SurgeVision

CMPE 492 - Senior Design Project II

Low Level Design Report

TEAM MEMBERS:

ABDULLAH DOĞANAY

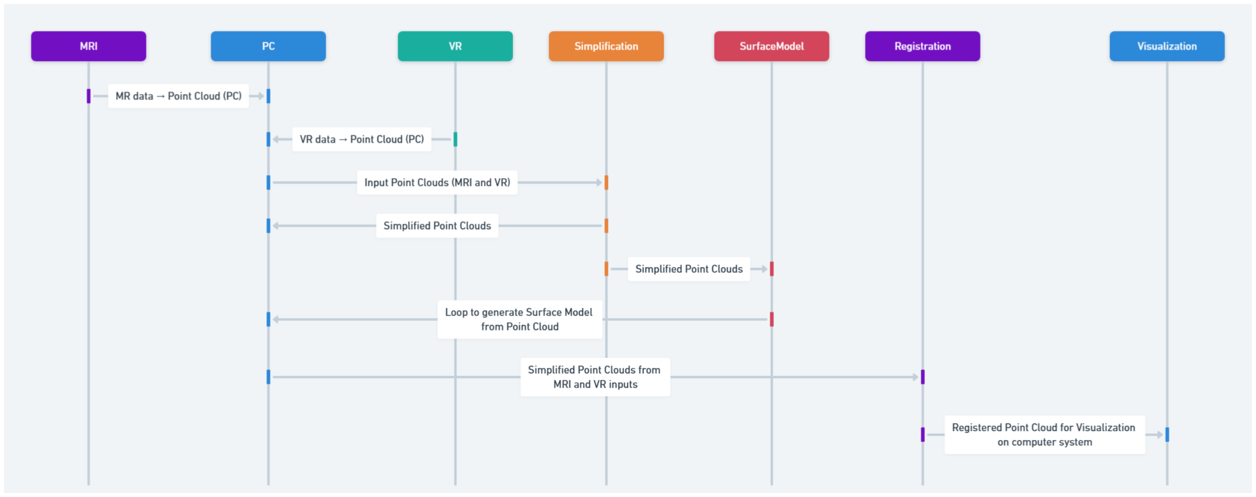
DENİZ POLAT

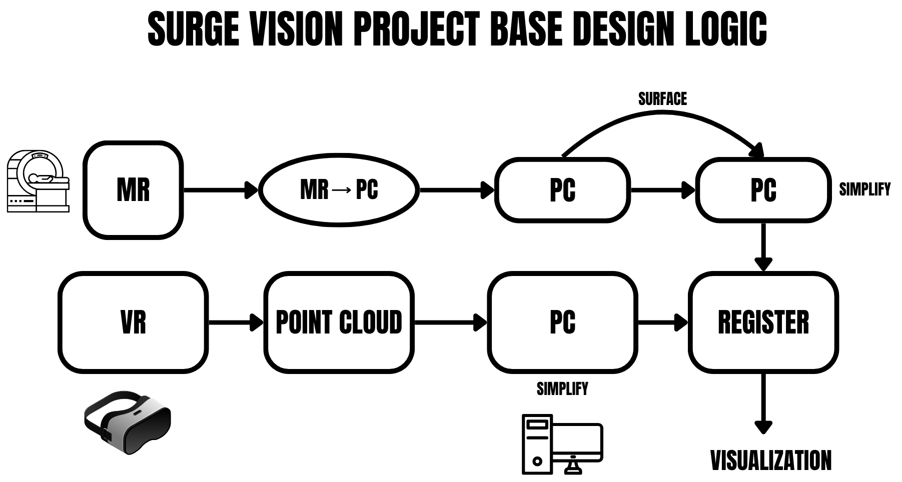
HAKAN UCA

ONUR USLU

1. Introduction

The purpose of this document is to provide detailed low-level design specifications for SurgeVision, a virtual reality (VR) and augmented reality (AR) system designed to facilitate the surgical planning and educational processes for hip replacement surgeries. The primary objectives include enhancing the precision and effectiveness of surgical planning through realistic 3D visualization, integrating real-time interactions with patient-specific anatomical models, and providing a safe, immersive educational platform for medical students and professionals. This document targets software engineers, system architects, medical professionals, stakeholders, and evaluators involved in the development, implementation, or assessment of the SurgeVision system. Clearly defined within this document's scope are architectural details, component interactions, interface specifications, and performance considerations. Identified limitations and constraints impacting the design include computational performance constraints, compliance with medical data privacy regulations, hardware capabilities of VR/AR equipment, and requirements for real-time responsiveness.


