Stippling of 2D Scalar Fields

VU Visualisierung 2 (186.833) - Research Paper Implementation

David Bauer (12120495) / Dominik Kanjuh (12433751)

Project Concept

The goal of our project is to visualize elevation data of Earth's surface through stippling, leveraging the Linde-Buzo-Gray algorithm to produce a point distribution based on the elevation levels. The application will offer interactivity (pan, zoom) and customization options (color and size of the stiples) which will allow users to adjust the level of detail and the transfer functions.

Implementation

Dataset

The project will use digital elevation data in .hgt files (NASA's Shuttle Radar Topography Mission (SRTM) file type) which contains a height map of the Earth, with latitude and longitude aligned z-values to represent elevation. The dataset, provided by J. de Ferranti and C. Hoffmann, which we will be using, was featured in the referenced paper. Other datasets will be visualisable, provided they have the following attributes: latitude, longitude, scalar value (e.g. height).

Database

We decided to use PostGIS (extension of PostgreSQL) to handle geospatial data efficiently. It will let us store, query, and manage location data directly in the database, making it faster and easier to work with large sets of elevation data.

Data Processing

We plan to store height values in a PostGIS database. This will make it easier for us to query the geospatial data instead of the raw binary SRTM files. The height values will be aggregated to the server based on the window settings of the client (visible area) and then sent over.

Backend

Our project’s backend will be built with Node.js and Typescript. Because of our experience with it, Node.js is a good choice for us. Since we don’t need high speed or multi-threading,

we aren’t missing out on valuable performance either. The API will expose a route used to get the precomputed stippling data, for example: *“/stipples?min_lat=45.8150&min_lng=15.9819&max_lat=53.94&max_lng=26.503”*.

The API will be called on every pan or zoom of the map.

Frontend

Our project’s frontend will be built using Typescript and WebGPU. The stipples will be requested from the PostGIS server. When the user pans the screen or zooms in and out, new data will be requested according to the new settings. The transfer-function will run client-side and will be able to be changed by the user during runtime.

Implementation Steps

1. Aggregate points on server

After the client requests data from the server for a specific region, the client will aggregate all height values within that region (for example via averaging nearby points) and return only a specified number of values. This will ensure that the Linde-Buzo-Gray algorithm doesn’t slow down with increasing area size.

2. Linde-Buzo-Gray algorithm

The extended Linde-Buzo-Gray algorithm as specified in the paper, will run on the points sent by the server and create stipples. Because the total number of points is restricted, a large slowdown here is not expected.

3. Display via WebGPU

The calculated stipples will be shown via WebGPU. If the window is interacted with the new context will be sent to the server and a new set of points will be queried.

Grading Criteria

Quality of algorithm as implemented in the reference paper	30% | 12/40
Visual result	20% | 8/40
Feature-richness / interactivity	20% | 8/40
Generalizability	20% | 8/40
Performance	10% | 4/40
Creativity concerning extensions of the paper	00% | 0/40