Homework 1
Scattered data interpolation

Advanced computer graphics 2020/21

1 Introduction

Many tasks start with gathering the data by scanning the real world. Often, this data comes in an unstructured (scattered) form, where each sample may fall anywhere in the domain. This so-called *scattered data* must not be confused with unstructured data, where the samples themselves have no specified structure. Here are a few examples of scattered data:

- Weather data (temperature, atmospheric pressure, humidity, wind speed and direction etc.) is usually measured at weather stations, which are scattered across the globe, either on land or sea.
- Hydrological data (salinity, groundwater level, evaporation rate etc.) is of critical importance in water supply networks, agriculture and environmental engineering.
- 3D scanned data with applications including 3D modeling, land surveying, archaeology, robotic mapping, industrial design and autonomous vehicles. A common example is lidar data, where each sample may hold information about color, intensity, scan angle, scan direction and classification.

The scattered data represents just a small set of values of a spatially varying quantity, which we often want to evaluate not only at the sample locations but also at arbitrary locations between the samples. We therefore need to use some kind of interpolation to be able to get a full view of the data.

1.1 Formal problem statement

Formally, the interpolation problem is rather simple to state. For this assignment, assume that we are working in Euclidean space. We are interested in a quantity $f : \mathcal{D} \to \mathbb{R}$ defined on a domain $\mathcal{D} \subseteq \mathbb{R}^n$. Given N distinct samples $x_1, ..., x_N \in \mathcal{D}$ and their values $y_1, ..., y_N \in \mathbb{R}$, construct the function $\hat{f}$ such that $\hat{f}(x_k) = y_k$ for $k = 1, ..., N$. The function $\hat{f}$ should resemble f as closely as possible.

This last statement is intentionally vague, since any additional desired or required properties (e.g. continuity, differentiability, boundedness) of the function $\hat{f}$ are specific to the task at hand. Additionally, given a limited amount of computational resources, the function $\hat{f}$ should be easy to compute, fast to evaluate, and it should provide local control.

1.2 Basic Shepard's method

One of the simplest methods for scattered data interpolation is *inverse distance weighting* (IDW) with its basic form, the Shepard's method [She68]. Shepard's interpolation is the result of minimizing the following energy function:

$$E(x, y) = \left(\sum_{k=0}^{N} \frac{(y - y_k)^2}{d(x, x_k)^p} \right)^{\frac{1}{p}}. \tag{1}$$

Solving $\frac{\partial E}{\partial y} = 0$ yields the Shepard's interpolant:

$$\hat{f}(x) = \begin{cases} \frac{\sum_{k=1}^{N} w_k(x) y_k}{\sum_{k=1}^{N} w_k(x)}, & \text{if } d(x, x_k) \neq 0 \text{ for all } k, \\ y_k, & \text{if } d(x, x_k) = 0 \text{ for some } k, \end{cases} \tag{2}$$

where

$$w_k(x) = \frac{1}{d(x, x_k)^p} \tag{3}$$

for some distance metric d. The parameter $p > 0$ controls the shape of the interpolant, giving more or less weight to the nearest points.

1.3 Modified Shepard's method

The described basic form of the Shepard's method suffers from a major drawback: the sample points act globally, so consequently all of them have to be taken into account when computing a single interpolated value. By limiting the radius of influence of each sample we can achieve local control. Additionally, it is possible to accelerate the search for the nearest neighbors by superimposing a space partitioning structure. This approach is known as Modified Shepard's method [FN80, Ren88].

The weights in this interpolation scheme have to be modified accordingly:

$$w_k(x) = \left(\frac{\max(0, R - d(x, x_k))}{R d(x, x_k)} \right)^2, \tag{4}$$

where R is the radius of influence.

2 Assignment

In this assignment you will be given a set of N samples (x_k, y_k), from $\mathcal{D} = \mathbb{R}^3$, such that $x_k \in \mathbb{R}^3, y_k \in \mathbb{R}$. Your task is to write a program that will implement **both the basic and the modified Shepard's method** in order to interpolate the input samples and output a regular grid (a volume) that can be visualized in a volume rendering application (e.g. VPT). The program will be given a bounding box $x_{\min}, x_{\max} \in \mathbb{R}^3$ along with the resolution $r \in \mathbb{N}^3$ in order to construct the volume, and the parameters $p, R \in \mathbb{R}$ for the interpolation itself. The distance metric should be Euclidean. To accelerate the search for the nearest neighbors, **you have to arrange the input samples in an octree**. Technical details (octree maximum depth and bucket size) are up to you.

2.1 Input format

The scattered data will be passed over standard input in text form. Each line of the standard input will contain one input sample, consisting of four decimal values (three spatial coordinates and a function value).

All other parameters of the program — the method to be used (`basic` or `modified`), the parameters p and R, the bounding box and the resolution of the output volume — will be passed as command line parameters, namely `--method`, `--p`, `--r`, `--min-x`, `--min-y`, `--min-z`, `--max-x`, `--max-y`, `--max-z`, `--res-x`, `--res-y`, and `--res-z`.

You can assume the input will always be valid, so do not waste time on validation and error handling.

2.2 Output format

The output will be binary, consisting of the volume values as 32-bit floats. The values have to be sorted in ascending order, first by z, then by y, then by x. Hint: this is the result of three nested for loops:

```
for z from min-z to max-z in res-z steps:
    for y from min-y to max-y in res-y steps:
        for x from min-x to max-x in res-x steps:
            output interpolated value at (x, y, z)
        endfor
    endfor
endfor
```

2.3 Example input

Standard input:

```
2.312 6.223 1.232 0.222
-0.442 4.623 -2.233 3.659
0.212 -1.776 0.781 5.845
```

Command call:

```
./interpolate < input.txt > output.raw \
    --method modified --r 0.5 \
    --min-x -1.5 --min-y -1.5 --min-z -1 \
    --max-x  1.5 --max-y  1.5 --max-z  1 \
    --res-x 128 --res-y 128 --res-z 64
```

2.4 Visualizing the data

This step is optional, but it will help with debugging.

To actually see your interpolated data, you can, for example, output each slice as an image. Hint: the PPM format is extremely easy to write. On the other hand, to visualize the whole volume in 3D, you can use VPT (`http://lgm.fri.uni-lj.si/ziga/vpt/`).

In both cases, the values have to be of encoded as uint8 instead of float32, so you will have to cast them accordingly. Before casting, you may want to change the range of the floats.

2.5 Grading

This assignment is worth 10 points:

- 5 points for the basic Shepard's method, and
- 5 points for the modified Shepard's method and octree acceleration.

Input parsing has to work in both cases, otherwise the assignment cannot be graded.

References

[FN80] Richard Franke and Greg Nielson. Smooth interpolation of large sets of scattered data. *International Journal for Numerical Methods in Engineering*, 15(11):1691–1704, 1980.

[Ren88] Robert J. Renka. Multivariate interpolation of large sets of scattered data. *ACM Trans. Math. Softw.*, 14(2):139–148, June 1988.

[She68] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, ACM '68, page 517–524, New York, NY, USA, 1968. Association for Computing Machinery.