MRI
PC
VR
Simplification
SurfaceModel
Registration
Visualization
MR data → Point Cloud (PC)
VR data → Point Cloud (PC)
Input Point Clouds (MRI and VR)
Simplified Point Clouds
Simplified Point Clouds
Loop to generate Surface Model from Point Cloud
Simplified Point Clouds from MRI and VR inputs
Registered Point Cloud for Visualization on computer system



1.1. Object Design Trade Offs

Key design decisions prioritize modularity, scalability, ease of maintenance, and user-friendly interactions. Trade-offs were necessary between achieving modularity and managing system complexity.

The final design was selected due to its effective balance, facilitating future enhancements and easier integration with evolving technologies while maintaining manageable complexity.

The chosen design approach potentially impacts future development agility, system performance, and the overall user experience.

1.2. Interface Documentation Guidelines

Documentation standards adopted for SurgeVision emphasize clarity, consistency, and readability, ensuring all system interfaces are well-documented and accessible for developers and users.

The documentation approach provides structured, uniform interface specifications detailing method signatures, parameters, return values, and expected interaction sequences.

Recommended templates and formats are provided to standardize interface documentation across all system components.

Guidelines are clearly outlined for specifying preconditions, postconditions, and error-handling protocols, ensuring robust and predictable interface behavior.

1.3. Engineering Standards

SurgeVision adheres to engineering standards such as Unified Modeling Language (UML) for visual documentation and IEEE standards for technical documentation to ensure high-quality, standardized outputs.

These standards were chosen based on their broad acceptance and relevance to the project's requirements, facilitating clear communication and accurate implementation.

Compliance examples, including UML diagrams and standard documentation formats, are provided throughout the report to illustrate adherence.

Any deviations from these standards, along with justifications based on project-specific needs, are explicitly documented.

1.4. Definitions, Acronyms, and Abbreviations

- AR (Augmented Reality): Technology that integrates digital information with the user's environment in real-time.
- VR (Virtual Reality): Computer-generated simulation of a 3D environment, allowing user interaction in a seemingly real or physical way.

- ARCore: Google's platform for building AR applications, providing tools for interacting with point cloud data.
- MRI (Magnetic Resonance Imaging): Imaging technique used to generate detailed images of organs and tissues.
- Performance Tracking: Monitoring and recording of user actions and decisions to evaluate proficiency and improvement.
- Point Cloud: A dataset representing points in 3D space, typically used for representing external surfaces of objects.
- SurgeVision: The proprietary name of the VR/AR system being developed for surgical planning and training.
- UML (Unified Modeling Language): Standardized modeling language used in software engineering to visualize system designs.

