NAAMI ASSIGNMENT - I

Author: Aakrit Dongol
Submission Date: 23 May 2025
Repository Link: *https://github.com/aakritd/naami-assignments*

1. Problem Understanding

This project centers on **bone segmentation in 3D CT volumes** of the knee region. The task is divided into four subtasks:

- **Task 1.1**: Segment the femur and tibia from the CT scan using only image processing techniques.
- **Task 1.2**: Expand the mask uniformly by 2 mm.
- **Task 1.3**: Randomize the segmentation contour between the original and expanded mask.
- **Task 1.4**: Detect the **medial** and **lateral lowest points** on the tibial surface for each mask.

2. Tools and Technologies Used

- **Language**: Python 3.10
- **Libraries**: `SimpleITK`, `NumPy`, `SciPy`, `Matplotlib`, `scikit-image`, `OpenCV`
- **Image Format**: `.nii.gz` (NIfTI)
- **Platform**: Jupyter Notebook
- **Version Control**: Git & GitHub

3. Approach and Methodology

Task 1.1 – Bone Segmentation

- First, the dataset was loaded using ‘nibabel’ library. It was found that there were 216 records of CT Scan, each with the dimension of (h=512, w=512).

  data = nib.load("3702_left_knee.nii").get_fdata()

- Then , the bones were segmented from all the CT Scan images. Each pixel value is represented by Hounsfield Unit. Bones have high density, so CT Scan results show higher HU values for bones compared to muscles and tissues. Using this logic, the bones were segmented with the help of ‘Threshold Segmentation’.

  segmented_images = data > segmentation_threshold

  The value of segmentation_threshold was chosen to be 500 as the Hounsfield Value for Bones are typically above 300.

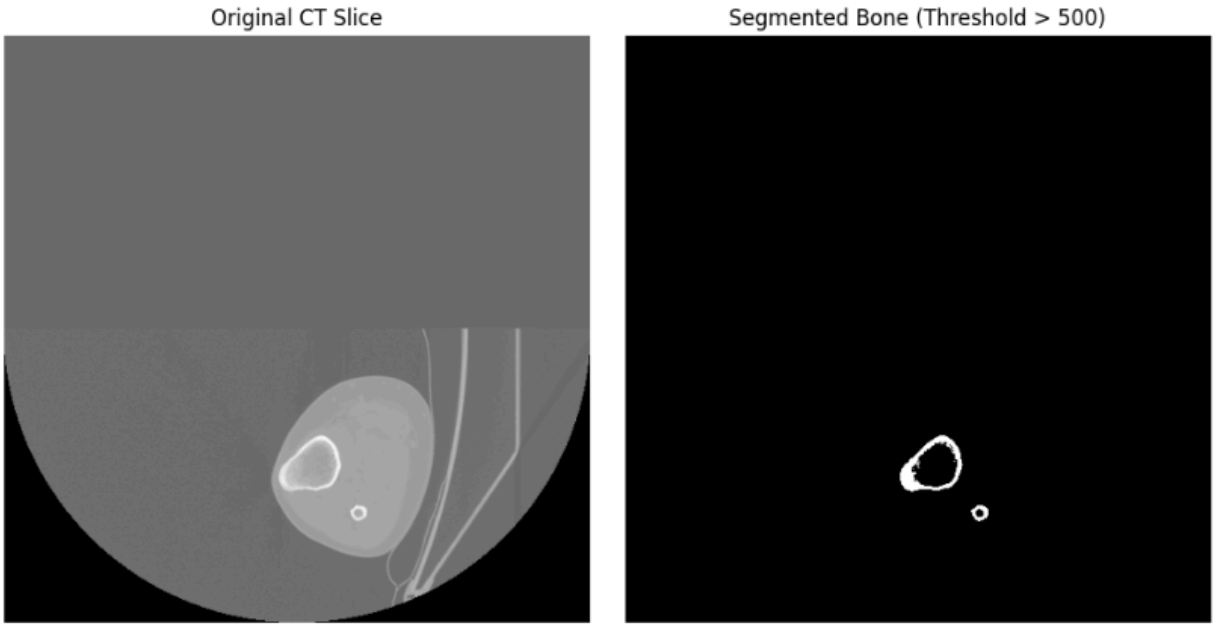


Fig. Original CT Scan and Segmented Bone

- Now, inorder to differentiate tibia and femur, we need to differentiate the lower leg and upper leg as lower leg contains tibia (along with fibia) and upper leg contains femur. In order to do this, we use the concept of connected components. Each connected

components can be seen as a single unit in the segmented image. If the segmented image has two connected components, it is treated as lower leg and if it ha one connected component, it is treated as upper leg. So, to find the number of connected components and it's corresponding image, we use 'label' function of 'scipy.ndimage' module.

labeled_arrays, num_features = label(segmented_images)

If the number of connected component is 1, then the white pixels representing femur is given red pixels by converting the 'gray-scale' image into 'rgb' format.

