King Abdullah University of Science and Technology KAUST

CS 247 – Scientific Visualization
Lecture 11: Scalar Fields, Pt. 7

Markus Hadwiger, KAUST

Reading Assignment #6 (until Mar 8)

Read (required):

- Real-Time Volume Graphics, Chapter 2
  (*GPU Programming*)
- Real-Time Volume Graphics, Chapters 5.5 and 5.6 (you already had to read - 5.4)
  (*Local Volume Illumination*)
- Refresh your memory on eigenvectors and eigenvalues:
  `https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors`

Gradient and Directional Derivative

Gradient $\nabla f(x,y,z)$ of scalar function $f\left(x,y,z\right)$: (in Cartesian coordinates)

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

(Cartesian vector components; basis vectors not shown)

But: always need **basis vectors**! With Cartesian basis:

$$\nabla f(x,y,z) = \frac{\partial f(x,y,z)}{\partial x}\,\mathbf{i} + \frac{\partial f(x,y,z)}{\partial y}\,\mathbf{j} + \frac{\partial f(x,y,z)}{\partial z}\,\mathbf{k}$$

What about the Basis?

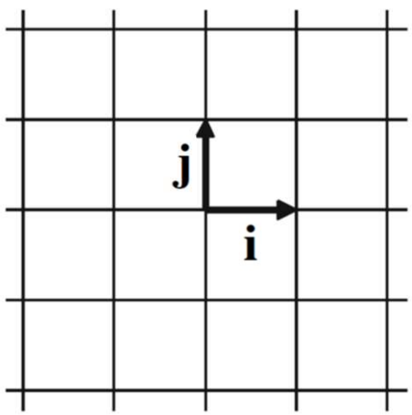
On the previous slide, this actually meant

$$\nabla f(x,y,z) = \frac{\partial f(x,y,z)}{\partial x}\,\mathbf{i}(x,y,z) + \frac{\partial f(x,y,z)}{\partial y}\,\mathbf{j}(x,y,z) + \frac{\partial f(x,y,z)}{\partial z}\,\mathbf{k}(x,y,z)$$

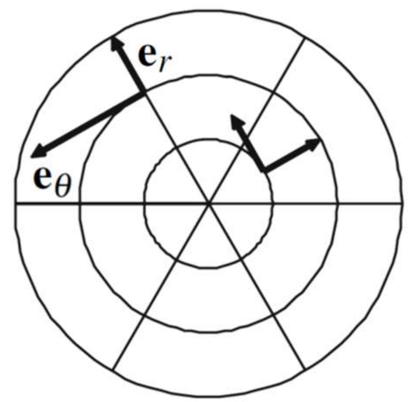
It's just that the Cartesian basis vectors are the same everywhere...

But this is not true for many other coordinate systems:

Cartesian coordinates



polar coordinates



What about the Basis?

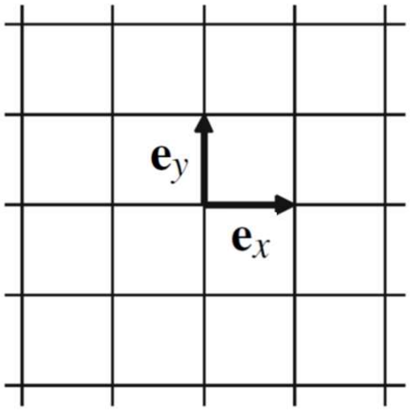
On the previous slide, this actually meant

$$\nabla f(x,y,z) = \frac{\partial f(x,y,z)}{\partial x}\,\mathbf{i}(x,y,z) + \frac{\partial f(x,y,z)}{\partial y}\,\mathbf{j}(x,y,z) + \frac{\partial f(x,y,z)}{\partial z}\,\mathbf{k}(x,y,z)$$

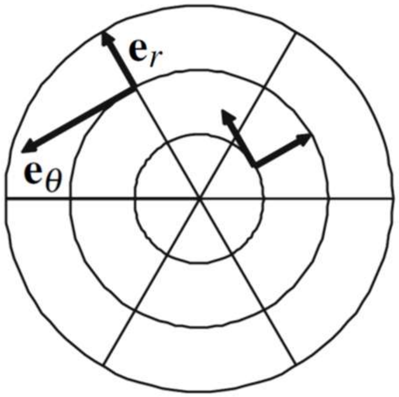
It's just that the Cartesian basis vectors are the same everywhere...

But this is not true for many other coordinate systems:

Cartesian coordinates



polar coordinates



The Gradient as a Differential Form

The gradient as a *differential* (differential 1-form) is the “primary” concept

$$df = \frac{\partial f}{\partial x}\,dx + \frac{\partial f}{\partial y}\,dy + \frac{\partial f}{\partial z}\,dz$$

A differential 1-form is a scalar-valued linear function that takes a (direction) vector as input, and gives a scalar as output

Each of the 1-forms df, dx, dy, dz takes a (direction) vector as input, gives scalar as output

In the expression of the gradient df above, all 1-forms on the right-hand side get the same vector as input

df is simply a linear combination of the coordinate differentials dx, dy, dz

The Gradient as a Differential Form

The gradient as a *differential* (differential 1-form) is the “primary” concept

$$df = \frac{\partial f}{\partial x}\,dx + \frac{\partial f}{\partial y}\,dy + \frac{\partial f}{\partial z}\,dz$$

