1. Task Overview

1. **Bone Segmentation (Task 1.1)**
    - **Goal**: Extract femur & tibia from the CT scan.
    - **Method**: Use thresholding + morphology (no AI).
    - **Output**: `femur_mask.nii.gz`, `tibia_mask.nii.gz`.

2. **Expand Mask by 2mm (Task 1.2)**
    - **Goal**: Grow tibia mask outward by 2mm.
    - **Note**: 2mm must be adjustable.
    - **Output**: `tibia_2mm.nii.gz`.

3. **Randomized Mask (Task 1.3)**
    - **Goal**: Create masks between original and 2mm edges.
    - **Rules**:
        - Never go beyond 2mm.
        - Never smaller than original.
    - **Output**: `tibia_rand1.nii.gz`, `tibia_rand2.nii.gz`.

4. **Landmark Detection (Task 1.4)**
    - **Goal**: Find lowest medial/lateral points on tibia
    - **Output**:
        - `landmarks.txt` with coordinates.

2. Methodology

2.1 Task 1.1: Bone Segmentation

1. **Thresholding**:
    - Applied threshold (HU=120) to extract bone structures
    - Created initial bone mask: `img_bones = np.where(img >= threshold, img, 0)`
2. **Edge Enhancement**:
    - Used unsharp masking to highlight edges
    - Created edge-enhanced image: `img_edges = img_bones + (img_bones - blurred) * 2`
3. **Mask Creation**:
    - Thresholded edge-enhanced image (value > 250)
    - Removed small objects (<2000 voxels)
    - Filled holes in both axial and sagittal planes
4. **Femur/Tibia Separation**:
    - Used connected components labeling
    - Separated bones by vertical position (centroid y-coordinate)
    - Femur = upper components, Tibia = lower component
5. **Post-processing**:
    - Applied morphological closing to each bone
    - Filled remaining holes
    - Multiplied with original image to get final segments
6. **Output**:
    - Saved combined mask as `overall_segment.nii.gz`
    - Note: Includes coordinate transformation to match original orientation

2.2 Task 1.2: Contour Expansion

1. **Voxel-Aware Dilation**:
    - Calculated expansion in voxels using image spacing (mm/voxel)
    - Created ellipsoidal structuring element matching physical dimensions
    - Used binary dilation to expand mask outward
2. **Outputs**:
    - Generated 2mm and 4mm expanded masks
    - Saved as `overall_segment_expanded_2mm.nii.gz` and `overall_segment_expanded_4mm.nii.gz`

2.3 Task 1.3: Randomized Contour Adjustment

1. **Ring Selection**:
    - Identified voxels between original and 2mm expanded boundaries
    - Randomly selected 50% and 80% of these boundary voxels
2. **Mask Creation**:
    - Combined original mask with randomly selected voxels
    - Ensured output stayed within expansion limits
3. **Outputs**:
    - Saved as `orginal_randomized_mask_1.nii.gz` (50%) and `orginal_randomized_mask_2.nii.gz` (80%)

2.4 Task 1.4: Landmark Detection

1. **Surface Extraction**:
    - Identified all surface voxels (bone voxels with at least one empty neighbor)
    - Split into medial/lateral regions using x-axis midpoint
2. **Lowest Point Detection**:
    - Found most inferior (minimum z-coordinate) points in each region
    - Returned as (z,y,x) coordinates in voxel space
3. **Implementation Notes**:
    - Handled edge cases where no surface points were found
    - Included coordinate transformation to match original DICOM orientation

3. Results

Landmark Detection Results (Tibia)

Coordinates in Voxel Space (Z,Y,X):

1. **Medial Lowest Point**: (72, 215, 244)
2. **Lateral Lowest Point**: (17, 0, 349)

4. Code Structure

```
NAMII/
├── data/
│   └── 3702_left_knee.nii         # Original CT scan input
├── results/
│   ├── coordinates.txt            # Landmark coordinates output
│   └── (masks/)                   # segmentation outputs
├── segmentation_env/               # Virtual environment (optional)
├── src/
│   ├── segmentation.ipynb         # Main segmentation code
├── .gitignore                      # Specifies untracked files
├── README.md                       # Project documentation
└── requirements.txt                # Python dependencies
```