If the number of connected components is not 1, then the area of each connected component is computed. The component which has the highest area is given 'green' pixel values and all the other components are given white.

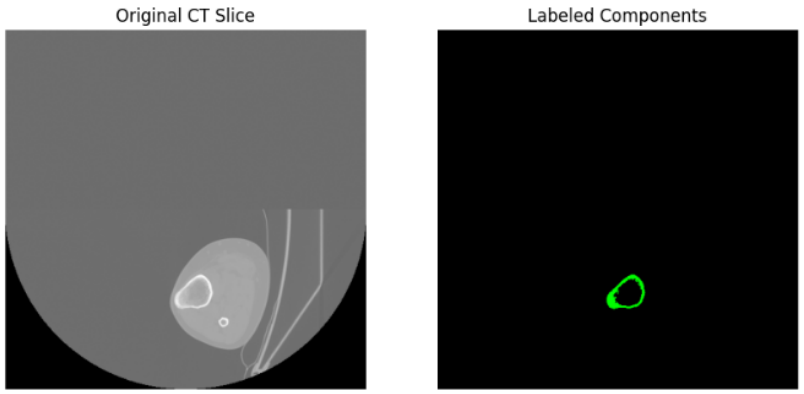


Fig. Segmentation of Tibia

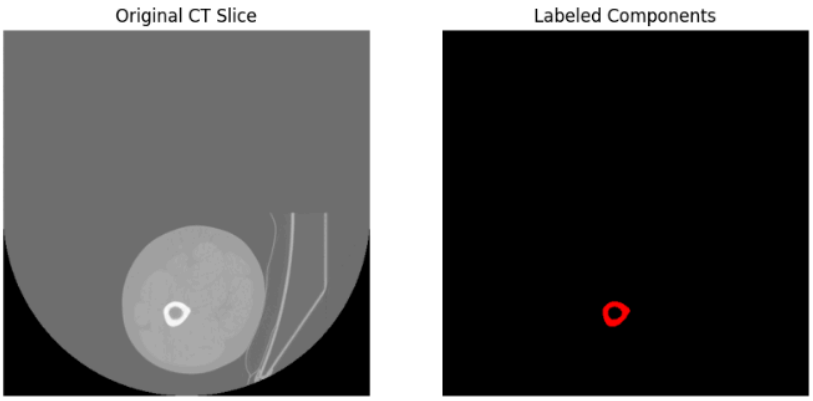


Fig. Segmentation of Femur

Task 1.2 – Contour Expansion

- The inputs for the contour expansion techniques are : Binary segmentation mask (`mask`), Image voxel spacing (`spacing`), and Desired expansion distance (`expansion_mm`, default = 2.0).

- Then, number of pixels corresponding to 2mm in both x and y directions were computed.

- To expand the segmentation accurately, we need a special shape called a **structuring element**. Since the CT images may have **different pixel sizes in the horizontal (x) and vertical (y) directions**, we can't just use a simple circle. Instead, we use an **ellipse** that matches the pixel size in each direction

    - For example, if 2 mm equals 5 pixels in the y-direction and 3 pixels in the x-direction, we create an ellipse of height 5 and width 3.

    - This elliptical shape ensures that the expansion happens evenly in real-world units (millimeters), not just in pixel units.

- Once we have the elliptical structuring element, we use it to **expand** the original segmented area. This is done using a technique called **binary dilation**, which grows the white (foreground) area in the mask outward. The dilation is applied to each 2D slice of the 3D CT scan (for example, each axial slice one by one). The elliptical shape determines how far the boundary expands in each direction.

Task 1.3 – Randomized Contour Adjustment

- Similarly, for random contour expansion between original segmented mask and 2mm, we multiple the contour expansion thickness with a ratio that is the value between 0 and 1, which results random contour adjustment between 0mm and 2mm.