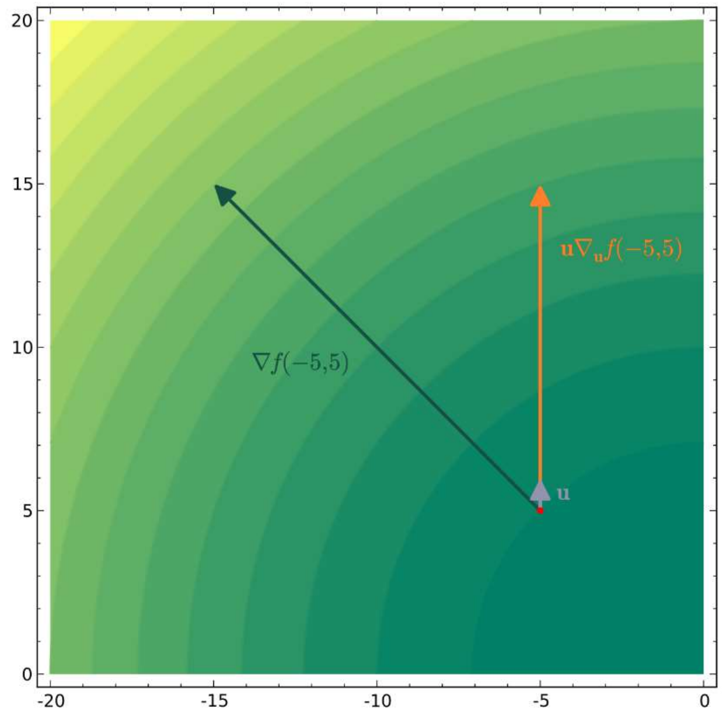
The directional derivative and the gradient vector

$$D_{\mathbf{u}} f = df(\mathbf{u})$$
$$df(\mathbf{u}) = \nabla f \cdot \mathbf{u}$$

The gradient vector is then *defined*, such that:

$$\nabla f \cdot \mathbf{u} := df(\mathbf{u})$$

Gradient Vectors and Differential 1-Forms

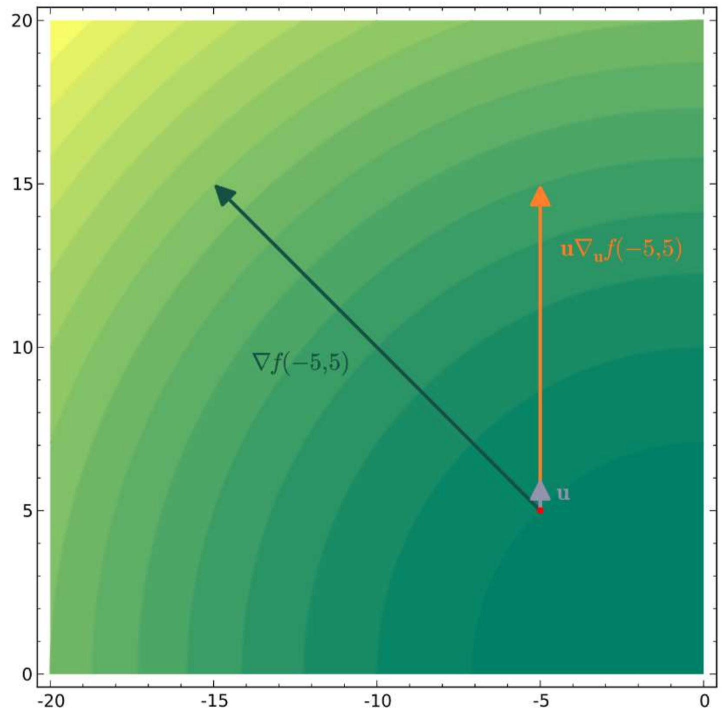


from Wikipedia (for **u** a unit vector),

the function here is $f(x,y) = x^2 + y^2$

$$\nabla f(x,y) = 2x\,\mathbf{i} + 2y\,\mathbf{j}$$

Gradient Vectors and Differential 1-Forms



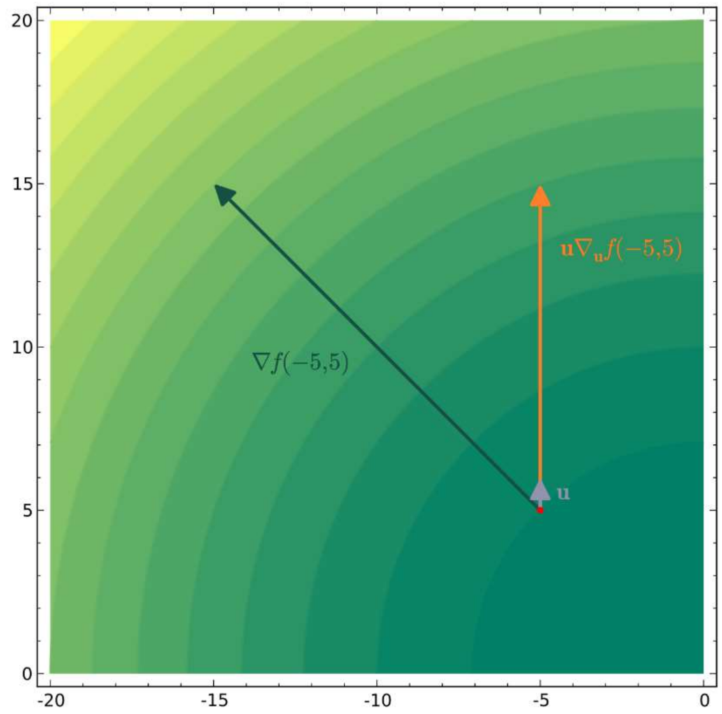
from Wikipedia (for **u** a unit vector),

the function here is $f(x,y) = x^2 + y^2$

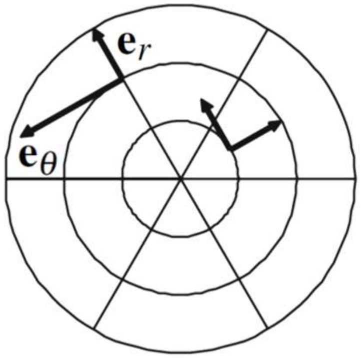
$$\nabla f(x,y) = 2x\,\mathbf{e}_x + 2y\,\mathbf{e}_y$$

$$df(x,y) = 2x\,dx + 2y\,dy$$

Gradient Vectors and Differential 1-Forms



how about in polar coordinates?



