Report on Bone Segmentation and Landmark Detection Tasks

Overview:

This project involved segmenting femur and tibia bones from 3D CT knee images, expanding the segmented contours, applying randomized contour adjustments, and detecting anatomical landmarks on the tibia. The tasks were implemented using image processing techniques without machine learning, and results were saved in .nii.gz format.

Task 1.1 – Bone Segmentation

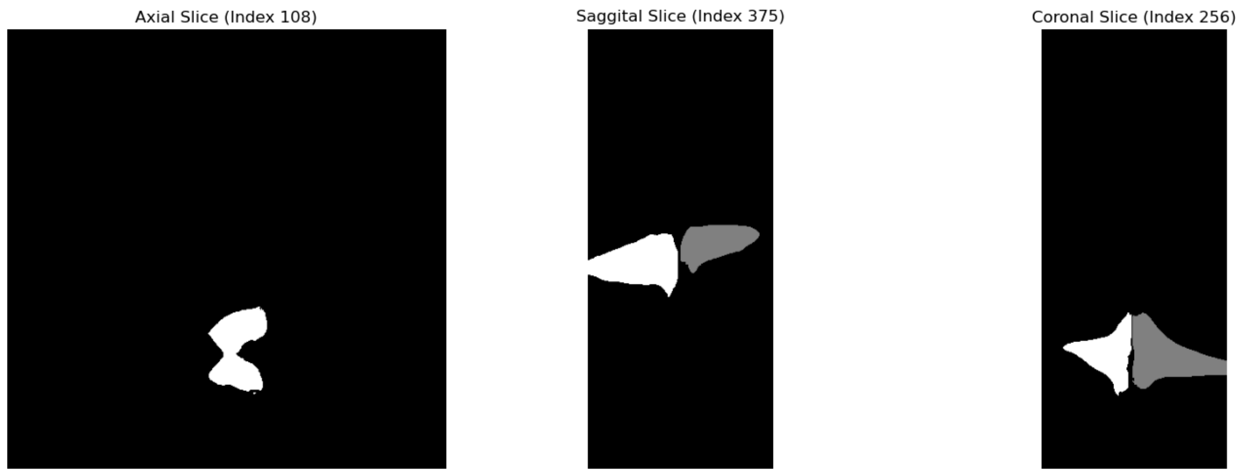
Goal: Segment femur and tibia bones from 3D CT images using image processing techniques.

Approach:

- Applied intensity thresholding on CT volume (HU > 320) to isolate bone regions.
- Preprocessed the binary mask by removing small noise components (remove_small_objects from scikit-image), filling internal holes slice-by-slice (binary_fill_holes from SciPy), and disconnecting femur and tibia by zeroing the connecting axial slice.
- Identified the two largest connected components using connected component labeling (ndimage.label from SciPy) and size filtering.
- Labeled components as femur or tibia based on their spatial position (center of mass along axial axis).
- Key packages used: NumPy, SciPy (ndimage), scikit-image (morphological operations), nibabel (for handling NIfTI files).

Outcome: Produced clean, labeled femur and tibia masks ready for subsequent processing.

Output Segmentation:



Task 1.2 – Contour Expansion

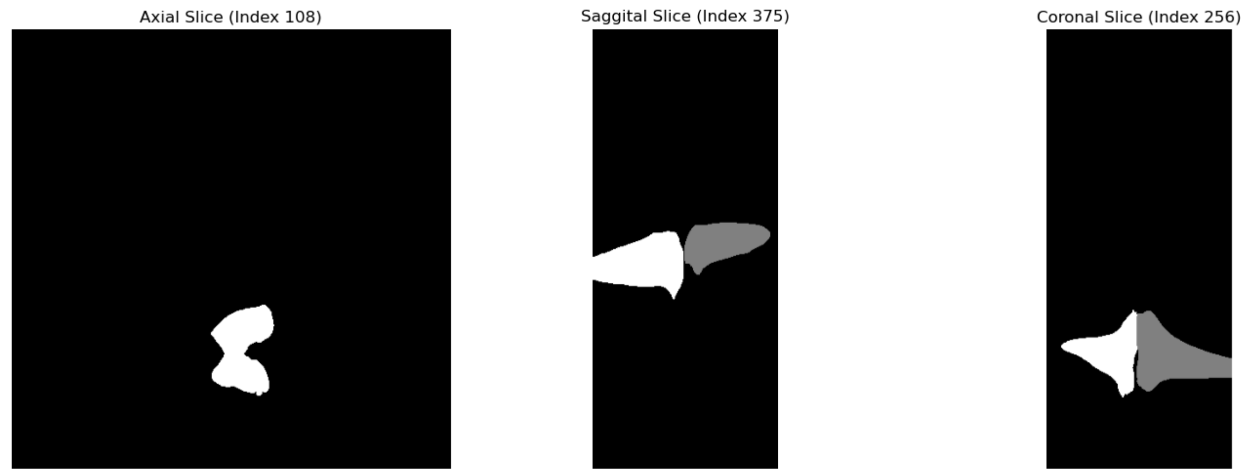
Goal: Expand the segmented femur and tibia masks outward uniformly by a specified distance (default 2 mm).

Approach:

- Calculated physical voxel spacing from the CT volume affine matrix to map voxel units to millimeters.
- Applied a Euclidean distance transform (distance_transform_edt from SciPy) to the inverted mask to measure the distance of each background voxel from the bone surface.
- Created an expanded mask by including voxels within the expansion radius.
- Processed femur and tibia masks separately to avoid overlap; resolved overlaps by assigning priority to the femur mask.
- Key packages used: NumPy, SciPy (distance_transform_edt), nibabel (for affine matrix handling).

Outcome: Generated expanded femur and tibia masks saved in .nii.gz format for further use.

Output Segmentation:



Task 1.3 – Randomized Contour Adjustment

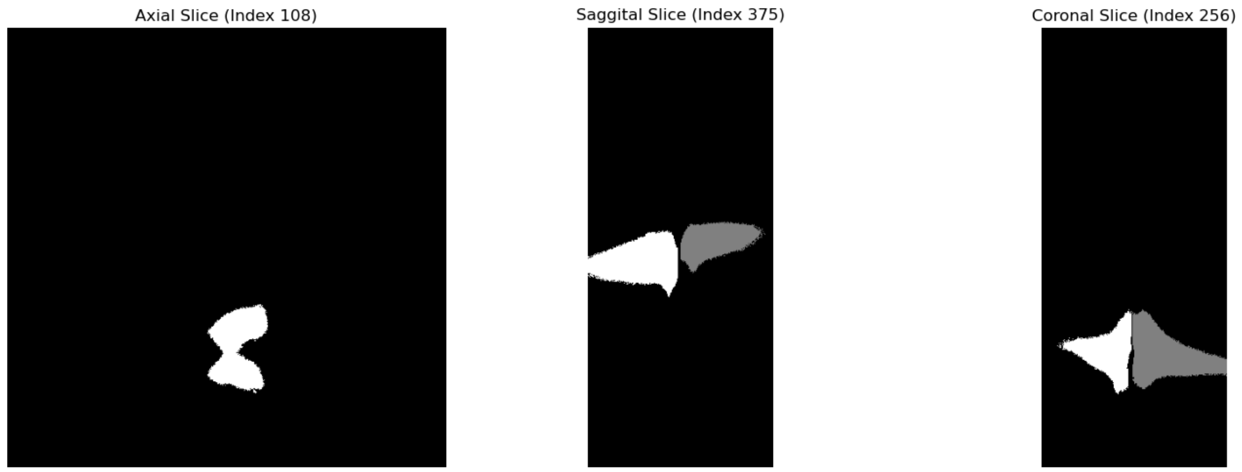
Goal: Generate randomized expanded masks that lie between the original segmentation and the 2 mm expanded contour, simulating natural variability.

