King Abdullah University of Science and Technology

KAUST

AMCS / CS 247 – Scientific Visualization
Lecture 9: Iso-Surface Lighting, Pt. 3

Markus Hadwiger, KAUST

Reading Assignment #5 (until Oct. 8)

Read (required):

- Real-Time Volume Graphics, Chapter 2 (*GPU Programming*)
- Real-Time Volume Graphics, Chapter 5.3 (*Gradient-Based Illumination*)
- Real-Time Volume Graphics, Chapter 5.4.1 (*Blinn-Phong Illumination*)

The Gradient as Normal Vector

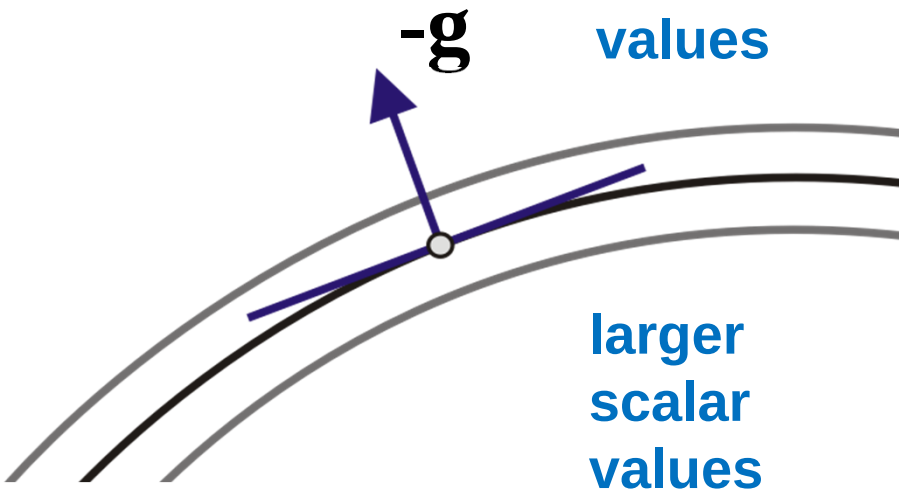
Gradient of the scalar field points in direction of highest rate of change

$$\mathbf{g} = \nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)^{\mathbf{T}}$$

Local approximation to isosurface at any point:
tangent plane = plane orthogonal to gradient

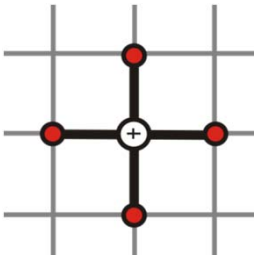
Unit normal of this isosurface:
normalized gradient

$$\mathbf{n} = -\mathbf{g}/|\mathbf{g}|$$



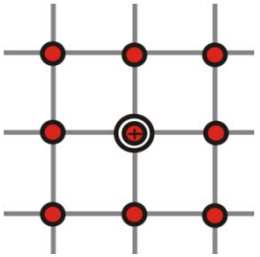
Gradient Reconstruction

We need to reconstruct the derivatives of a continuous function given as discrete samples



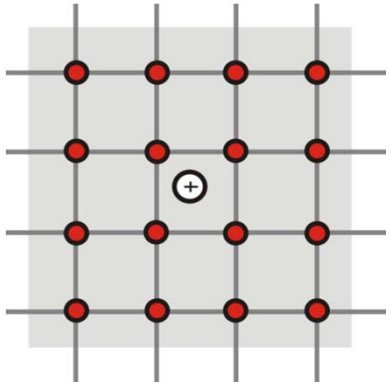
Central differences

- Cheap and quality often sufficient (2+2+2 neighbors in 3D)



Discrete convolution filters on grid

- Image processing filters; e.g. Sobel (3x3x3 neighbors)



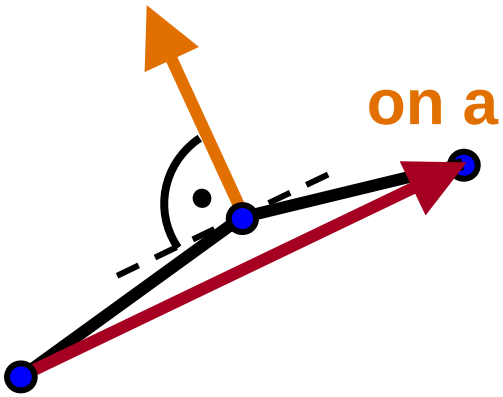
Continuous convolution filters

- Derived continuous reconstruction filters
- E.g., the cubic B-spline and its derivatives (4x4x4 neighbors)

