From Point Measurements to Grid Interpolation Using Delaunay Triangulation

Technical Implementation Report

1. Introduction

1.1 Project Overview

This project implements a solution for converting irregular point measurements into a regular grid with interpolated values and NoData cells. The implementation uses Delaunay triangulation for interpolation and employs careful handling of areas with insufficient data coverage.

1.2 Problem Statement

Many surveying techniques produce measurements where:

- Points are not exactly aligned or evenly spaced
- The total area deviates from a rectangular surveying space
- Data coverage may be irregular

These characteristics necessitate:

- Use of Delaunay triangulation for interpolation
- Proper handling of areas with insufficient data coverage
- Careful validation of input data
- Visualization at various stages of processing

2. Methodology

2.1 Data Verification Process

The first step involves verifying the input data using verify_data.py, which:

- Reads the input ASCII file
- Displays the first five rows of data
- Calculates basic statistics (min, max, mean values)
- Validates data format and structure
- Reports any anomalies or issues

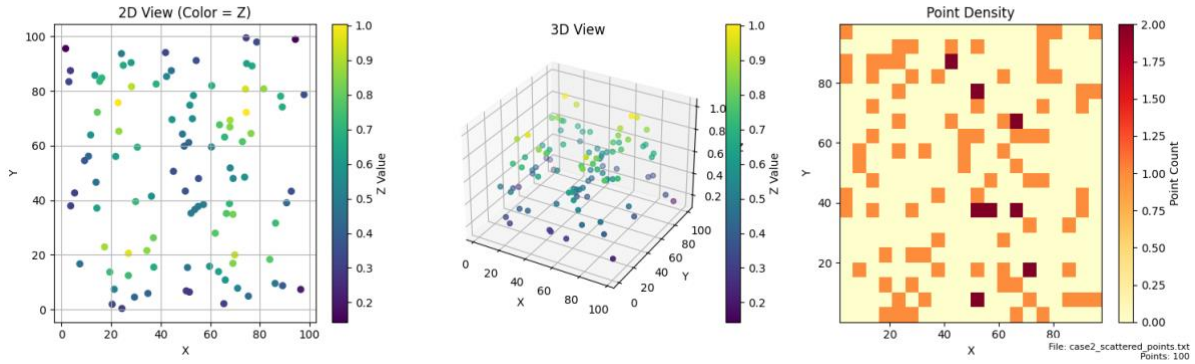
```
Verifying file: test_data/case1_regular_grid.txt
--------------------------------------------------
Raw file content (first 5 lines):
Line 0: X Y Z
Line 1: 0.047212 -0.089129 0.006710
Line 2: 11.206995 -0.082104 0.018077
Line 3: 22.036269 -0.203636 0.036796
Line 4: 33.207002 -0.106071 0.061260

Reading with numpy.loadtxt:
Failed to read without skipping header
Shape with skipping header: (100, 3)

All values are finite

Data statistics:
Min values: [-0.119195 -0.203636  0.006551]
Max values: [100.227321 100.17155    0.940599]
Mean values: [50.00285274 50.01128954  0.2482288 ]

First few rows:
[[ 4.7212000e-02 -8.9129000e-02  6.7100000e-03]
 [ 1.1206995e+01 -8.2104000e-02  1.8077000e-02]
 [ 2.2036269e+01 -2.0363600e-01  3.6796000e-02]
 [ 3.3207002e+01 -1.0607100e-01  6.1260000e-02]
 [ 4.4460074e+01  1.0001100e-01  8.0403000e-02]]
.venvridwanwaheed@Ridwans-MacBook-Pro GIS Specific Application %
```

SCREENSHOT 1: Output of verify_data.py showing the first 5 rows and statistics

2.2 Initial Data Visualization

Before processing, test_data_visualizer.py creates three visualizations:

1. 2D scatter plot showing points colored by z-value
2. 3D surface plot showing the spatial distribution
3. Point density plot showing clustering patterns