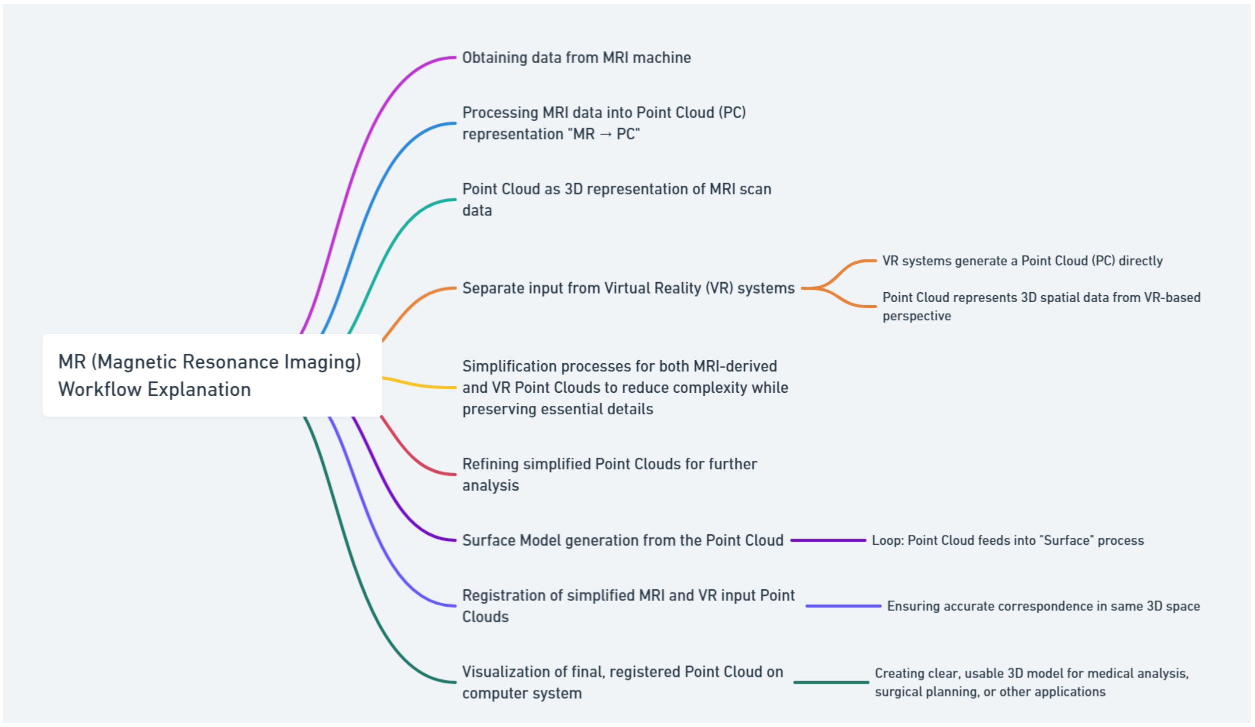
2. Packages

2.1. System Decomposition

The system is divided into logical modules to ensure clarity, maintainability, and efficiency. Each module has a specific responsibility within the project.

Module Name	Description
Data Acquisition Module	Manages the creation of point clouds from input data from MRI/CT scans.
AR/VR Data Acquisition Module	Using data from the AR/VR device, it scans the object in real time and obtains the point cloud of the object.

Data Preprocessing Module	Cleans, filters, and simplifies point cloud data to remove noise and reduce computation.
Registration Module	Implements Iterative Closest Point (ICP) and Coherent Point Drift (CPD) algorithms to align point clouds.
Visualization Module	Renders the registered point clouds in AR/VR space.
User Interface Module	Provides interaction with AR/VR devices and displays alignment results.
Tracking Module	Tracks objects in AR/VR space and updates point clouds dynamically.


MR (Magnetic Resonance Imaging) Workflow Explanation
Obtaining data from MRI machine
Processing MRI data into Point Cloud (PC) representation "MR → PC"
Point Cloud as 3D representation of MRI scan data
Separate input from Virtual Reality (VR) systems
VR systems generate a Point Cloud (PC) directly
Point Cloud represents 3D spatial data from VR-based perspective
Simplification processes for both MRI-derived and VR Point Clouds to reduce complexity while preserving essential details
Refining simplified Point Clouds for further analysis
Surface Model generation from the Point Cloud
Loop: Point Cloud feeds into "Surface" process
Registration of simplified MRI and VR input Point Clouds
Ensuring accurate correspondence in same 3D space
Visualization of final, registered Point Cloud on computer system
Creating clear, usable 3D model for medical analysis, surgical planning, or other applications

The SurgeVision system is structured into distinct software packages to ensure modularity, maintainability, and scalability. Each package corresponds to a specific functional module, working together to process and visualize MRI/CT scan data and AR/VR point clouds. The following section details each package and its responsibilities.

2.1.1. Data Acquisition Package

Purpose: Handles the creation of point clouds from MRI/CT scans.

Responsibilities:

- Reads MRI/CT scan data.
- Converts raw medical images into point cloud format.
- Ensures the accuracy and integrity of scan-derived point clouds.

Key Components:

- MedicalDataLoader: Loads MRI/CT scan files in standard formats.
- PointCloudGenerator: Converts medical images into 3D point cloud representation.
- DataValidator: Ensures completeness and correctness of point cloud data.

2.1.2. AR/VR Data Acquisition Package

Purpose: Captures real-time 3D scans from AR/VR devices.

Responsibilities:

- Uses AR/VR device sensors to scan physical objects.
- Generates real-time point cloud data from the environment.
- Synchronizes captured data with MRI/CT-derived point clouds.

Key Components:

- ARScanner: Uses ARCore-based technology for point cloud scanning.
- DepthSensorHandler: Extracts depth information for precise 3D modeling.
- ScanSynchronizer: Aligns real-time scans with pre-existing MRI/CT data.

2.1.3. Data Preprocessing Package

Purpose: Filters, cleans, and simplifies point cloud data.