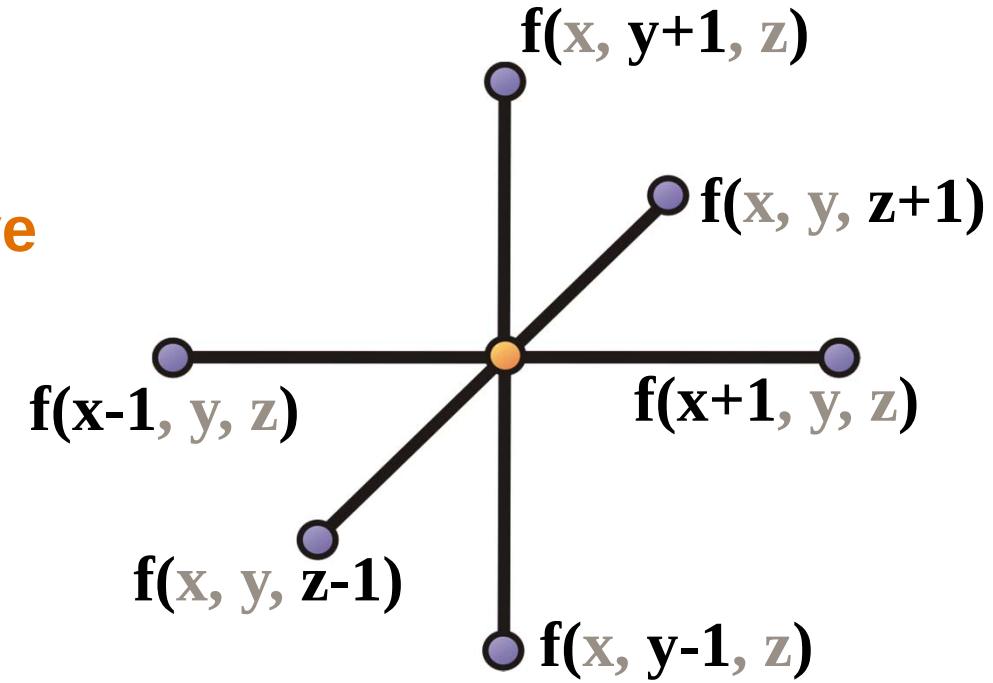
Central Differences

Need only two neighboring voxels per derivative

Most common method



on a curve



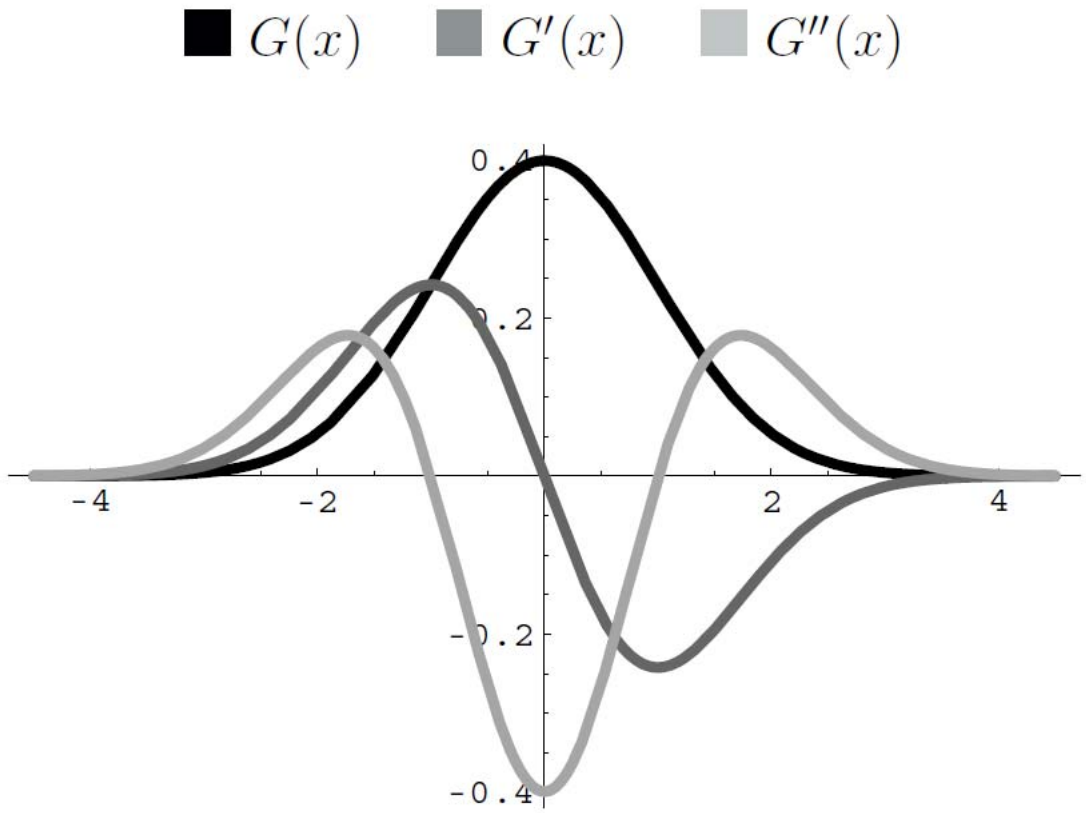
g_x = 0.5(f(x+1, y, z) – f(x-1, y, z))

g_y = 0.5(f(x, y+1, z) – f(x, y-1, z))