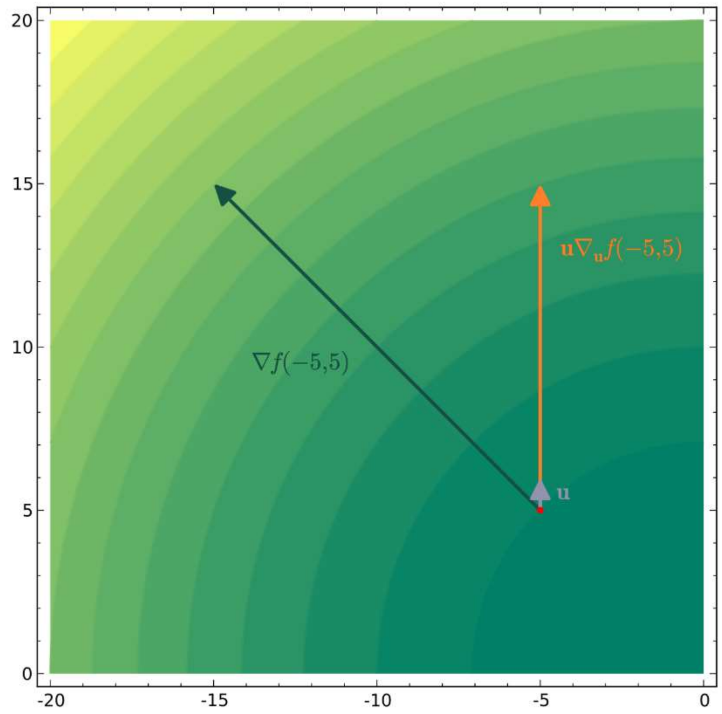
from Wikipedia (for **u** a unit vector),

the function here is $f(r,\theta) = r^2$

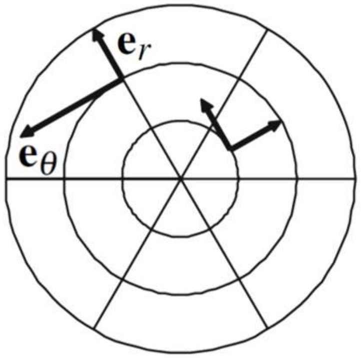
$$\nabla f(r,\theta) = 2r\,\mathbf{e}_r + 0\frac{1}{r^2}\,\mathbf{e}_\theta = 2r\,\mathbf{e}_r$$

$$df(r,\theta) = 2r\,dr + 0\,d\theta = 2r\,dr$$

Gradient Vectors and Differential 1-Forms



how about in polar coordinates?



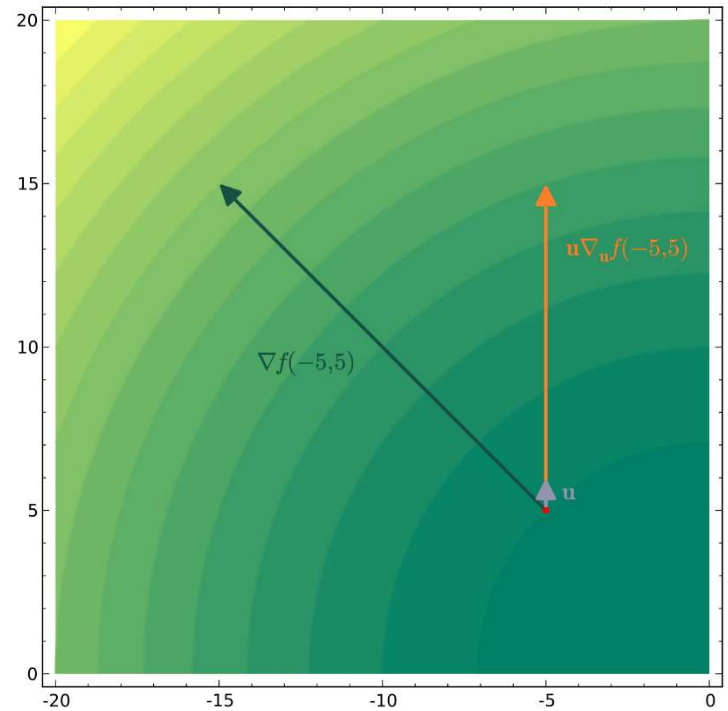
from Wikipedia (for **u** a unit vector),

the function here is $f(r,\theta) = r^2$

$$\nabla f(r,\theta) = 2r\,\mathbf{e}_r + 0\frac{1}{r^2}\mathbf{e}_\theta = 2r\,\mathbf{e}_r$$

$$df(r,\theta) = 2r\,dr + 0\,d\theta = 2r\,dr$$

Gradient Vectors and Differential 1-Forms

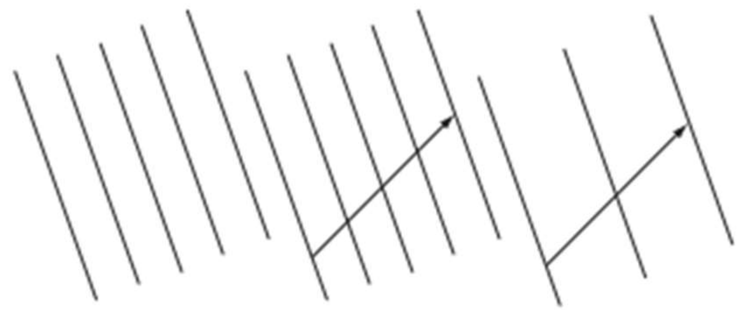


from Wikipedia (for **u** a unit vector),
the function here is $f(r,\theta) = r^2$