Responsibilities:

- Removes noise and outliers from raw point cloud datasets.
- Simplifies complex point clouds to enhance performance.
- Prepares data for registration and visualization.

Key Components:

- NoiseFilter: Applies denoising algorithms to improve data quality.
- PointCloudSimplifier: Uses down sampling techniques to reduce point count.
- SurfaceExtractor: Converts raw point clouds into structured surface models.

2.1.4. Registration Package

Purpose: Aligns multiple point clouds using advanced registration algorithms.

Responsibilities:

- Implements Iterative Closest Point (ICP) for fine-tuned alignment.

- Uses Coherent Point Drift (CPD) for robust point cloud matching.
- Ensures accurate overlay of medical scan data and AR/VR scans.

Key Components:

- ICPAligner: Performs rigid transformation for point cloud alignment.
- CPDAligner: Uses probabilistic methods to adjust point cloud positions.
- RegistrationValidator: Evaluates alignment accuracy using error metrics.

2.1.5. Visualization Package

Purpose: Renders registered point clouds in AR/VR environments.

Responsibilities:

- Displays 3D point cloud models for medical analysis.
- Provides real-time updates and animations of the registered data.
- Supports multi-view rendering for better surgical planning.

Key Components:

- PointCloudRenderer: Displays point clouds in a 3D space.
- SceneManager: Manages virtual environment layouts.
- LightingManager: Enhances visual clarity using shading and depth cues.

2.1.6. User Interface Package

Purpose: Provides interactive tools for AR/VR device users.

Responsibilities:

- Allows surgeons and medical professionals to interact with 3D models.
- Displays alignment results and registration accuracy feedback.
- Supports gesture-based and controller-based interactions.

Key Components:

- UIManager: Handles menus, controls, and overlays.
- InteractionHandler: Captures user inputs from AR/VR devices.
- FeedbackDisplay: Shows real-time registration metrics.

2.1.7. Tracking Package

Purpose: Monitors object positions and dynamically updates point clouds.

Responsibilities:

- Tracks head, hand, or tool movements in AR/VR space.
- Updates point cloud data dynamically as objects move.
- Maintains real-time synchronization between virtual and real-world objects.

Key Components:

- MotionTracker: Detects and logs movement in AR/VR environments.

- PointCloudUpdater: Continuously adjusts point cloud positions.
- LatencyHandler: Minimizes delay in real-time tracking.

2.1.8. Summary of Package Interactions

These packages interact in a modular pipeline, ensuring seamless processing from data acquisition to visualization. The Registration Package serves as the core, aligning MRI, CT, and AR/VR scans into a unified point cloud representation for surgical planning and training.

2.2. Subsystems and Components

This section describes the hardware and software components that interact within the system.

2.2.1. Hardware Components

AR/VR Device (Meta Quest 3) – Captures real-world point cloud and displays MRI-derived point clouds.

Camera – AR/VR device scans patient data in real-time using depth sensing.

2.2.2. Software Components

- Point Cloud Library (PCL) – Handles point cloud processing and registration.
- FAISS (Facebook AI Similarity Search) – Efficient nearest neighbor search.

- FLANN (Fast Library for Approximate Nearest Neighbors) – Accelerates point cloud nearest neighbor computations.
- Eigen – Provides linear algebra operations.
- OpenMP (omp.h) – Enables parallel processing for performance improvements.
- TBB (Threading Building Blocks) – Supports multi-threaded computation.
- Unity – Displays results in an interactive AR/VR environment.
- ARCore – Handles real-time 3D scanning and tracking for AR applications.
- Zlib / znzlib – Compression libraries used for efficient data handling.
- Platform Details (Hardware/Software Environment)

2.2.3. Operating System

Android (for Meta Quest)

2.2.4. Programming Languages

Kotlin/Java – Android development.

C# – Unity development.

C++ – High-performance processing with PCL and other libraries.

Unity – AR/VR development.