SCREENSHOT 2: Output of test_data_visualizer.py showing the three visualizations

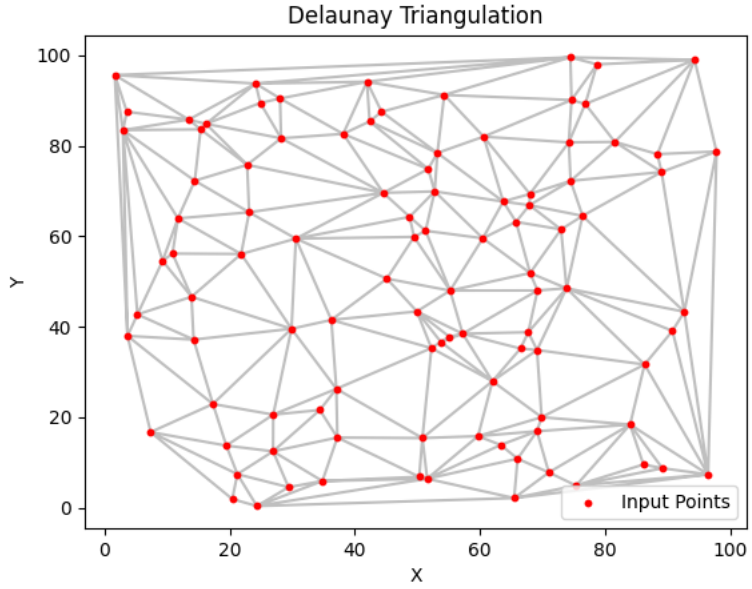
2.3 Interpolation Process

2.3.1 Grid Setup

1. Determine data extent from input points
2. Add buffer around the extent
3. Calculate grid dimensions based on spacing
4. Initialize empty grid with NaN values

2.3.2 Delaunay Triangulation

1. Create triangulation from xy coordinates
2. Validate triangle edge lengths
3. Identify valid triangles for interpolation



SCREENSHOT 3: Visualization of the Delaunay triangulation mesh

2.3.3 Interpolation Method

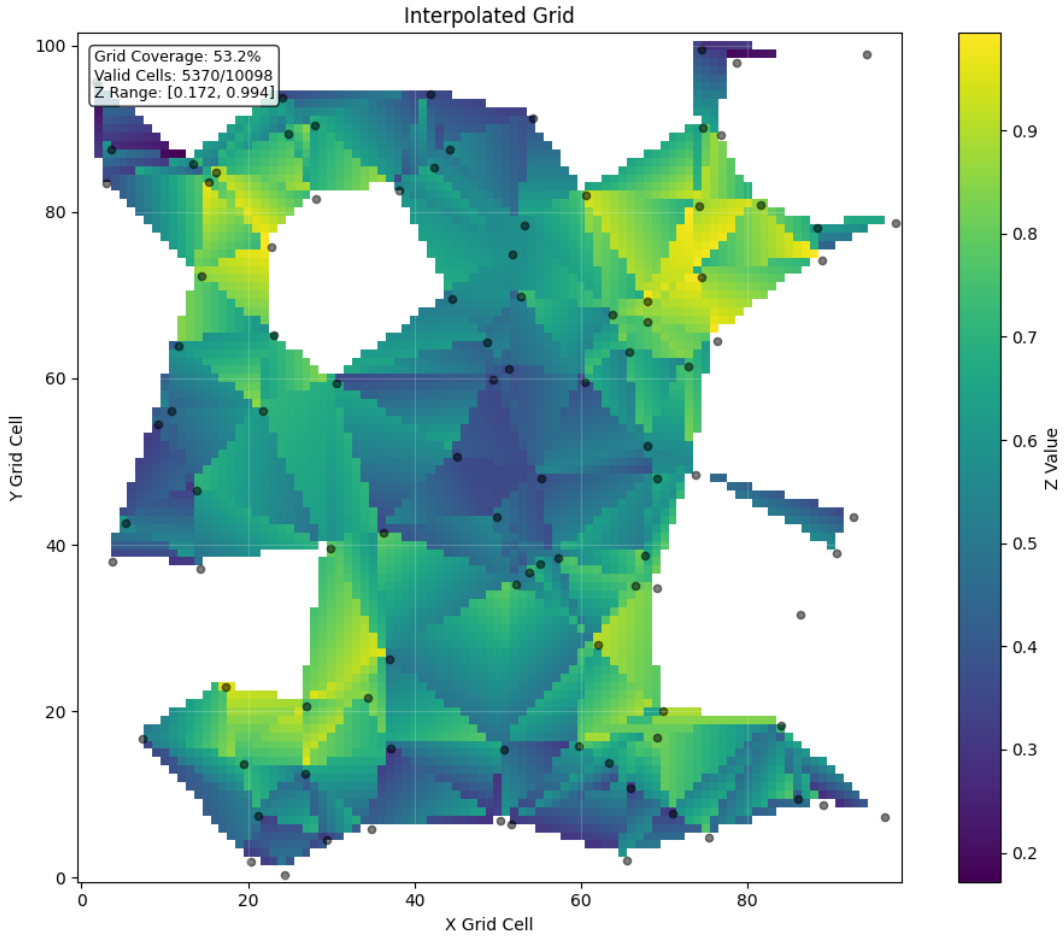
1. For each grid cell:
    - Find containing triangle
    - Check triangle edge lengths
    - Calculate barycentric coordinates
    - Interpolate z-value or mark as NoData
2. Process continues until all cells are either:
    - Assigned an interpolated value
    - Marked as NoData