Approach:

- Computed the Euclidean distance transform (distance_transform_edt from SciPy) on the inverted original mask to measure voxel-wise distances from the bone surface.
- Created a voxel-wise random expansion radius within [0, 2 mm].
- Constructed the randomized mask by including all original voxels plus background voxels whose distance to the original mask is less than or equal to their respective random radius.
- Applied independently to femur and tibia masks with different random seeds to generate multiple randomized contours.
- Key packages used: NumPy, SciPy (distance_transform_edt).

Outcome: Produced distinct randomized masks that do not shrink below the original and within 2 mm expansion limit, providing controlled randomness in bone contour expansion with a small margin for error in the results.

Output Segmentation Randomized



Task 1.4 – Landmark Detection on Tibia

Goal: Identify the medial and lateral lowest points on the tibial surface.

Approach:

- Processed and saved all five required masks: original, 2 mm expanded, 4 mm expanded, randomized mask 1, and randomized mask 2.
- Loaded tibia masks using nibabel.
- Extracted surface points by finding the highest Z index per (X, Y) coordinate.
- Split the surface into medial and lateral regions using the median X coordinate.
- Located the lowest points (minimum Z) in each region as anatomical landmarks.
- Converted voxel coordinates to real-world coordinates using affine matrices.
- Key packages used: nibabel, NumPy.

Outcome: Anatomical landmark coordinates were extracted for all processed masks, though maybe erroneous.

Output Coordinates:

Medial lowest (Tibia Mask): [-103.42786849,   15.64444613, -716.5]

Lateral lowest (Tibia Mask): [-117.33412421,   18.25186908, -742.5]

Medial lowest (Tibia Mask 2mm expanded): [-103.42786849,   18.25186908, -722.5]

Lateral lowest (Tibia Mask 2mm expanded): [-118.20326519,   19.99015105, -742.5]

Medial lowest (Tibia Mask 4mm expanded): [-103.42786849,   20.85929203, -728.5]

Lateral lowest (Tibia Mask 4mm expanded): [-119.07240617,   21.72843301, -742.5]

Medial lowest (Tibia Mask Randomized mask 1): [-99.95130455,   17.3827281, -720.5]

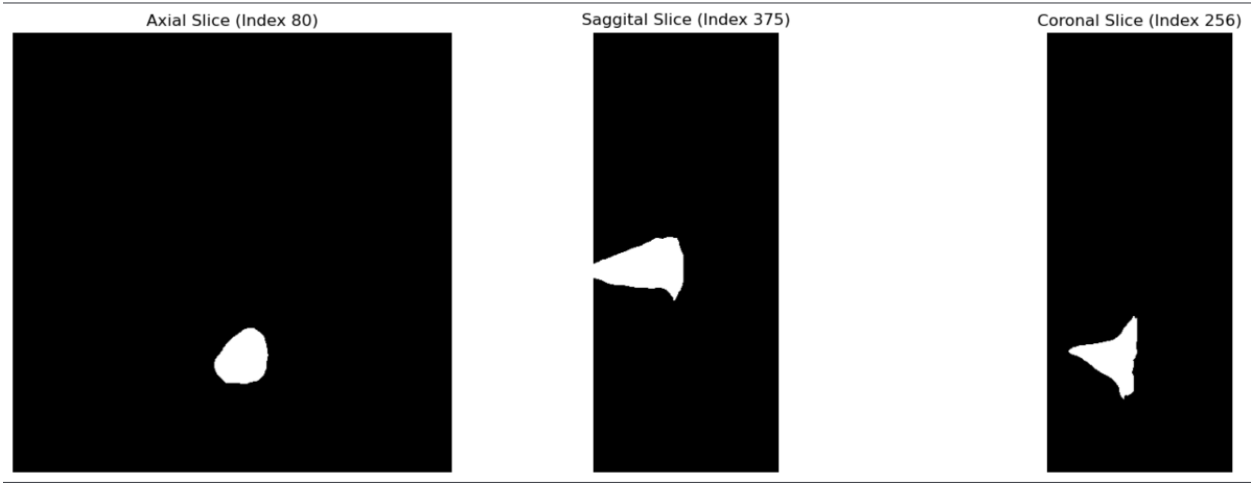
Lateral lowest (Tibia Mask Randomized mask 1): [-114.72670126,   19.99015105, -746.5]