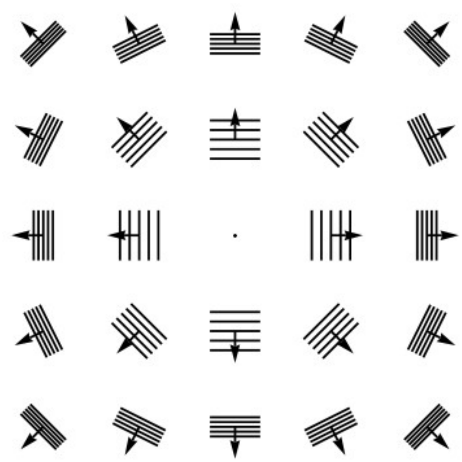
$$\nabla f(r,\theta) = 2r\,\mathbf{e}_r + 0\frac{1}{r^2}\,\mathbf{e}_\theta = 2r\,\mathbf{e}_r$$

$$df(r,\theta) = 2r\,dr + 0\,d\theta = 2r\,dr$$

different 1-forms
evaluated in some direction



1-form (field) df



Interlude: Tensor Calculus

In tensor calculus, first-order tensors can be

- Contravariant $\mathbf{v} = v^i \, \mathbf{e}_i$
- Covariant $\boldsymbol{\omega} = v_i \, \boldsymbol{\omega}^i$

The gradient vector is a contravariant vector $\mathbf{v} = v^i \, \boldsymbol{\partial}_i$

The gradient 1-form is a covariant vector (a covector) $df = \frac{\partial f}{\partial x^i} dx^i$

Very powerful; necessary for non-Cartesian coordinate systems

On (intrinsically) curved manifolds (sphere, ...):
Cartesian coordinates not even possible

Interlude: Tensor Calculus

In tensor calculus, first-order tensors can be

- Contravariant $\mathbf{v} = v^i \, \mathbf{e}_i$
- Covariant $\boldsymbol{\omega} = v_i \, \boldsymbol{\omega}^i$

The gradient vector is a contravariant vector $\mathbf{v} = v^i \, \boldsymbol{\partial}_i$

The gradient 1-form is a covariant vector (a covector) $df = \frac{\partial f}{\partial x^i} \, dx^i$

This is also the fundamental reason why in graphics a normal vector transforms differently: as a covector, not as a vector!

(typical graphics rule: **n** transforms with transpose of inverse matrix)

Metric Tensor (Field)

Symmetric second-order tensor field: *Defines* inner product

$$\langle \mathbf{v}, \mathbf{w} \rangle := \mathbf{g}(\mathbf{v}, \mathbf{w}) = \|\mathbf{v}\| \|\mathbf{w}\| \cos\theta$$

$$\begin{aligned} \|\mathbf{v}\|^2 &= \langle \mathbf{v}, \mathbf{v} \rangle \\ &= \mathbf{g}(\mathbf{v}, \mathbf{v}) \\ &= g_{ij} v^i v^j \\ &= \mathbf{v}^T \mathbf{g} \, \mathbf{v} \end{aligned}$$

Metric Tensor (Field)

Symmetric second-order tensor field: *Defines* inner product

$$\langle \mathbf{v}, \mathbf{w} \rangle := \mathbf{g}(\mathbf{v}, \mathbf{w}) = \|\mathbf{v}\| \|\mathbf{w}\| \cos\theta$$

(2D)

$$\begin{aligned} \|\mathbf{v}\|^2 &= \langle \mathbf{v}, \mathbf{v} \rangle \\ &= \mathbf{g}(\mathbf{v}, \mathbf{v}) \\ &= g_{ij} v^i v^j \\ &= \mathbf{v}^T \mathbf{g}\, \mathbf{v} \end{aligned}$$

$$\mathbf{g} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}$$

$$\|\mathbf{v}\|^2 = \begin{bmatrix} v^1 & v^2 \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} v^1 \\ v^2 \end{bmatrix}$$

Metric Tensor (Field)

Symmetric second-order tensor field: *Defines* inner product

$$\langle \mathbf{v}, \mathbf{w} \rangle := \mathbf{g}(\mathbf{v}, \mathbf{w}) = \|\mathbf{v}\| \|\mathbf{w}\| \cos\theta$$

(2D)

$$\begin{aligned} \|\mathbf{v}\|^2 &= \langle \mathbf{v}, \mathbf{v} \rangle \\ &= \mathbf{g}(\mathbf{v}, \mathbf{v}) \\ &= g_{ij} v^i v^j \\ &= \mathbf{v}^T \mathbf{g}\, \mathbf{v} \end{aligned}$$

$$\mathbf{g} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}$$

$$\|\mathbf{v}\|^2 = \begin{bmatrix} v^1 & v^2 \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} v^1 \\ v^2 \end{bmatrix}$$

Cartesian coordinates:

$$\mathbf{g} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\|\mathbf{v}\|^2 = \begin{bmatrix} v^1 & v^2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v^1 \\ v^2 \end{bmatrix} = \mathbf{v}^T \mathbf{v}$$

Metric Tensor (Field)

Components referred to coordinates

$$g_{ij} := \langle \mathbf{e}_i, \mathbf{e}_j \rangle$$

A second-order tensor field is bi-linear, i.e.,
linear in each (vector) argument separately

Therefore, we immediately get:

