Bone Segmentation Project Report

Ishan Maharjan

May 23, 2025

Contents

1 Introduction

The Bone Segmentation project develops a Python package to process knee CT scans, segmenting femur and tibia regions, analyzing the lowest points on the tibia, and generating visualizations and summary reports. The project integrates custom exception handling and logging to enhance error tracking and debugging, ensuring robust execution. This report details the project's objectives, methodology, implementation, results, challenges (including resolution of a `ModuleNotFoundError`), and outcomes.

The primary goals were:

- Segment femur and tibia from knee CT scans using thresholding and 3D post-processing.
- Generate original, expanded, and random segmentation masks.
- Identify medial and lateral lowest points on the tibia in voxel and mm coordinates.
- Produce visualizations and a CSV summary of tibia points.
- Implement custom exception handling and logging across all modules.
- Resolve import errors to ensure seamless execution.

2 Methodology

2.1 Project Structure

The project adopts a `src/` layout for modularity, with the following structure:

- `src/`:
  - `main.py`: Orchestrates the pipeline and provides a console script.
  - `exception.py`: Defines `CustomException` for detailed error messages.
  - `logger.py`: Configures logging to timestamped files in `logs/`.
  - `components/`: Contains modules for I/O, segmentation, analysis, and visualization.
- `data/`: Stores input CT scan (`3702_left_knee.nii.gz`).
- `output/`: Stores masks, visualizations (`viz_img/`), and CSV summary.
- `logs/`: Stores log files.
- `setup.py`, `requirements.txt`: Define package setup and dependencies.

2.2 Functionality

- **Segmentation**: Uses Otsu thresholding, watershed, and 3D post-processing to segment femur (label 2) and tibia (label 1). Generates original, 2mm/4mm expanded, and random (65%/40%) masks.
- **Analysis**: Identifies tibia lowest points (medial and lateral) in voxel and mm coordinates for all masks.

- **Visualization**: Produces coronal slices, segmentation overlays, and tibia point plots, saved as PNGs.
- **Summary**: Exports tibia points to `tibia_points_summary.csv`.
- **Exception Handling**: `CustomException` provides file, line, and error details.
- `Logging`: Logs operations, warnings, and errors to `logs/<timestamp>.log`.

2.3 Tools and Libraries

- **Python Libraries**: `nibabel` (NIfTI I/O), `numpy` (array operations), `scipy` (image processing), `scikit-image` (segmentation), `matplotlib` (visualization), `pandas` (CSV output).
- **Package Management**: `setuptools` for package installation and console script.

3 Implementation

3.1 Core Pipeline

The pipeline, implemented in `main.py`, processes the input CT scan (`data/3702_left_knee.nii.gz`):

1. Loads the NIfTI file using `io_utils.load_nifti_image`.
2. Segments femur and tibia slices (`segmentation.segment_slice`), applies 3D post-processing, and generates masks.
3. Analyzes tibia lowest points (`analysis.find_tibia_lowest_points`).
4. Visualizes results (`visualization.*`) and saves to `output/viz_img/`.
5. Exports tibia points to `output/tibia_points_summary.csv`.

The console script (`bone_segmentation`) is defined in `setup.py` via `entry_points`.

3.2 Custom Exception Handling

`exception.py` defines `CustomException`, which captures:

- Script name, line number, and error message using `sys.exc_info()`.
- Example output:

```
Error occurred in python script name [main.py] line number [15]
   error message[Input file data/3702_left_knee.nii.gz does not
   exist]
```

All modules (`main.py`, `io_utils.py`, etc.) wrap operations in try-except blocks, raising `CustomException` for errors.

3.3 Logging

`logger.py` configures logging to write to `logs/<timestamp>.log` with:

- Format: `[ %Y-%m-%d %H:%M:%S ] line name - level - message`.
- Levels: `INFO` (operations), `WARNING` (e.g., insufficient bone volume), `ERROR` (failures).

Example log entry:

```
[ 2025-05-22 22:48:00 ] 10 root - INFO - Starting bone_segmentation
   console script
```

3.4 Error Resolution

A `ModuleNotFoundError: No module named 'logger'` was encountered when running `python main.py`. The issue stemmed from an incorrect import in `exception.py` (`from logger import logging`). The fix involved:

- Updating to `from .logger import logging` (relative import) in `exception.py`.
- Ensuring all modules use `from src.logger import logging` and `from src.exception import CustomException`.
- Installing the package (`pip install .`) or running as a module (`python -m src.main`) to resolve `src/` imports.

4 Results

4.1 Outputs

Running `bone_segmentation` produces:

- **Masks** (`output/`):
  - `original_mask.nii.gz`, `expanded_2mm_mask.nii.gz`, `expanded_4mm_mask.nii.gz`, `random_mask_1.nii.gz`, `random_mask_2.nii.gz`.
- **Visualizations** (`output/viz_img/`):
  - Tibia points: `tibia_lowest_points_original.png`, `tibia_points_all_masks.png`.
  - Segmentations: `segmentations_2mm_group.png`, `segmentations_4mm_group.png`, `knee_segmentation_*.png`.
- **CSV Summary** (`output/tibia_points_summary.csv`):
  - Columns: `Mask`, `Medial_X_mm`, `Medial_Y_mm`, `Medial_Z_mm`, `Lateral_X_mm`, `Lateral_Y_mm`, `Lateral_Z_mm`, `Medial_X_voxel`, `Medial_Y_voxel`, `Medial_Z_voxel`, `Lateral_X_voxel`, `Lateral_Y_voxel`, `Lateral_Z_voxel`.
  - Rows for each mask.
- **Logs** (`logs/<timestamp>.log`): Detailed execution trace.

4.2 Performance

The pipeline successfully:

- Segments femur and tibia with accurate labeling.
- Generates expanded and random masks, maintaining anatomical integrity.
- Identifies tibia lowest points, verified via visualizations.
- Logs all operations, aiding debugging.

The `ModuleNotFoundError` was resolved, enabling seamless execution.

5 Challenges

- **Import Error**: The `ModuleNotFoundError` required adjusting imports in `exception.py` and ensuring proper package installation.
- **Path Management**: The `src/` layout necessitated consistent imports (`src.logger`, `src.exception`) and running as a module or installed package.
- **Input Path Hardcoding**: `main.py` initially hardcoded the input path, fixed to use the provided parameter.
- **Logging Overhead**: Timestamped log files may accumulate; a single log file or rotation could be implemented.

6 Conclusion

The Bone Segmentation project delivers a robust Python package for knee CT analysis, achieving segmentation, tibia point analysis, visualization, and reporting. Custom exception handling and logging enhance reliability, while the resolution of the `ModuleNotFoundError` ensures seamless execution. Future improvements could include:

- Adding tests in `tests/` for validation.
- Implementing log file rotation.
- Supporting multiple input scans via command-line arguments.

The project is ready for use and extension, with comprehensive documentation in `README.md`.