Medial lowest (Tibia Mask Randomized mask 2): [-99.08216357,   17.3827281, -718.5   ]

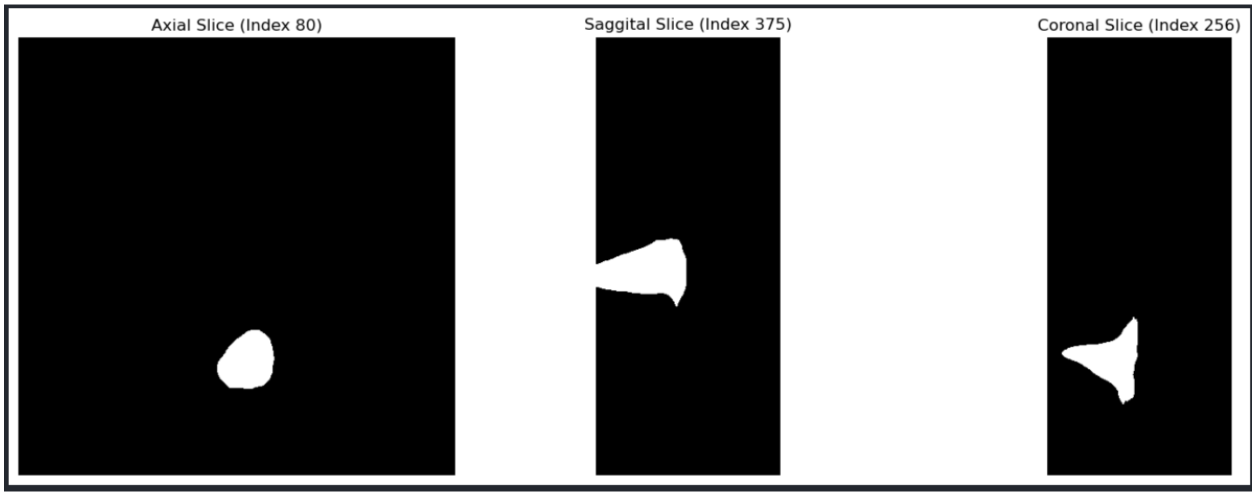
Lateral lowest (Tibia Mask Randomized mask 2): [-114.72670126, 19.99015105, -744.5]

