Hermite interpolation of heightmaps

Róbert Bán
rob.ban@inf.elte.hu

Gábor Valasek
valasek@inf.elte.hu

Eötvös Loránd University, Budapest, Hungary

1. Introduction

The goal of this research is to extend heightmap rendering and shading. The heightmap is a real-valued function over a two-dimensional domain, describing geometric detail over an underlying coarser surface. It is usually encoded as a texture. Displacement mapping uses the heightmap to determine the positions of a fine vertex grid. In contrast, parallax mapping renders the coarser surface but adjusts the lighting according to the heightmap. See [1] for a detailed survey on these methods.

2. Problem statement

Heightmaps are stored in 2D textures: height values and normal XY.

Bilinear filtering results in a C^0-continuous surface.

For a smooth output, we might need to increase the resolution considerably.

Our goal is to generate smooth, C^1-continuous heightfields without increasing the storage requirements.

3. Hermite heightmap

Store partial derivatives/normals in samples.

Use Hermite interpolation instead of bilinear interpolation inside of the cells.

The reconstructed surface is C^1-continuous.

For the improved continuity we need more stored data, but it is often already stored in the form of normal vectors.

4. Implementation

The stored partial derivatives can be calculated with numerical differentiation methods, such as central differences, or automatic differentiation if the exact function is known.

The four surrounding samples are gathered upon sampling, which are then interpolated with two-dimensional Hermite basis functions.

The basis functions are evaluated at the local coordinates of the sampling position in the cell.

Our C++ sample implementation is available at `https://github.com/Bundas102/falcor-hermite-heightmap`.