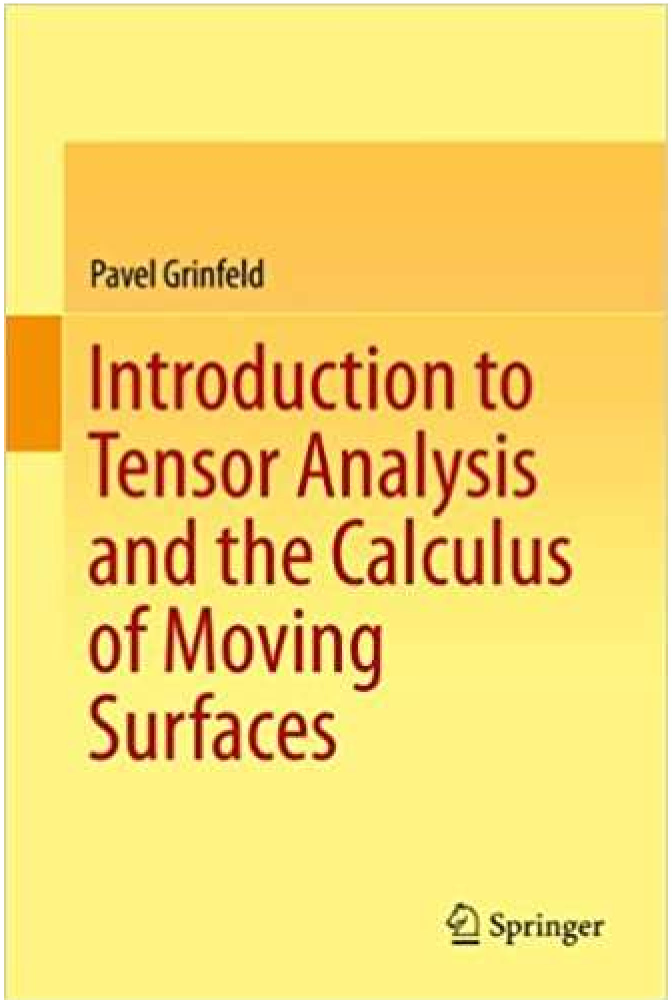
$$\begin{aligned} \mathbf{g}(\mathbf{v}, \mathbf{v}) &= \mathbf{g}\left(v^i \mathbf{e}_i, v^j \mathbf{e}_j\right) \\ &= v^i v^j \mathbf{g}\left(\mathbf{e}_i, \mathbf{e}_j\right) \\ &= g_{ij} v^i v^j \end{aligned}$$

Tensor Calculus

Highly recommended:

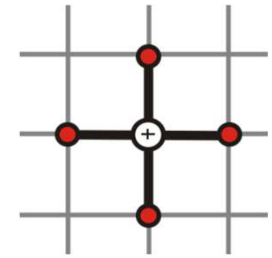
Very nice book,

complete lecture on Youtube!



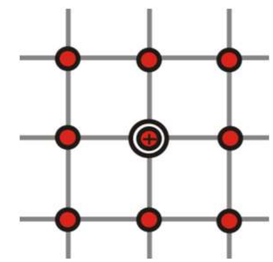
(Numerical) Gradient Reconstruction

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



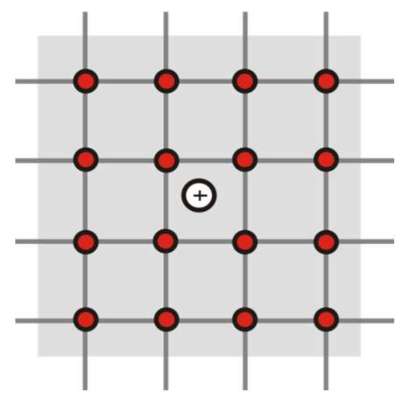
Central differences

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



Discrete convolution filters on grid

- Image processing filters; e.g. Sobel (3^3 neighbors in 3D)



Continuous convolution filters

- Derived continuous reconstruction filters
- E.g., the cubic B-spline and its derivatives (4^3 neighbors)

Finite Differences

Obtain first derivative from Taylor expansion

$$
\begin{aligned}
f(x_0 + h) &= f(x_0) + \frac{f'(x_0)}{1!} h + \frac{f''(x_0)}{2!} h^2 + \ldots \\
&= \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} h^n .
\end{aligned}
$$

Forward differences / backward differences

$$
\begin{aligned}
f(x_0)' &= \frac{f(x_0 + h) - f(x_0)}{h} + o(h) \\
f(x_0)' &= \frac{f(x_0) - f(x_0 - h)}{h} + o(h)
\end{aligned}
$$

Finite Differences

Central differences

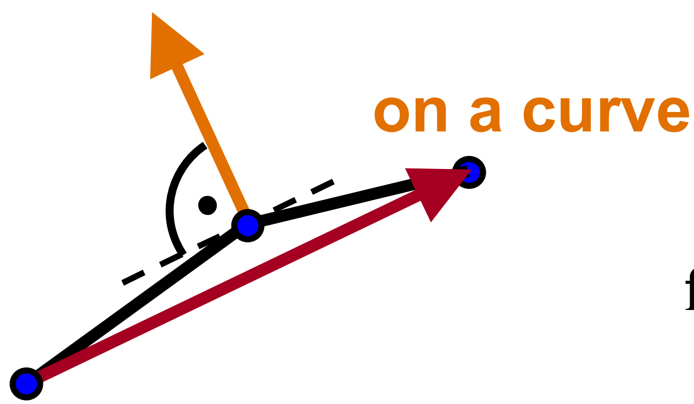
$$
\begin{aligned}
f(x_0+h) &= f(x_0) + \frac{f'(x_0)}{1!}h + \frac{f''(x_0)}{2!}h^2 + o(h^3) \\
f(x_0-h) &= f(x_0) - \frac{f'(x_0)}{1!}h + \frac{f''(x_0)}{2!}h^2 + o(h^3)
\end{aligned}
$$

