Requirements Document: Extracting Spinal Midline from Depth Camera Images

1. Project Overview

This document outlines the requirements for processing depth camera images of a human back to extract and visualize the spinal midline. The task involves converting 2D depth data into a 3D point cloud, identifying key anatomical landmarks (C7 vertebra, sacrum, and posterior superior iliac spines - PSIS), detecting the spinal "valley" (midline groove), and overlaying the computed midline on a 3D surface model. This is inspired by biomechanical analysis techniques, such as those using low-resolution depth sensors like Kinect or similar (e.g., Orbbec cameras supporting OB_FORMAT_Y16).

The system should handle input data from depth cameras, perform geometric computations, and output visualizations or data files (e.g., point clouds with annotated midlines). Accuracy targets: Landmark detection within ~5-10mm (based on research benchmarks); midline fitting to capture spinal curvature deviations.

2. Functional Requirements

2.1 Input Data Handling

- **Depth Images**: Support for 16-bit grayscale images (format: OB_FORMAT_Y16 or similar, resolution: 640x576 or variable).
- **Metadata**: Camera intrinsics (Fx, Fy, Cx, Cy) and extrinsics if available; timestamp and frame info for multi-frame processing.
- **Additional Files**: Optional color images for overlay; research documents (e.g., DOCX with methods) for reference.
- **Validation**: Filter invalid depths (e.g., 0 or saturated values); handle noise from sensor artifacts.

2.2 Processing Pipeline

- **Point Cloud Generation**: Convert depth map to 3D XYZ coordinates using intrinsics.
- **Landmark Detection**:
  - Manual: Interactive selection tools.
  - Automatic: Curvature-based detection (Gaussian/Mean curvature maps) for C7, sacrum, L/R PSIS.
- **Midline Detection**:
  - Identify spinal "valley" via minima detection, symmetry analysis, and polynomial fitting.
  - Compute 3D spinal midline considering vertebral rotation and depth estimates (e.g., $L(Ys) = 0.132T - 0.035 * Ys$ from research).
- **Visualization**: Overlay midline curve on 3D point cloud or mesh; support for rotation, zoom, and export (e.g., images or PLY files).
- **Output**: Annotated point cloud, midline coordinates (CSV/JSON), visualizations (PNG/ interactive viewers).

2.3 Non-Functional Requirements

- **Performance**: Process single frame in <5 seconds on standard hardware (CPU/GPU optional for ML enhancements).
- **Accuracy**: Align with research methods (e.g., curvature criteria: convex/concave/parabolic); validate against ground truth if available.
- **Usability**: Beginner-friendly interface (e.g., script with CLI options or simple GUI).
- **Scalability**: Handle batch processing for multiple subjects/frames.
- **Error Handling**: Graceful failures for invalid data; logging for debugging.

3. Technical Requirements

3.1 Hardware

- **Minimum**: Standard PC (e.g., Intel i5, 8GB RAM) for basic processing.
- **Recommended**: GPU (NVIDIA) for advanced features like ML-based detection.
- **Sensor Compatibility**: Depth cameras (e.g., Kinect 2, Orbbec) at ~1m distance, 1.3m height, as per research.

3.2 Software and Libraries

- **Programming Language**: Python 3.8+ (for accessibility).
- **Core Libraries**:
  - NumPy/SciPy: For array operations, filtering, and curve fitting.
  - OpenCV/ImageIO: For image loading (16-bit support).
  - Open3D: For point cloud handling, normals/curvature estimation, and visualization.
  - Matplotlib/Plotly: For 2D/3D plotting.
  - Scikit-Image/SciPy Signal: For edge/minima detection.
- **Optional for Advanced**:
  - PyTorch/TensorFlow: For ML-based landmark detection (e.g., pre-trained models on anatomical data).
  - MeshLab/CloudCompare: Standalone tools for manual validation.
- **Development Environment**: Anaconda/Jupyter for easy setup; no internet-required packages beyond defaults.

3.3 Data Requirements

- **Sample Data**: At least one depth image + metadata; ideally raw 16-bit files (not colorized visualizations).
- **Ground Truth**: Annotated landmarks for testing (if provided by client).
- **Privacy**: Ensure anonymized data; comply with health data regulations (e.g., HIPAA if applicable).

3.4 Skills and Resources

- **Developer Skills**: Basic Python programming; familiarity with computer vision/3D geometry. Advanced: Surface analysis, biomechanics knowledge.
- **Time Estimate**: 10-20 hours for prototype (beginner level); additional for automation/testing.
- **Dependencies**: Research papers (e.g., IEEE on Kinect-based back analysis) for method validation.
- **Testing**: Unit tests for each module (e.g., point cloud projection); integration tests on sample data.

4. Assumptions and Constraints

- Subject positioning: Standing straight, back facing camera, no clothing artifacts.
- No real-time requirements; offline processing.
- Budget: Open-source tools only; no paid software.
- Risks: Sensor noise affecting accuracy; manual intervention needed if auto-detection fails.

5. Deliverables

- Prototype script/codebase.
- Processed results for provided data (point cloud, landmarks, midline visualization).
- Documentation: User guide and this requirements doc.