Generated Masks

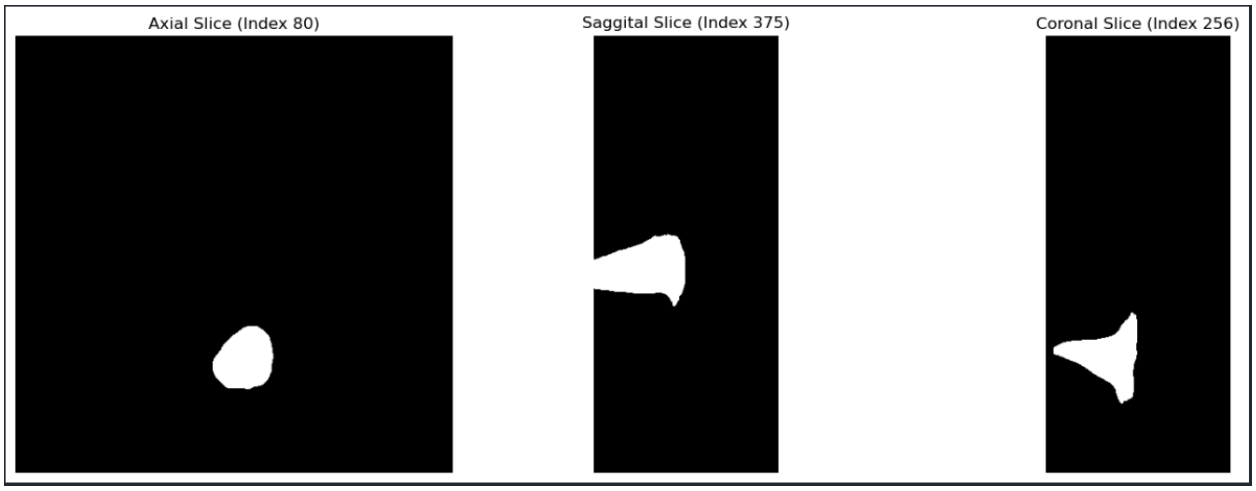
Tibia Mask



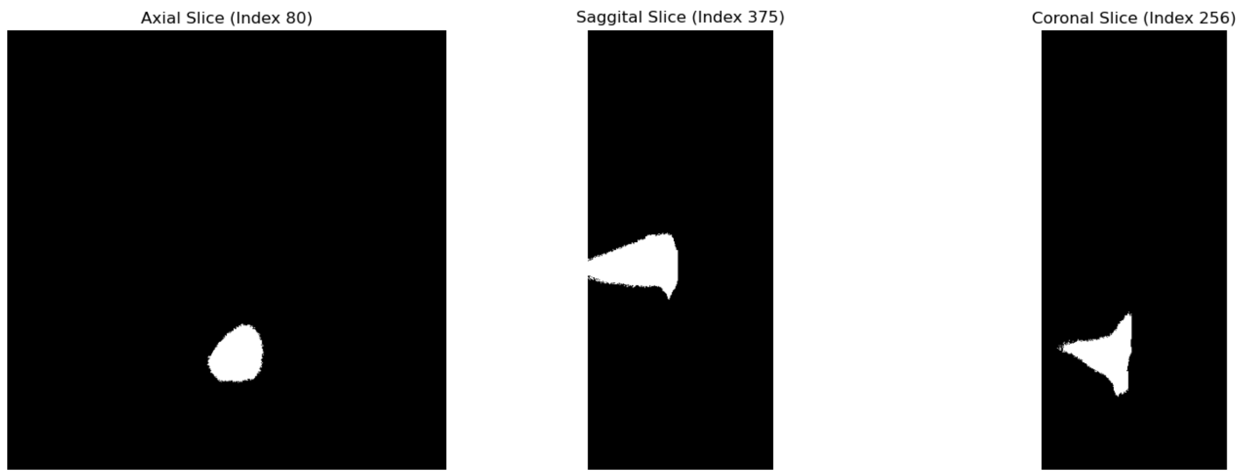
2mm Expanded



4mm Expanded

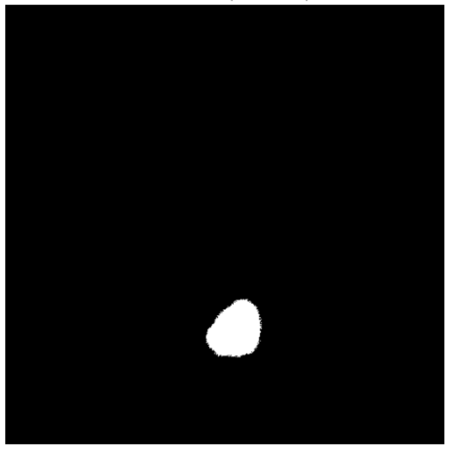


Randomized Mask 1 (Seed 42)

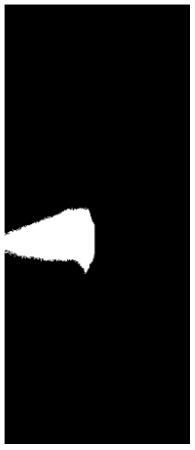


Randomized Mask 2 (Seed 99)

Axial Slice (Index 80)



Saggital Slice (Index 375)



Coronal Slice (Index 256)