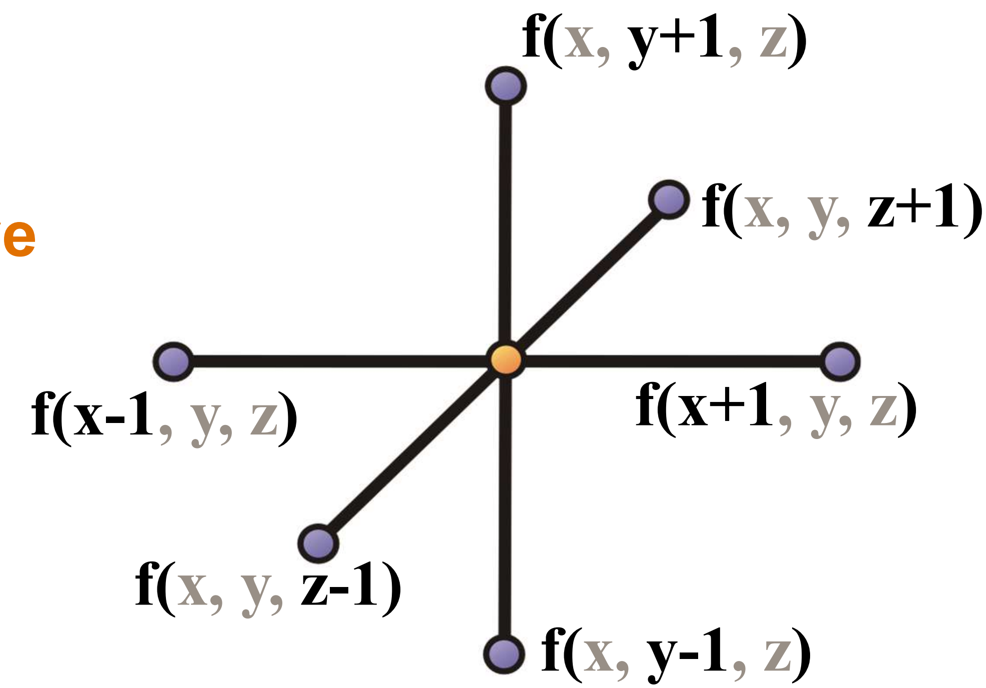
$$
f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h} + o(h^2)
$$

Central Differences

Need only two neighboring voxels per derivative

Most common method





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

Multi-Linear Interpolation

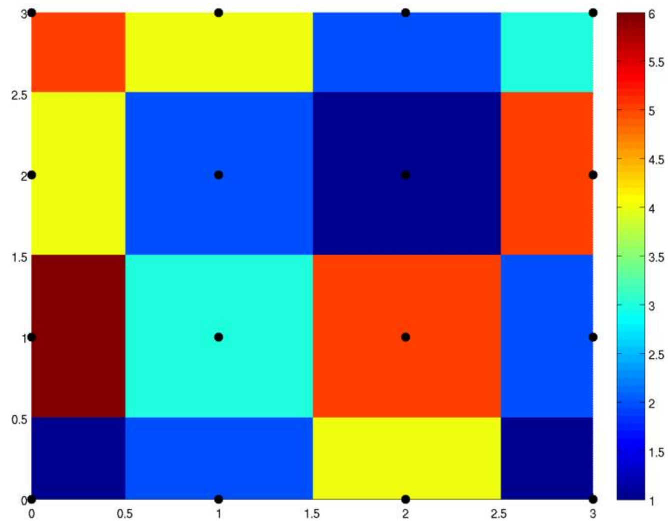
Bi-linear Filtering Example (Magnification)