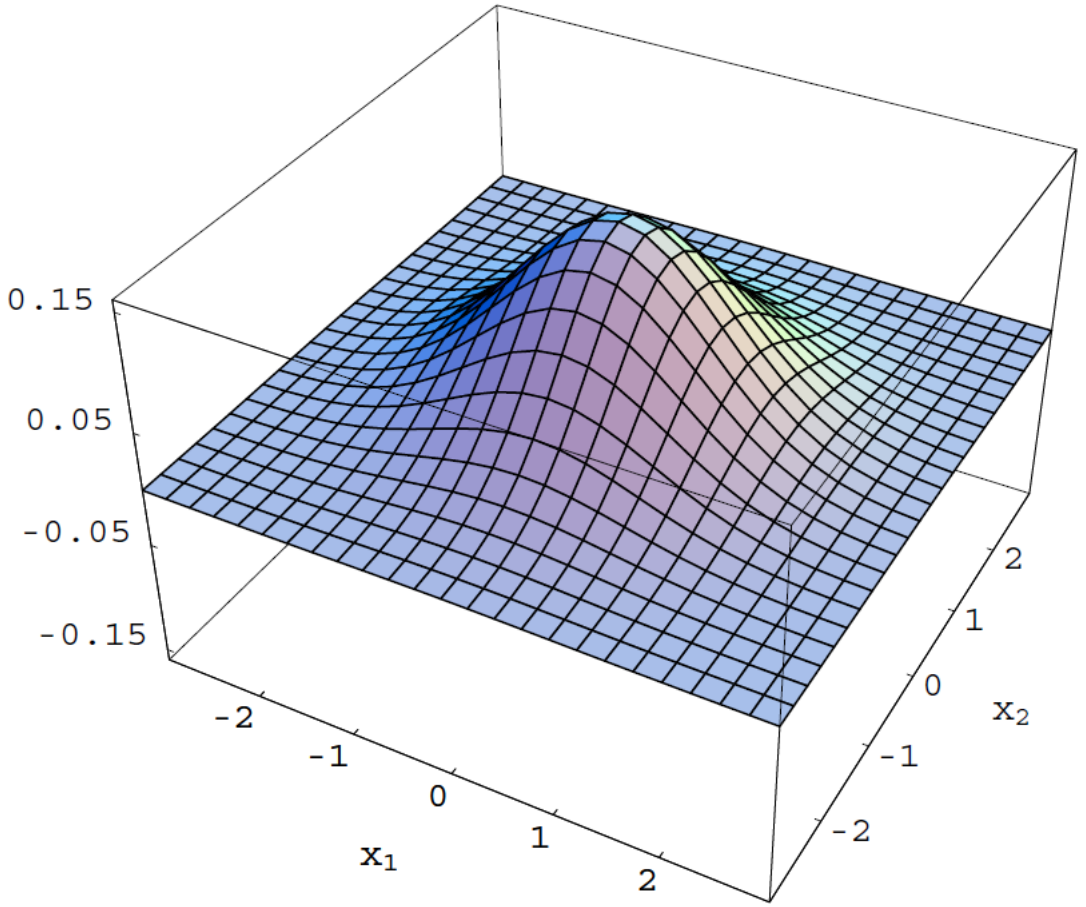
g_z = 0.5(f(x, y, z+1) – f(x, y, z-1))

in a volume

Example: Gaussian Kernel (1D)