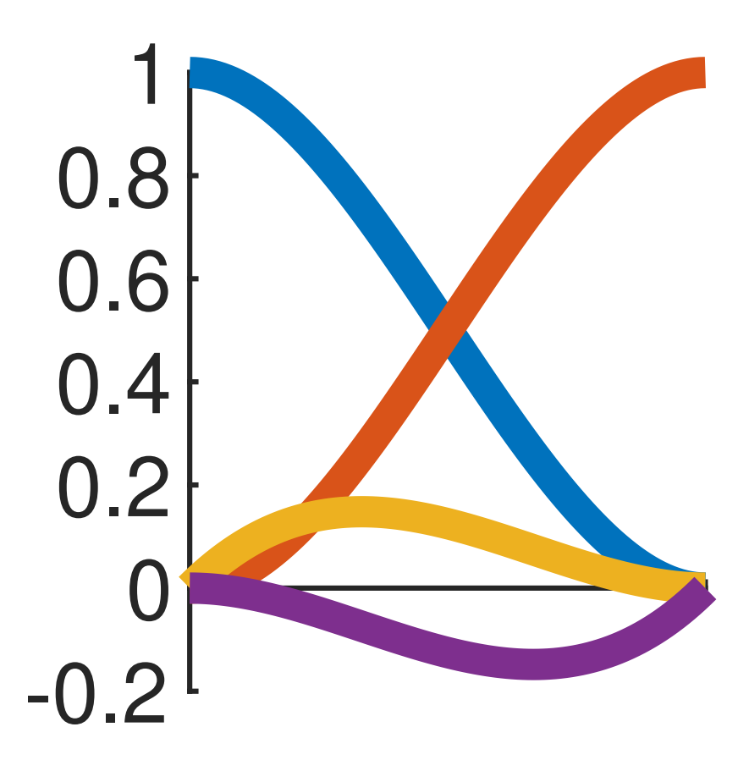
5. Interpolation details

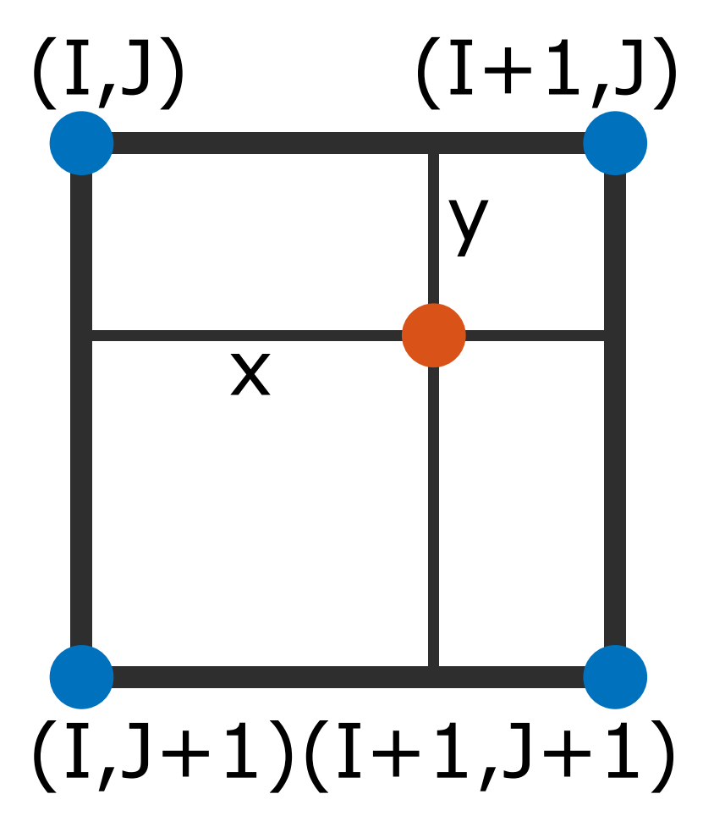
The heightmap is stored in a texture as a matrix of samples: $F : \{0..N\} \times \{0..M\} \to \mathbb{R}^3$. For an $f : [0,1]^2 \to \mathbb{R}$ input function, we store the following values in F ($i \in \{0..N\}, j \in \{0..M\}$):

$$F[i][j][0] = f\left(\tfrac{i}{N}, \tfrac{j}{M}\right), \quad F[i][j][1] = \partial_x f\left(\tfrac{i}{N}, \tfrac{j}{M}\right), \quad F[i][j][2] = \partial_y f\left(\tfrac{i}{N}, \tfrac{j}{M}\right).$$

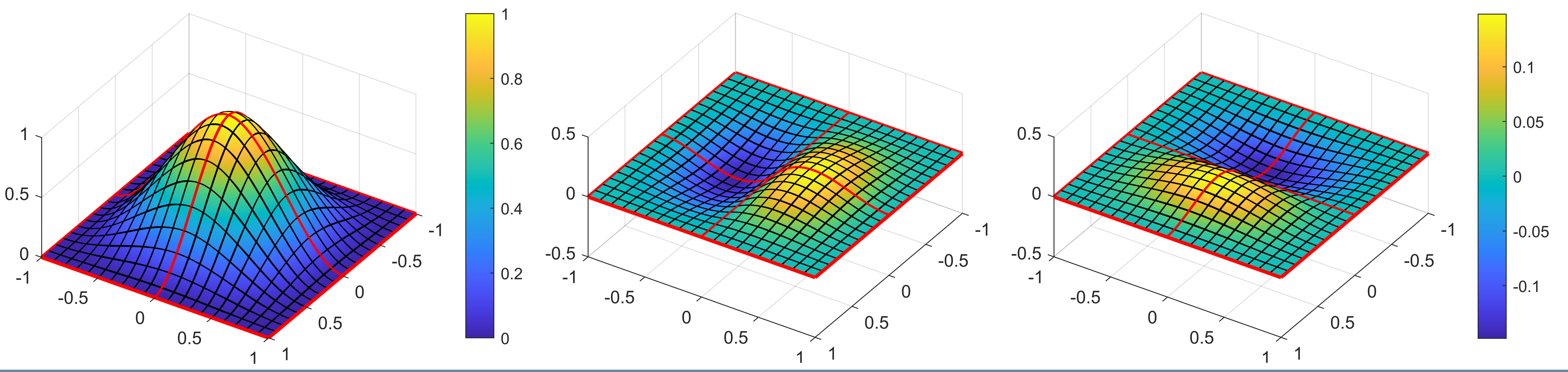
Let $(u,v) \in [0,1]^2$ be the sampling position. Then calculate $(U,V) = (Nu, Mv)$ scaled coordinates, $(I,J) = (\lfloor U \rfloor, \lfloor V \rfloor)$ cell indices and $(x,y) = (U - \lfloor U \rfloor, V - \lfloor V \rfloor)$ local coordinates. Using the 1D cubic Hermite basis polynomials, the value of the reconstructed function is computed as

$$\begin{aligned}\hat{f}(u,v) = \sum_{i=0}^{1}\sum_{j=0}^{1}(&F[I+i][J+j][0]h_i^0(x)h_j^0(y)+\\ &F[I+i][J+j][1]h_i^1(x)h_j^0(y)+\\ &F[I+i][J+j][2]h_i^0(x)h_j^1(y)).\end{aligned}$$