2.2.5. Libraries/Frameworks

2.2.5.1.C++ Standard Libraries:

- <iostream>
- <vector>
- <cmath>
- <limits>
- <chrono>
- <memory>
- <cassert>
- <cctype>
- <cfloat>
- <cinttypes>
- <climits>
- <clocale>
- <cstring>
- <ctime>
- <cstdio>
- <cstdlib>
- <cstddef>

- <cstdint>
- <unistd.h> (For UNIX-based compatibility)

2.2.5.2.Point Cloud Processing:

- <pcl/io/ply_io.h> (PCL - Point Cloud Library)
- <pcl/point_types.h>
- <pcl/registration/icp.h> (Iterative Closest Point - ICP)
- <pcl/visualization/pcl_visualizer.h>
- <pcl/point_cloud.h>

2.2.5.3.Nearest Neighbor Search:

- <faiss/IndexFlat.h> (FAISS - Facebook AI Similarity Search)
- <flann/flann.hpp> (FLANN - Fast Library for Approximate Nearest Neighbors)

2.2.5.4.Parallel Processing:

- <omp.h> (OpenMP)
- <tbb/parallel_for.h> (Intel Threading Building Blocks)
- <tbb/blocked_range.h>
- <tbb/parallel_reduce.h>

2.2.5.5.Mathematics & Compression:

- <Eigen/Dense> (Eigen library for Linear Algebra)
- <zlib.h> (Compression)
- <znzlib.h> (Compression)

2.2.5.6.Unity & AR/VR Development:

- UnityEngine
- System.Collections.Generic
- System.IO
- System.Text
- System
- UnityEngine.InputSystem
- ARCore (Google's AR library)

2.2.5.7.Android Development:

- androidx.compose.ui:ui
- androidx.compose.material3:material3
- androidx.compose.foundation:foundation

- androidx.compose.runtime:runtime
- androidx.lifecycle:lifecycle-runtime-compose
- androidx.compose.runtime:runtime-livedata
- androidx.navigation:navigation-compose
- androidx.lifecycle:lifecycle-viewmodel-compose

2.3. Software Architecture

The system follows the MVVM (Model-View-ViewModel) architecture to ensure modularity, maintainability, and separation of concerns.

Model (M): Represents the core data structures, including point cloud data, medical scan information, and real-time AR/VR tracking data.

View (V): Consists of UI components, including the AR/VR visualization layer, user interactions, and feedback mechanisms.

ViewModel (VM): Acts as a bridge between the View and Model, handling business logic, data transformations, and state management for real-time processing.

This architecture is particularly beneficial for managing complex data flows between real-time scanning, point cloud processing, and user interactions while keeping the UI layer responsive and efficient.

2.4. Development Environment

Visual Studio + CMake – For C++ development.

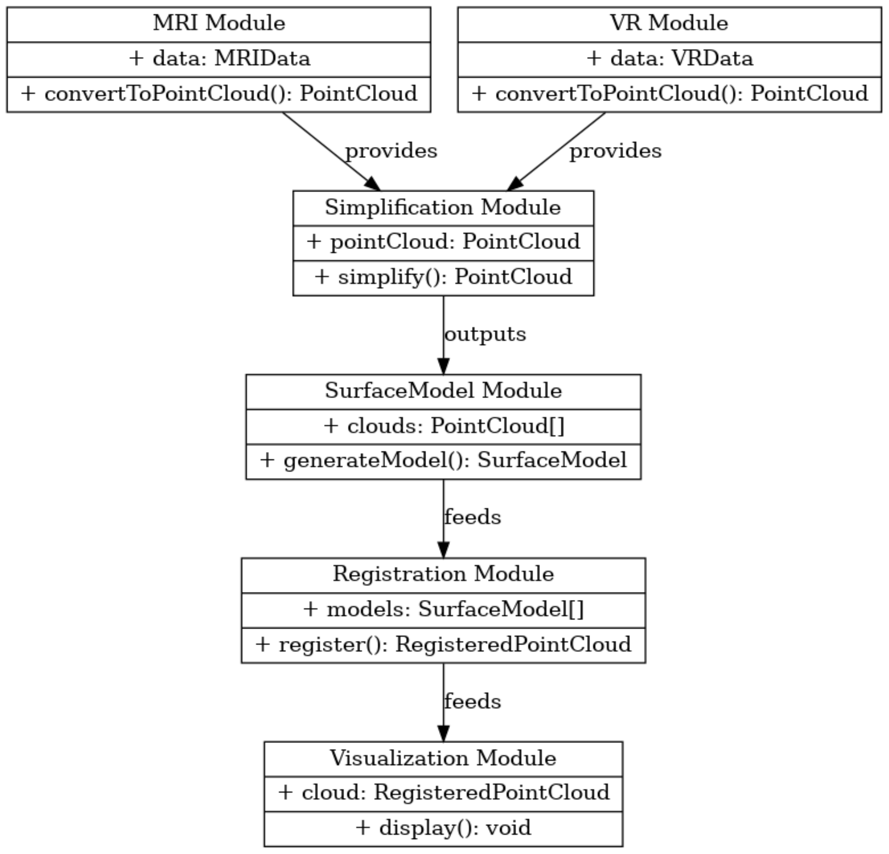
Unity Editor – For VR/AR interactions.

