare*. https://taylorandfrancis.com/knowledge/Medicine_and_healthcare/Surgery/Surgical_planning/. Accessed: 2025-05-03. 2025.

[35] Unity Technologies. *Unity Documentation*. Accessed: 2025-06-11. 2025. URL: https://docs.unity.com/en-us.

[36] *The accuracy verification of AR-based surgical navigation system*. ResearchGate. Accessed: 2025-06-08. URL: https://www.researchgate.net/figure/The-accuracy-verification-of-AR-based-surgical-navigation-system_fig8_275049311.

[37] *Uses of augmented reality in surgical consent and patient education – A systematic review*. PubMed Central. Accessed: 2025-06-08. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC12036893/.

[38] Caitlin T. Yeo et al. "Utility of 3D Reconstruction of 2D Liver Computed Tomography/Magnetic Resonance Images as a Surgical Planning Tool for Residents in Liver Resection Surgery". In: *Journal of Surgical Education* (2017). Accessed: 2025-05-03. DOI: https://sci-hub.se/10.1002/ca.22350.

[39] Wei Yu et al. “Liver Vessels Segmentation Based on 3D Residual U-Net”. In: *2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*. Accessed: 2025-06-14. IEEE. 2019. URL: https://sci-hub.se/https://ieeexplore.ieee.org/document/8802951.

[40] Rui Zhang et al. “An improved fuzzy connectedness method for automatic three-dimensional liver vessel segmentation in CT images”. In: *Computational and Mathematical Methods in Medicine* 2018 (2018), pp. 1–12. DOI: 10.1155/2018/4691797. URL: https://pubmed.ncbi.nlm.nih.gov/30510670/.