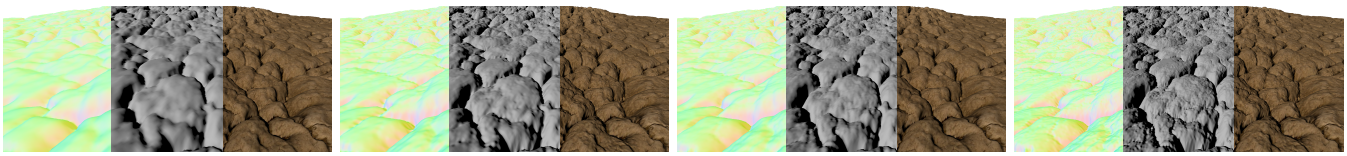
Hermite interpolation of heightmaps

Róbert Bán and Gábor Valasek

Eötvös Loránd University, Hungary
{rob.ban,valasek}@inf.elte.hu



(a) 256 × 256 *bilinear* **(b)** 256×*256 Hermite* **(c)** 512 × 512 *bilinear* **(d)** 512 × 512 *Hermite*

Figure 1: *Comparison of the usual bilinear interpolation and Hermite interpolation. Left: normal, middle: diffuse, right: shaded.*

Abstract

Heightmaps are ubiquitous in real-time computer graphics. They are used to describe geometric detail over an underlying coarser surface. Various techniques, such as parallax occlusion mapping and relief mapping, use heightmap textures to impose mesostructural details over macrostructural elements without increasing the actual complexity of the rendered geometries. We aim to improve the quality of the fine resolution surface by incorporating the gradient of the original function into the sampling procedure. The traditional representation consists of simple height values stored on a regular grid. During rendering, bilinear filtering is applied. We propose to store the partial derivatives with the height values and use Hermite interpolation between the samples. This guarantees a globally C^1 continuous heightfield instead of the C^0-continuity of bilinear filtering. Moreover, incorporating higher order information via partial derivatives allows us to use lower resolution heightmaps while retaining the appearance of a higher resolution map. In parallax mapping, surface normals are often stored alongside the height values, as such, our method does not require additional storage, since normals and partial derivatives can be calculated from one another. The exact normals of the reconstructed cubic Hermite surface can also be calculated, resulting in a storage efficient replacement for normal mapping with richer visual appearance.

CCS Concepts

• ***Computing methodologies*** → *Rendering; Shape modeling;* • ***Mathematics of computing*** → *Continuous functions;*

1. Introduction

Heightmaps are ubiquitous in real-time computer graphics. 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 [SKU] for a detailed survey on these methods.

In the traditional representation, heightmaps are stored as simple height values in textures. The texture is sampled with bilinear interpolation as it is accelerated in hardware. Its most significant disadvantage is that the reconstructed surface is only C^0-continuous, which means that for a smooth output, we might need to increase the resolution considerably. Our primary goal is to increase the smoothness of the result without significantly increasing storage.

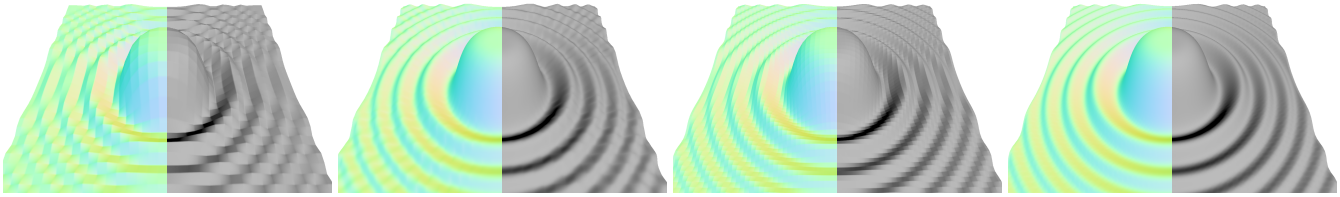
2. Hermite heightmap

We extend the usual height samples with the partial derivatives of the original function. This means that instead of only a height value, we store the tangent plane at every sample position. Bilinear interpolation is replaced by Hermite interpolation on the cells. The advantage is that the reconstructed surface is guaranteed to be C^1-continuous. In exchange for the higher continuity, we have to store more data – three times as much to be precise, but often this information is already available in the form of normal vectors.