Android Studio – For Kotlin-based development.

VS Code / PyCharm – For scripting and additional tools.

3. Class Interfaces

This section details the class interfaces for key components of the SurgeVision system, defining their attributes, methods, and interactions.



3.1. Data Acquisition Package

- **Class: MedicalDataLoader**
    - **Attributes:**
        - filePath: String
    - **Methods:**
        - loadData(): PointCloud - Loads MRI/CT scan files and converts them into point clouds.
- **Class: PointCloudGenerator**
    - **Attributes:**
        - rawData: MRI/CT Scan
    - **Methods:**
        - generatePointCloud(): PointCloud - Converts medical images into a 3D point cloud representation.

3.2. AR/VR Data Acquisition Package

- **Class: ARScanner**
    - **Attributes:**
        - deviceID: String
    - **Methods:**

        - scanObject(): PointCloud - Captures 3D scans from AR/VR devices.

- **Class: DepthSensorHandler**
    - **Attributes:**
        - sensorRange: float
    - **Methods:**
        - extractDepthData(): DepthMap - Retrieves depth information for accurate modeling.

3.3. Data Preprocessing Package

- **Class: NoiseFilter**
    - **Attributes:**
        - filterThreshold: float
    - **Methods:**
        - applyFilter(PointCloud): PointCloud - Removes noise and outliers from point cloud data.
- **Class: PointCloudSimplifier**
    - **Attributes:**
        - simplificationRatio: float
    - **Methods:**

        - simplifyCloud(PointCloud): PointCloud - Reduces the number of points for performance optimization.

3.4. Registration Package

- **Class: ICPAligner**
    - **Attributes:**
        - iterationLimit: int
    - **Methods:**
        - alignClouds(PointCloud, PointCloud): PointCloud - Aligns point clouds using Iterative Closest Point (ICP).
- **Class: CPDAligner**
    - **Attributes:**
        - regularization: float
    - **Methods:**
        - alignClouds(PointCloud, PointCloud): PointCloud - Uses Coherent Point Drift (CPD) to align point clouds.

3.5. Visualization Package

- **Class: PointCloudRenderer**
    - **Attributes:**

        - renderSettings: RenderOptions
    - **Methods:**
        - render(PointCloud): void - Displays 3D point cloud data in AR/VR.

3.6. User Interface Package

- **Class: UIManager**
    - **Attributes:**
        - uiElements: List<UIComponent>
    - **Methods:**
        - displayMenu(): void - Displays the main user interface.

3.7. Tracking Package

- **Class: MotionTracker**
    - **Attributes:**
        - trackingMode: String
    - **Methods:**
        - trackMovement(): MotionData - Monitors object movements in AR/VR.

4. Glossary

- ARCore: Google’s software development kit (SDK) for building AR applications, supporting real-time 3D scanning, tracking, and rendering.

- CPD (Coherent Point Drift): A probabilistic algorithm used to align point clouds by minimizing the distance between them while considering non-rigid transformations.

- Data Acquisition Module: A system component responsible for collecting medical imaging data (MRI/CT scans) and converting them into point clouds.

- Eigen: A C++ library for performing linear algebra operations, commonly used in 3D graphics and point cloud transformations.

- FAISS (Facebook AI Similarity Search): A high-performance library for fast nearest neighbor search in high-dimensional spaces, used in point cloud processing.

- FLANN (Fast Library for Approximate Nearest Neighbors): A C++ library used to efficiently search for nearest neighbors in large datasets, improving point cloud alignment performance.

  ICP (Iterative Closest Point): A widely used algorithm for aligning two-point clouds by iteratively minimizing the distance between corresponding points.

- OMP (OpenMP): A parallel programming library that enables efficient multi-threaded execution for high-performance computations, such as point cloud processing.

- TBB (Threading Building Blocks): A parallel computing library that optimizes multi-threaded execution in CPU-intensive operations such as point cloud alignment.
- Zlib: A compression library used for efficient data storage and transmission in AR/VR applications.
- Znzlib: An extended version of Zlib optimized for handling large medical imaging datasets.

5. References

https://ieeexplore.ieee.org/Xplore/home.jsp