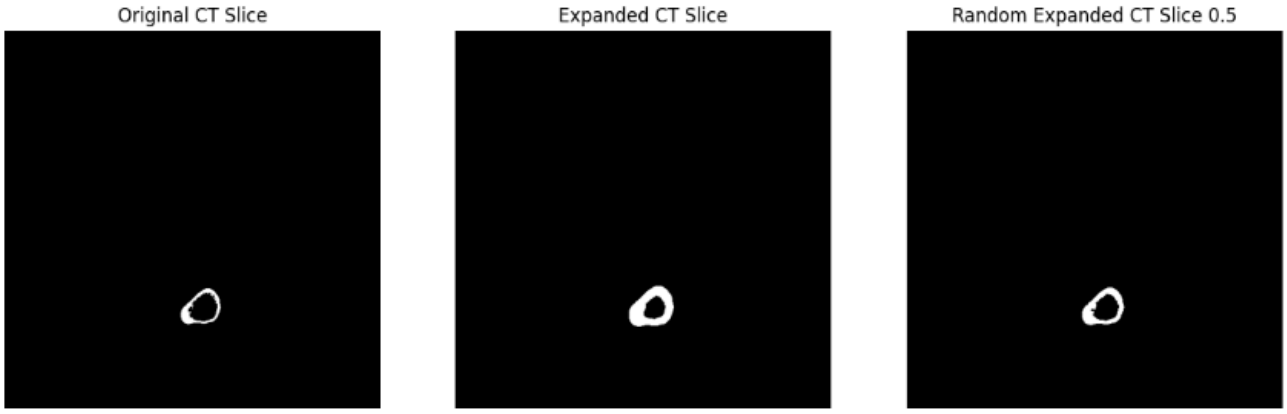
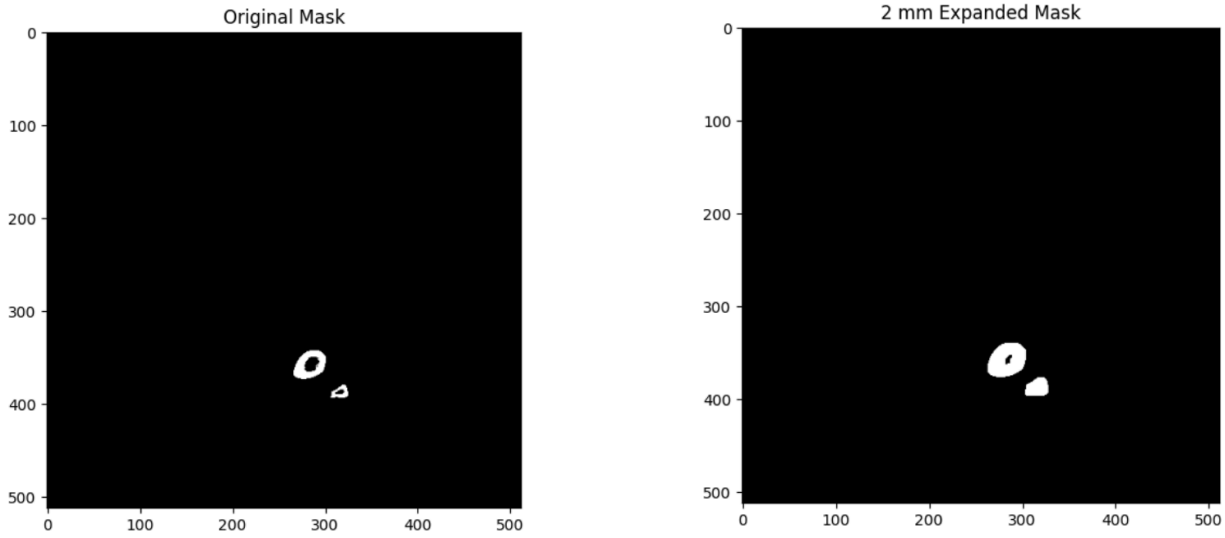
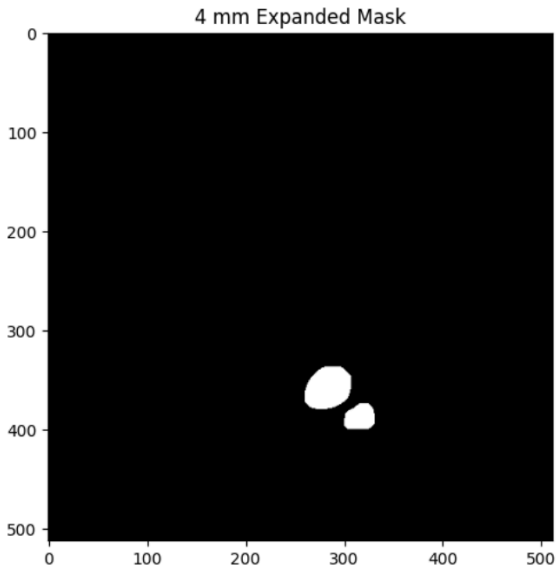