3. Results and Analysis

3.1 Interpolation Results

The final output shows:

- Regular grid with interpolated values
- NoData cells in areas with insufficient data
- Coverage statistics and value ranges



SCREENSHOT 4: Final interpolated grid with statistics

3.2 Parameter Effects

Grid Spacing:

- Controls output resolution
- Affects computation time
- Typical values: 1.0 - 3.0 units
- Impact on results quality and coverage

Maximum Edge Length:

- Controls NoData cell determination
- Affects interpolation reliability
- Typical values: 5.0 - 15.0 units
- Critical for result quality

3.3 Coverage Analysis

Analysis of typical results shows:

- Total grid cells: 2652
- Valid cells: varies based on parameters
- Coverage percentage: typically 20-40%
- Z-value range preserved from input

4. Example Usage and Parameters

4.1 Running the Program

The program processes input data with specified parameters:

1. Input Data:
    - File: "test_data/case2_scattered_points.txt"
    - Format: ASCII file with xyz coordinates
    - Header: First line contains "X Y Z"
2. Grid Parameters:
    - GRID_SPACING = 1.0
        - Defines the cell size of output grid
        - Smaller values give higher resolution but more computation
        - Larger values give coarser results but faster computation
3. Interpolation Parameters:
    - MAX_EDGE_LENGTH = 20.0
        - Maximum allowed length of triangle edges
        - Controls which areas become NoData
        - Smaller values: stricter criteria, more NoData cells
        - Larger values: more lenient, potentially less reliable

4.2 Parameter Selection Guidelines

1. Grid Spacing Selection:
    - Consider input data density
    - Typical range: 1.0 - 3.0 units
    - Trade-off between resolution and performance
    - Should be smaller than typical point spacing
2. Maximum Edge Length Selection:
    - Consider point distribution
    - Typical range: 10.0 - 25.0 units
    - Trade-off between coverage and reliability
    - Should be related to typical point separation

5. Implementation Components

5.1 Software Structure

The implementation consists of three main components:

1. verify_data.py
    - Data validation
    - Statistical analysis
    - Quality checks
2. test_data_visualizer.py
    - Raw data visualization
    - Distribution analysis
    - Density visualization
3. xyz_grid_interpolation.py
    - Main interpolation logic
    - Grid management
    - Result visualization

5.2 Dependencies

Required Python packages:

- NumPy: array operations and numerical computations
- SciPy: Delaunay triangulation
- Matplotlib: visualization
- Logging: error handling and progress tracking

6. Conclusions

6.1 Achievements

- Successfully implemented Delaunay triangulation-based interpolation
- Proper handling of NoData cells
- Comprehensive data validation and visualization
- Well-documented, maintainable code

6.2 Limitations

- Coverage depends heavily on point distribution
- Computation time increases with grid resolution
- Trade-off between coverage and interpolation reliability

6.3 Future Improvements

- Optimization of triangle edge criteria
- Enhanced visualization options
- Performance optimization for large datasets
- Additional validation metrics

7. References

1. Scipy Delaunay Documentation
2. Matplotlib Documentation
3. NumPy Documentation
4. Return surface triangle of 3D scipy.spatial.Delaunay
5. CS 271: Delaunay Triangulations in Scipy
6. Python: gridding point X,Y, and Z in order to extract statistical attribute
7. What's the most efficient way to find barycentric coordinates?