**Spine Segmentation Project Document**

Table of Contents

[**Spine Segmentation Project Document** 2](#_Toc204943407)

[**Table of Contents** 2](#_Toc204943408)

[**Project Overview** 2](#_Toc204943409)

[**Purpose** 2](#_Toc204943410)

[**Intended Audience** 2](#_Toc204943411)

[**Summary** 2](#_Toc204943412)

[**Technologies Used** 3](#_Toc204943413)

[**System Architecture** 3](#_Toc204943414)

[**High-Level Workflow** 3](#_Toc204943415)

[**Directory Structure** 4](#_Toc204943416)

[**Pipeline Workflow** 4](#_Toc204943417)

[**1. DICOM Receiving** 4](#_Toc204943418)

[**2. Segmentation Trigger (API)** 4](#_Toc204943419)

[**3. Series Selection** 4](#_Toc204943420)

[**4. DICOM to NIfTI** 4](#_Toc204943421)

[**5. Segmentation Pipeline** 4](#_Toc204943422)

[**6. Postprocessing** 4](#_Toc204943423)

[**7. Convert to DICOM SEG** 5](#_Toc204943424)

[**8. Return to Client** 5](#_Toc204943425)

[**Module-wise Explanation** 5](#_Toc204943426)

[**1. receiver.py** 5](#_Toc204943427)

[**2. segmentation.py** 5](#_Toc204943428)

[**3. runner.py** 5](#_Toc204943429)

[**4. label\_burner.py** 5](#_Toc204943430)

[**5. series\_selector.py** 5](#_Toc204943431)

[**How to Deploy and Run** 6](#_Toc204943432)

[**🛠 Environment Setup** 6](#_Toc204943433)

[**🚀 Run Services** 6](#_Toc204943434)

[**API Reference** 6](#_Toc204943435)

[**POST /segment** 6](#_Toc204943436)

[**Troubleshooting & Logs** 7](#_Toc204943437)

[**Future Scope** 7](#_Toc204943438)

[**Conclusion** 7](#_Toc204943439)

**Project Overview**

**Purpose**

This project performs **automatic segmentation of vertebrae in spine MRI scans**. It takes in DICOM series of sagittal T1-weighted MR images and outputs a **DICOM SEG file containing labeled vertebrae**. The segmentation uses a two-stage deep learning pipeline built on **nnUNet** (referred to as Model 1 and Model 2).

**Intended Audience**

* Medical Imaging AI Developers
* PACS Integrators
* Radiology IT Engineers
* Anyone deploying or maintaining the segmentation pipeline

**Summary**

* **Input**: Sagittal T1 MRI DICOM series
* **Processing**:
  + Convert to NIfTI
  + Segment using TotalSpineSeg (Model 1 & 2)
  + Post-process and embed vertebra labels
  + Convert labeled NIfTI to DICOM SEG
* **Output**: DICOM SEG file with vertebral labels only (no masks)

**Technologies Used**

* **Language**: Python 3.11.9
* **Frameworks**:
  + FastAPI for the API layer
  + pydicom and pynetdicom for DICOM handling
  + [nnUNet](https://github.com/MIC-DKFZ/nnUNet) for deep learning segmentation
  + nibabel and OpenCV for NIfTI and image processing
  + totalspineseg for segmentation models