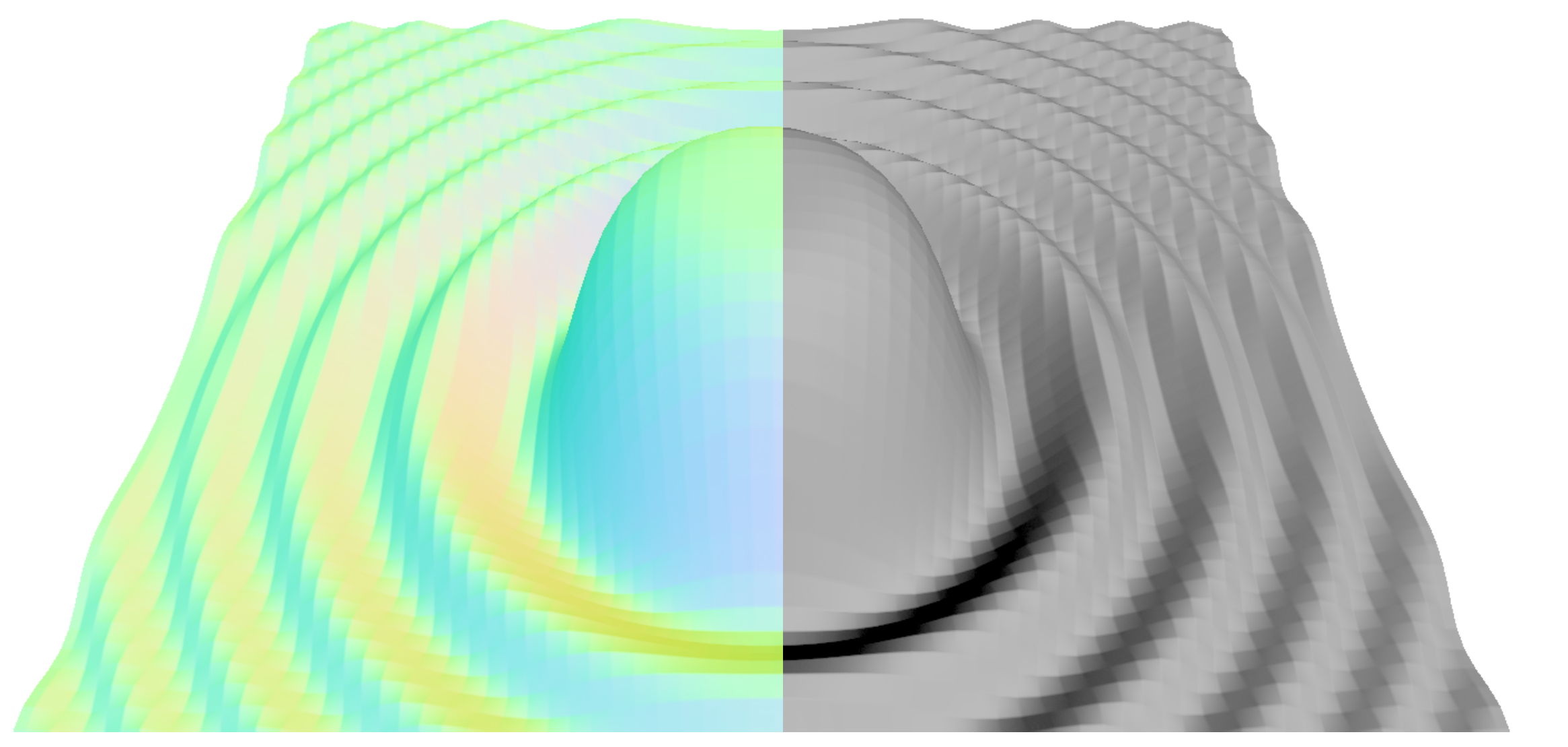
The two-dimensional Hermite basis functions are the result of the tensor product of the one-dimensional Hermite basis functions.