Original image

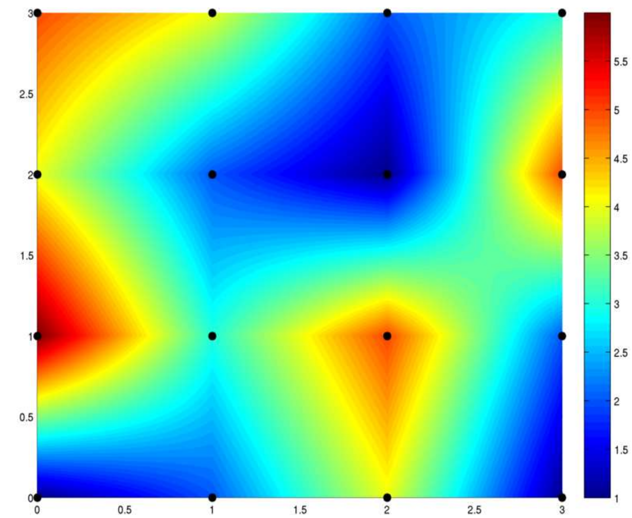
Nearest neighbor

Bi-linear filtering

Bi-Linear Interpolation vs. Nearest Neighbor



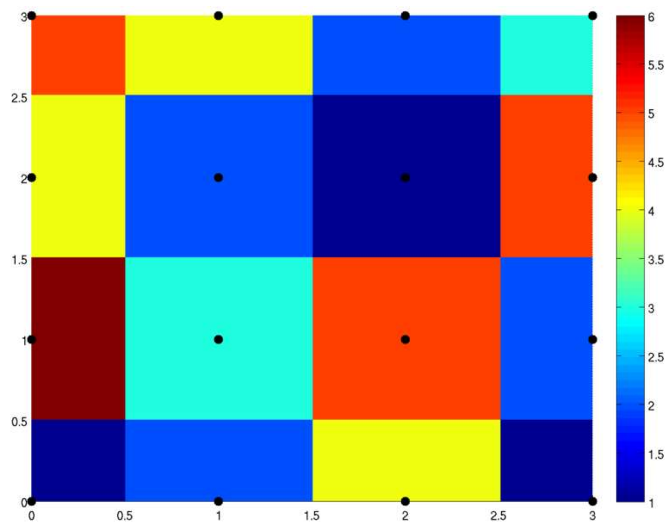
nearest-neighbor



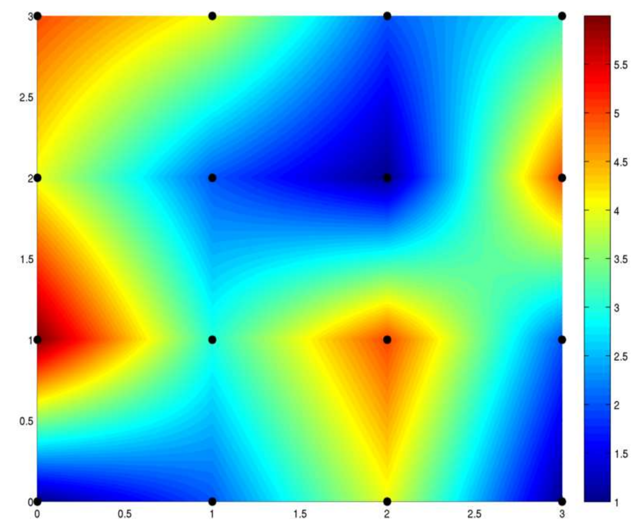
bi-linear

wikipedia

Bi-Linear Interpolation vs. Nearest Neighbor



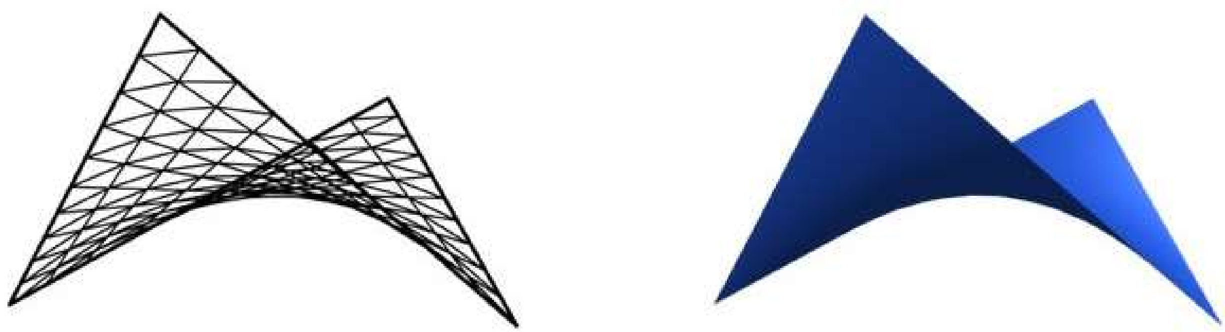
nearest-neighbor



bi-linear

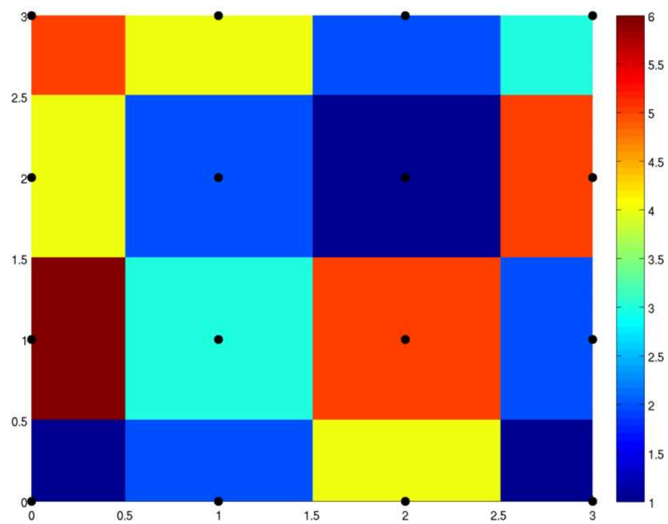
wikipedia

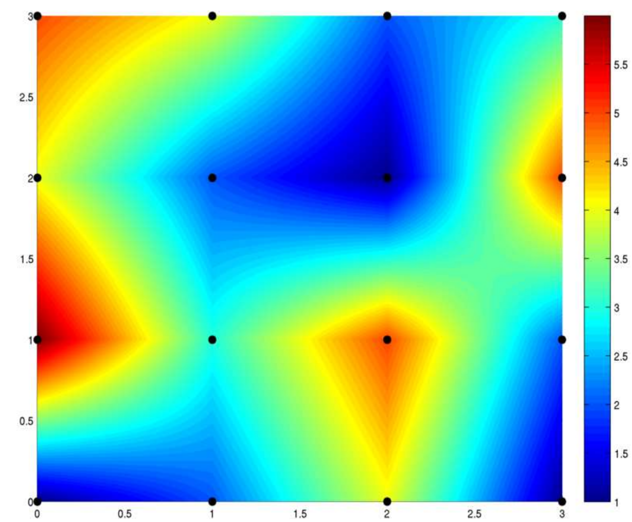
for surfaces,
height interpolation:



Bilinear patch (courtesy J. Han)