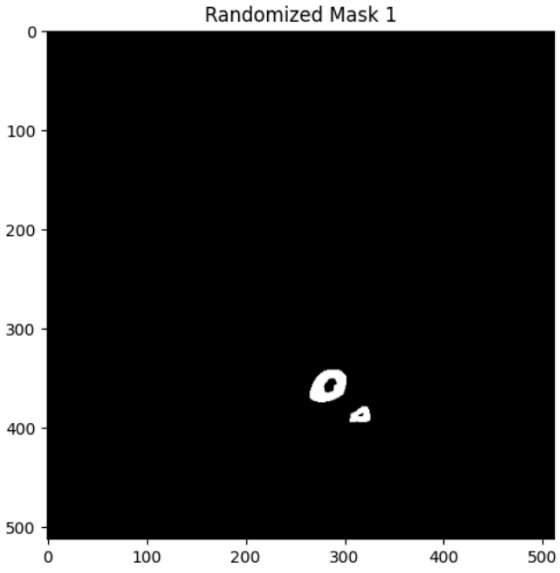
fig. Images showing original CT slice, expanded CT slice to 2mm, and Random Expanded CT slice

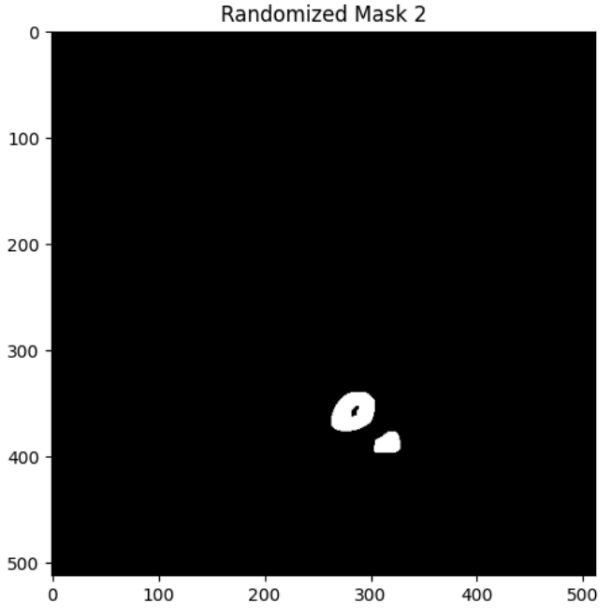
Task 1.4 – Landmark Detection on Tibia

- For this task, first we generated 2mm, 4mm and two kinds of random masks from the original mask using the same procedure as above. All these 5 kinds of images has the shape of (512, 512,216).









- In order to save these images, a dictionary was created as follows, where each key corresponds to the numpy array representeing the segmented masks :

masks = {

'original_mask.nii.gz': original_masks,

'expanded_2mm.nii.gz': mask_2,

'expanded_4mm.nii.gz': mask_4,

'randomized_1.nii.gz': random_1,

'randomized_2.nii.gz': random_2,

}

The generated masks were saved in the `.nii.gz` format, which is standard for storing medical imaging data. To ensure the saved masks align correctly with the original CT scan, the affine matrix from the original file (`3702_left_knee.nii`) was used. This matrix contains essential spatial information such as orientation and position. Each mask, including the original, 2mm and 4mm expanded versions, and the two randomized ones were converted into a NIfTI image using the `nibabel` library. The data was cast to `uint8` to optimize file size, and the original affine was applied to maintain spatial consistency. Each mask was then saved with a descriptive filename to make identification straightforward for later use.