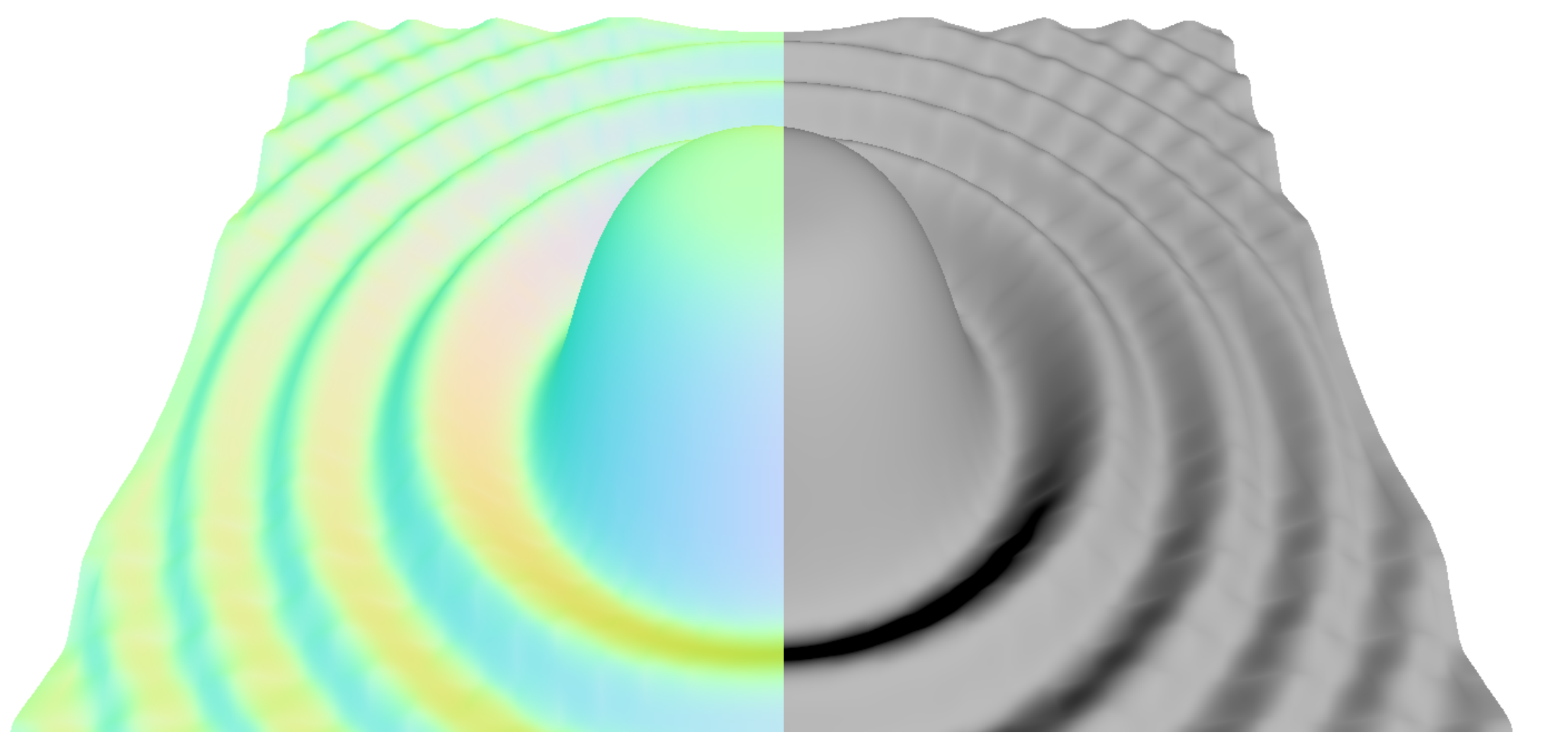


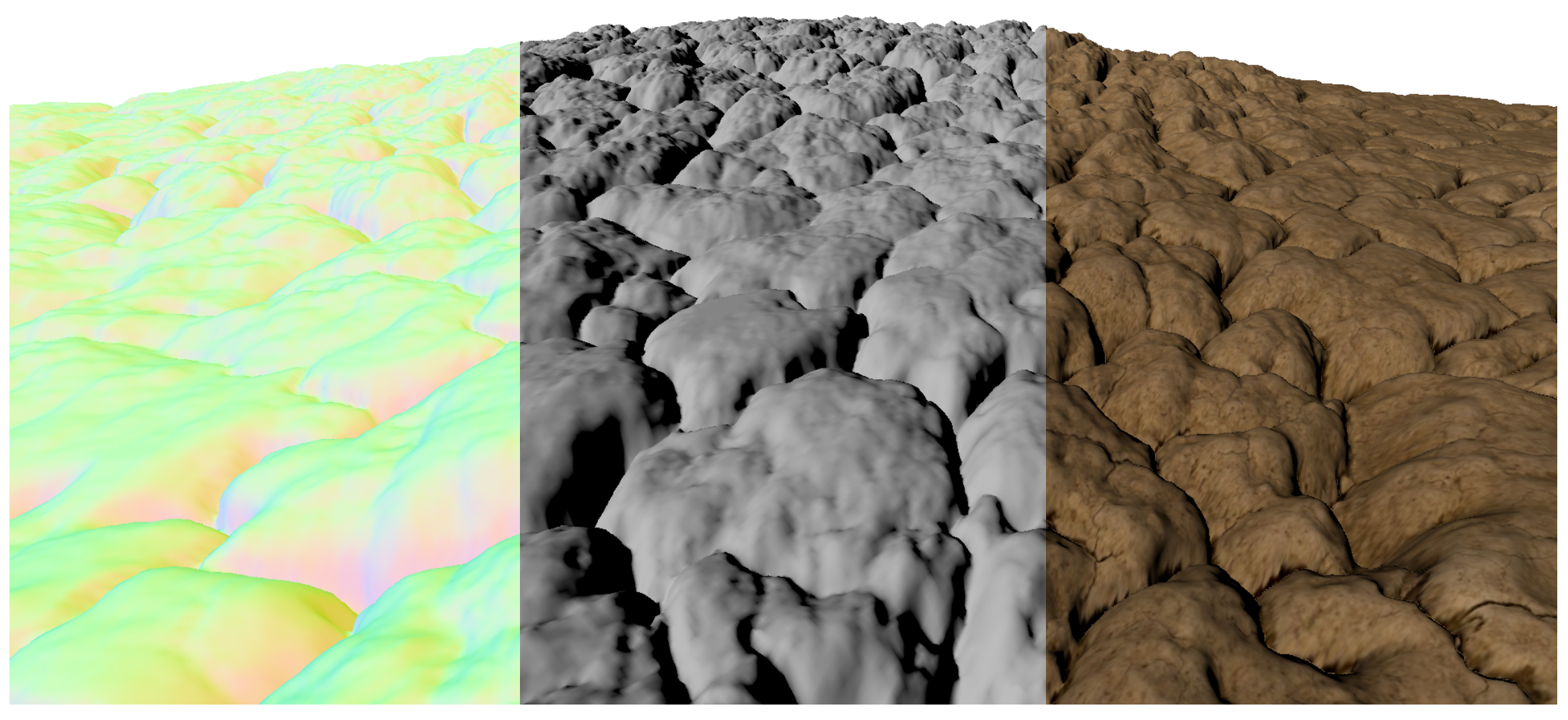
6. Results

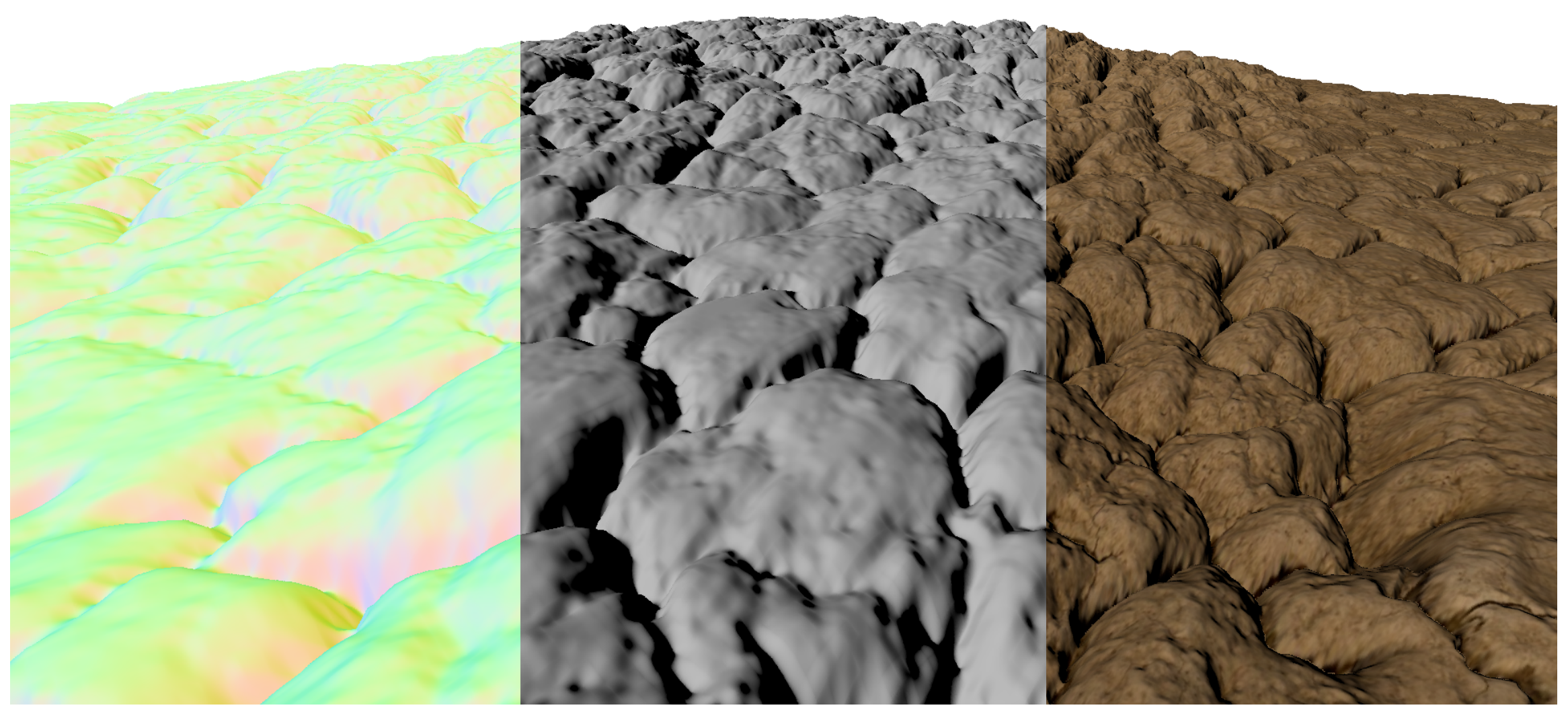
The figures below show comparisons between traditional bilinear and the proposed Hermite interpolation techniques. The latter achieves similar visuals to higher resolution bilinear filtering. The table of render times were measured on a desktop AMD RX 5700 at full HD resolution, using 32 relaxed cone map steps and 5 binary search iterations. Bilinear heightfield with Hermite normals (*H.normal*) achieved similar visual quality to Hermite interpolation of both the heightmap and normals. As such, we propose to use bilinear heightfields