**System Architecture**

**High-Level Workflow**

+---------------------+

| DICOM Sender (e.g. PACS) |

+----------+----------+

|

Receives via port 5001

|

+--------v--------+

| receiver.py | <-- Listens for DICOM

+--------+--------+

|

v

+-------------------------+

| incoming\_temp / study\_uid |

+-----------+-------------+

|

Calls FastAPI endpoint with folder

|

+--------------+-------------+

AI inference -> segmented nifty output

+--------------+-------------+

|

+---------v----------+

| Label Burner |

+---------+----------+

|

+---------v----------+

| NIfTI to DICOM SEG |

+---------+----------+

|

+---------v----------+

| Return Final SEG |

+---------------------+

**Directory Structure**

totalspineseg\_inference/

├── app/

│ ├── main.py # FastAPI main entry

│ ├── api/segmentation.py # Handles POST /segment

│ └── service/

│ ├── runner.py # Core segmentation pipeline

│ ├── dic2nifti.py # DICOM to NIfTI conversion

│ ├── label\_burner.py # Adds vertebral labels

│ ├── resample\_nifti.py # Resampling logic

│ ├── nifti2dcmseg.py # Converts to DICOM SEG

│ └── series\_selector.py # Picks best sagittal T1 series

├── receiver.py # DICOM listener

├── totalspineseg/ # TotalSpineSeg models

│ ├── inference.py # Calls nnUNet APIs

│ ├── model1\_api/main.py # Stage 1 model

│ ├── model2\_api/main.py # Stage 2 model

**Pipeline Workflow**

**1. DICOM Receiving**

* receiver.py sets up a listener on port 5001.
* Incoming DICOMs are saved in: incoming\_temp/{StudyInstanceUID}/{SeriesInstanceUID}/
* If stable for 5s, the folder is sent to the API endpoint (http://localhost:8080/segment).

**2. Segmentation Trigger (API)**

* app/main.py starts FastAPI and mounts segmentation.py router at /segment.
* The endpoint receives a folder\_path and initiates processing.

**3. Series Selection**

* series\_selector.py finds the best **sagittal T1 series** using DICOM metadata.

**4. DICOM to NIfTI**

* The selected DICOMs are converted to a single .nii.gz file using dic2nifti.py.

**5. Segmentation Pipeline**

* runner.py calls run\_totalspineseg\_pipeline() from inference.py.
* Stage 1 and Stage 2 models are applied sequentially.
* Output: Segmented NIfTI mask saved in temp\_output/...

**6. Postprocessing**

* Resample mask to match original image size.
* Burn labels (e.g., "L1", "T10") using label\_burner.py.

**7. Convert to DICOM SEG**

* Final label-only NIfTI is converted to a DICOM SEG file using nifti2dcmseg.py.

**8. Return to Client**

* Final \*.dcm SEG is returned to the user from the FastAPI route.

**Module-wise Explanation**

**1. receiver.py**

* Listens for incoming DICOMs.
* Detects completed studies using file count stability.
* Calls API when stable.

**2. segmentation.py**

* Orchestrates full pipeline:
  + Series selection
  + NIfTI conversion
  + Inference
  + Resampling, labeling, SEG generation

**3. runner.py**

* Core pipeline engine.
* Loads config from .env
* Calls run\_totalspineseg\_pipeline
* Moves output NIfTI to final destination.

**4. label\_burner.py**

* Assigns human-readable vertebral labels into each slice.
* Uses OpenCV to burn labels.
* Only labels are kept (not full masks).

**5. series\_selector.py**

* Identifies T1 sagittal series using:
  + Series Description ("Sag T1")
  + TR/TE ranges (e.g., TR: 300-800ms, TE: 10-30ms)

**How to Deploy and Run**

**🛠 Environment Setup**

1. Install dependencies:

pip install -r requirements.txt

1. Create a .env file in root:

MODEL1\_API=http://localhost:8001/infer

MODEL2\_API=http://localhost:8002/infer

DEVICE=cpu

PREVIEW=False

QUIET=False

CHECKPOINT=checkpoint\_final.pth

**🚀 Run Services**

**Start FastAPI:**

uvicorn model1\_api.main:app --host 0.0.0.0 --port 8001

uvicorn model2\_api.main:app --host 0.0.0.0 --port 8002

uvicorn app.main:app --host 0.0.0.0 --port 8080

**Start DICOM Receiver:**

python receiver.py

**API Reference**

**POST /segment**

Triggers segmentation from a folder of DICOM series.

**Request JSON:**

{

"folder\_path": "absolute/path/to/dicom/study"

}

**Response:**

* 200 OK: Returns DICOM SEG file.
* 400: Invalid folder
* 500: Processing error

**Troubleshooting & Logs**

|  |  |
| --- | --- |
| **Problem** | **Solution** |
| No output SEG | Check model logs and ensure proper series selection |
| FastAPI not starting | Ensure correct Python version and uvicorn installed |
| DICOM not received | Verify port 5001 is open, no firewall blocks |

Logs are printed to the console by default. You can redirect them or add logging config for production.

**Future Scope**

* Add support for IVD (disc) segmentation.
* Support multiple input series or multi-modality.
* Integrate DICOMweb for PACS transfer.
* Add UI for manual series verification.

**Conclusion**

This document provides an in-depth walkthrough of the spine segmentation pipeline, from incoming DICOMs to final labeled DICOM SEG outputs. It is modular, reusable, and designed for clinical or research integration.

For any maintenance or upgrade, refer to:

* runner.py for core logic
* receiver.py for DICOM interface
* segmentation.py for workflow coordination