Our heightmaps store the height values and their partial derivatives with respect to the axes of the texture plane. For the latter, one could use numerical differentiation methods, such as central differences, or automatic differentiation if the exact function is known.

Sample data are gathered from the underlying cell, which is then interpolated with two-dimensional Hermite basis functions. The basis functions are evaluated at the local coordinates of the sampling position in the cell.

DOI: 10.2312/egp.20221017



(a) 32×32 *bilinear – 1024 values* **(b)** 32×32 *Hermite – 3072 values* **(c)** 64×64 *bilinear – 4096 values* **(d)** 64×64 *Hermite – 12288 values*

Figure 2: *Comparison of the usual bilinear interpolation and Hermite interpolation. Left: normal vector, right: diffuse shading. The bilinear case shows the exact normals of the rendered geometry instead of a filtered one. Hermite normals are exact.*

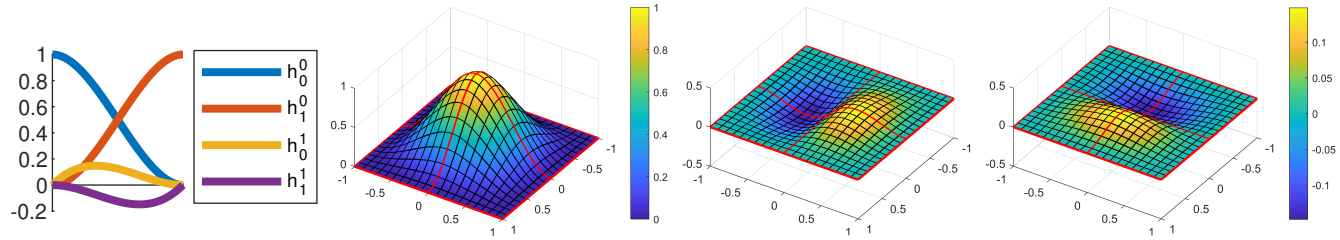


Figure 3: *The one and two-dimensional Hermite basis functions. The two-dimensional basis functions are the tensor product of the one-dimensional functions. The basis functions in order correspond to the function value and the two partial derivatives ($\partial_x f$ and $\partial_y f$).*

3. 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 ($i \in \{0..N\}, j \in \{0..M\}$):

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

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 1D cubic Hermite basis, the reconstructed function is

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

The two-dimensional Hermite basis functions are the result of the tensor product of the one-dimensional Hermite basis functions shown on Figure 3. The partial derivatives are then calculated, and the normal vector is $[1,0,\partial_x \hat{f}]^T \times [0,1,\partial_y \hat{f}]^T = [-\partial_x \hat{f}, -\partial_y \hat{f}, 1]^T$.

4. Results

Figures 1 and 2 show comparisons between traditional bilinear and the proposed Hermite interpolation techniques. The latter achieves similar visuals to higher resolution bilinear filtering. The following table of render times were measured on a desktop AMD RX 5700 at full HD resolution in the scene of Figure 1, using 32 relaxed cone map steps and 5 binary search iterations. Render times are in milliseconds. *Bilinear* is traditional storage with normal mapping, *Hermite* is calculated with the proposed method and *H.normal* uses traditional bilinear interpolation for intersection search but Hermite interpolation for shading normals.

Heightmap res.	Bilinear	Hermite	H.normal
128×128	0.17 ms	0.42 ms	0.21 ms
256×256	0.20 ms	0.44 ms	0.24 ms
512×512	0.28 ms	0.49 ms	0.31 ms

Bilinear heightfield with Hermite normals 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.

5. 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.

Acknowledgement 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. We would like to thank Visual Concepts for providing the AMD GPU.

References

[SKU] SZIRMAY-KALOS L., UMENHOFFER T.: Displacement mapping on the GPU - state of the art. *Comput. Graph. Forum*, 6, 1567–1592. 1