with Hermite interpolated normals for maximum performance and quality. This also facilitates the use of lower resolution heightfields.

Map res.	Bilinear	Hermite	H.normal
128^2	0.17 ms	0.42 ms	0.21 ms
256^2	0.20 ms	0.44 ms	0.24 ms
512^2	0.28 ms	0.49 ms	0.31 ms


64×64 bilinear – 4096 values


32×32 Hermite – 3072 values


512×512 bilinear


256×256 Hermite

7. Conclusion

We proposed a method for high quality heightmap rendering. Our heightmap representation is best suited for rendering smooth surfaces. As height values are usually stored along their geometric surface normals, our proposed method incurs no additional storage cost. Our method does require manual filtering of the data and extra calculations to evaluate the Hermite basis polynomials to compute the interpolated result. This additional cost may be reduced by overlapping arithmetic work while texture data is being transferred.

8. References

[1] László Szirmay-Kalos and Tamás Umenhoffer. Displacement mapping on the GPU - state of the art. *Comput. Graph. Forum*, 27(6):1567–1592, 2008.

9. Acknowledgement

We would like to thank Visual Concepts for providing the AMD GPU used in the tests.

Supported by the ÚNKP-21-3 New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund.

National Research, Development and Innovation Office
Hungary

Új Nemzeti Kiválóság Program