- Now, we find the medial and lateral points of the lowest tibia. The steps are as follows :

1. **Label Connected Regions**:
   The algorithm starts by identifying all connected components (distinct regions) in the selected image slice (in this case, slice 60). This helps separate the femur and tibia, which are both visible in the image.

2. **Check and Select Tibia**:
   If only one region is found, it means the tibia is not present or not separated. If multiple regions are detected, it assumes the tibia is the largest one (based on area) and keeps only that region.

3. **Extract Tibia Contour**:
   It then finds the outer boundary (contour) of the tibia region, which forms a closed shape outlining the bone.

4. **Split into Medial and Lateral Parts**:
   The contour is split into two halves vertically:

   - The **medial part** is on the inner side (closer to the body's center).

- The **lateral part** is on the outer side.

5. **Find Lowest Points**:
   For each half, the point that is lowest (i.e. with the highest y-coordinate) is selected. These are the lowest medial and lateral points.

6. **Visualization**:
   Finally, the contour, midline, and lowest points are all drawn on the image using different colors to clearly visualize and verify the results.

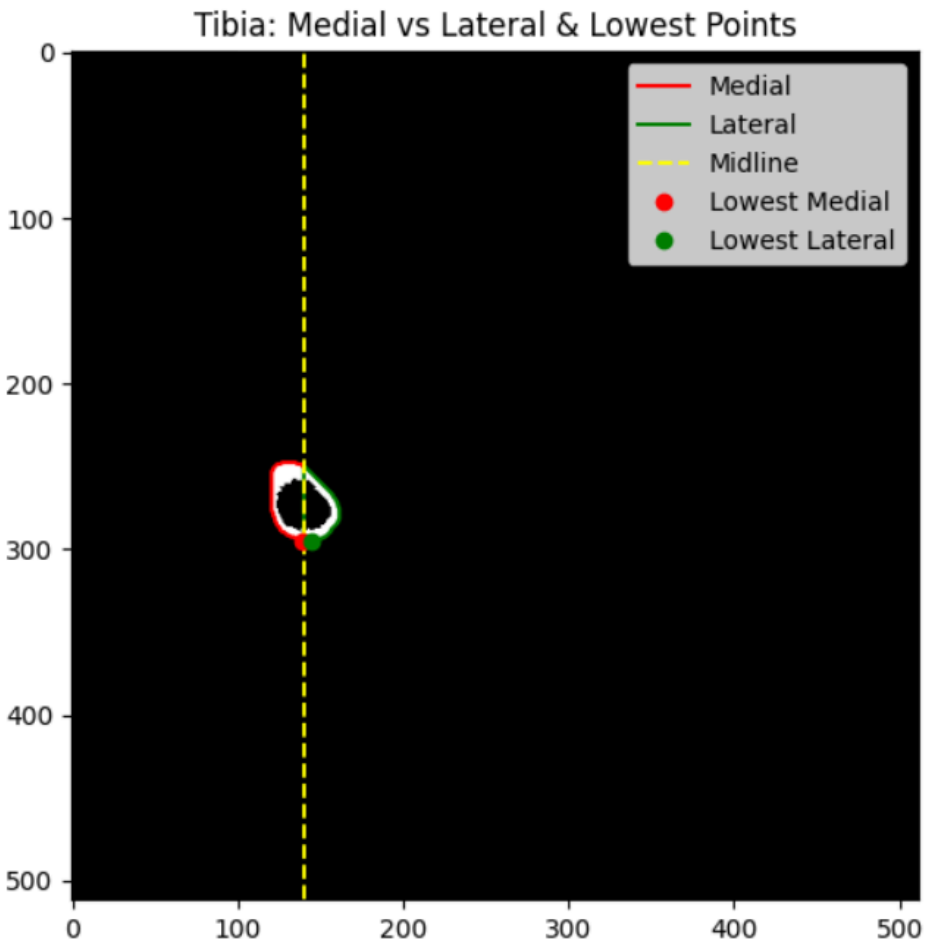


Fig. Image showing the lowest medial and lowest lateral of a tibia.

4. Use of AI Tools

AI Tools such as ChatGPT and Internet was used to:

1. Understand various terminologies such as tibia, fibula, femur, lateral and medial, Hounfield Unit and more.
2. To understand the the techniques of segmentation, connected components and contour expansion and it’s algorithm.