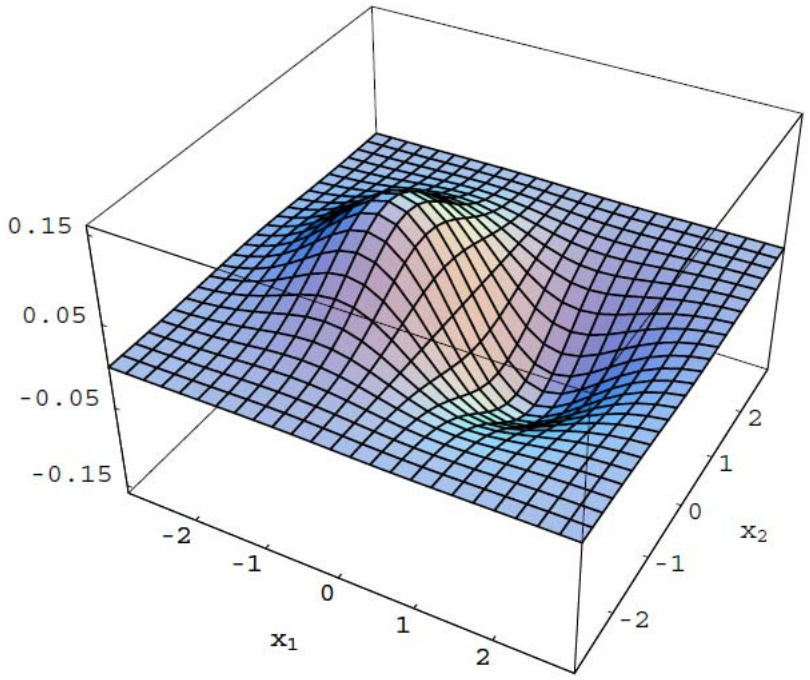


2D Gaussian Kernel

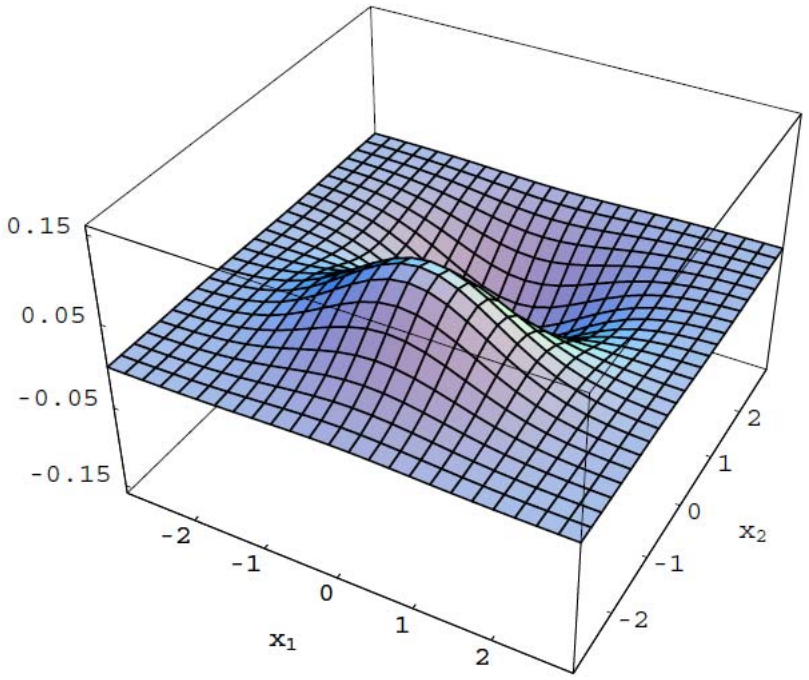


$$G(x_1, x_2)$$

2D Gaussian Derivative Kernels



$\partial G(x_1, x_2)/\partial x_1$



$\partial G(x_1, x_2)/\partial x_2$

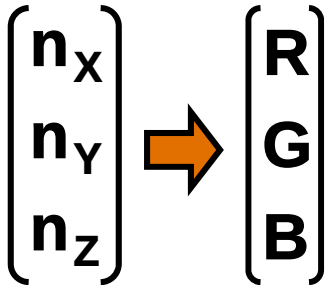
Pre-Computed Gradients/Normal Vectors

Pre-compute gradients/normals at grid points with any method

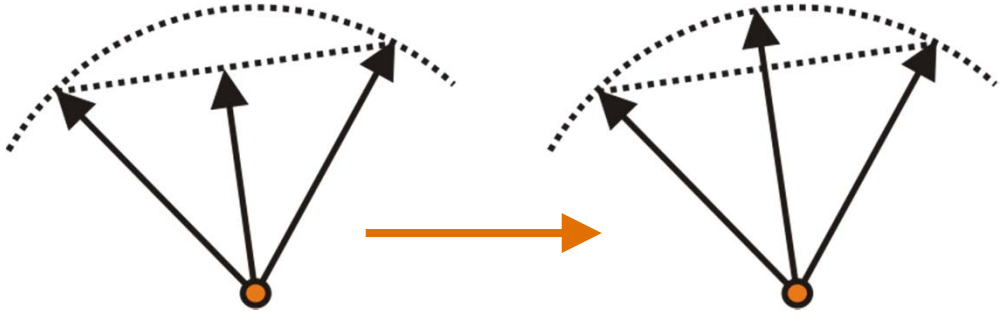
Store normalized gradient/normals directions in RGB texture

Sample gradient texture in fragment shader: (bi/tri-)linear interpolation

Re-normalize after interpolation (after texture fetch)!



RGB gradient texture



lerp of texture filter **renormalize!**

Thank you.

Thanks for material

- Helwig Hauser
- Eduard Gröller
- Daniel Weiskopf
- Torsten Möller
- Ronny Peikert
- Philipp Muigg
- Christof Rezk-Salama