Bi-Linear Interpolation vs. Nearest Neighbor



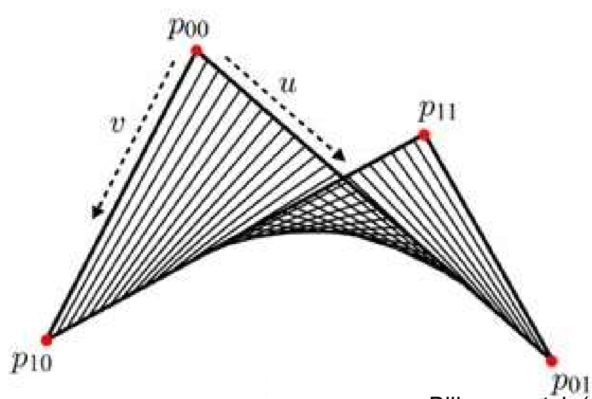


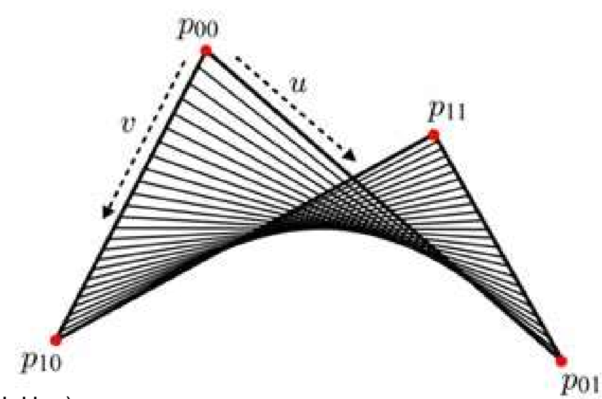
wikipedia

nearest-neighbor

bi-linear

for surfaces,
height interpolation:



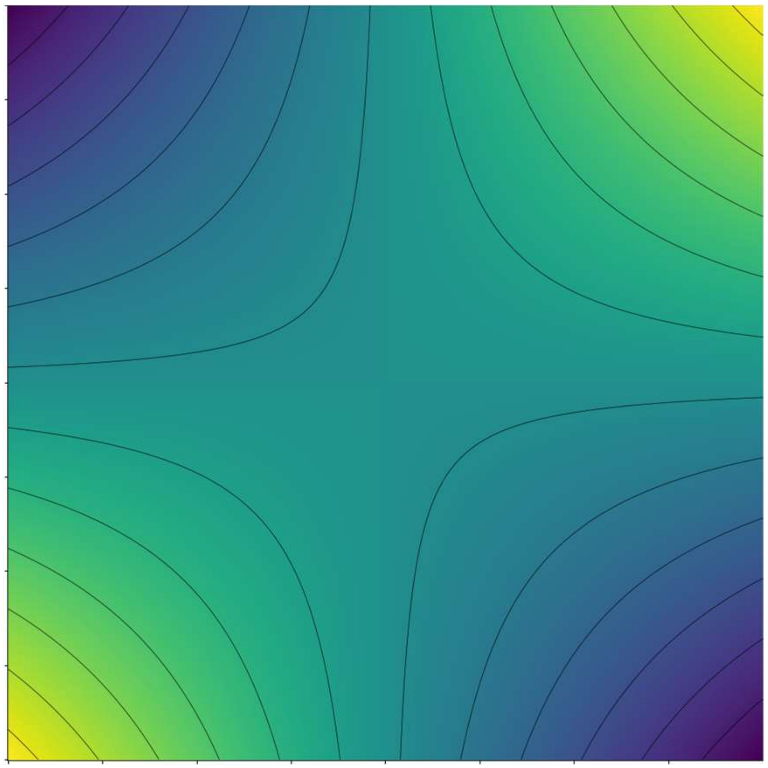


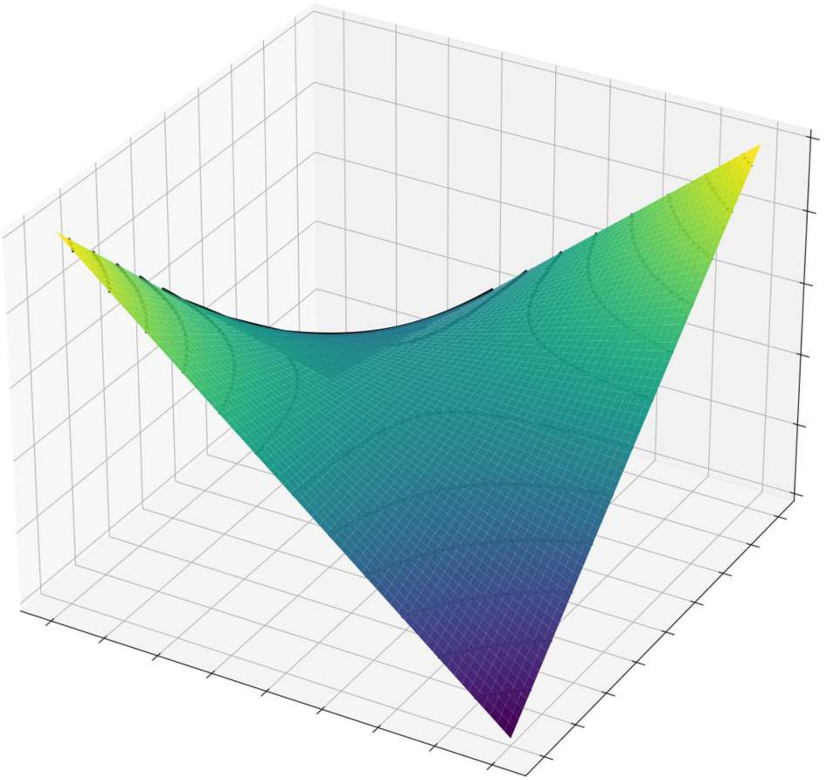
Bilinear patch (courtesy J. Han)

Bi-Linear Interpolation

Consider area between 2x2 adjacent samples (e.g., pixel centers)

Example #1: 1 at bottom-left and top-right, 0 at top-left and bottom-right

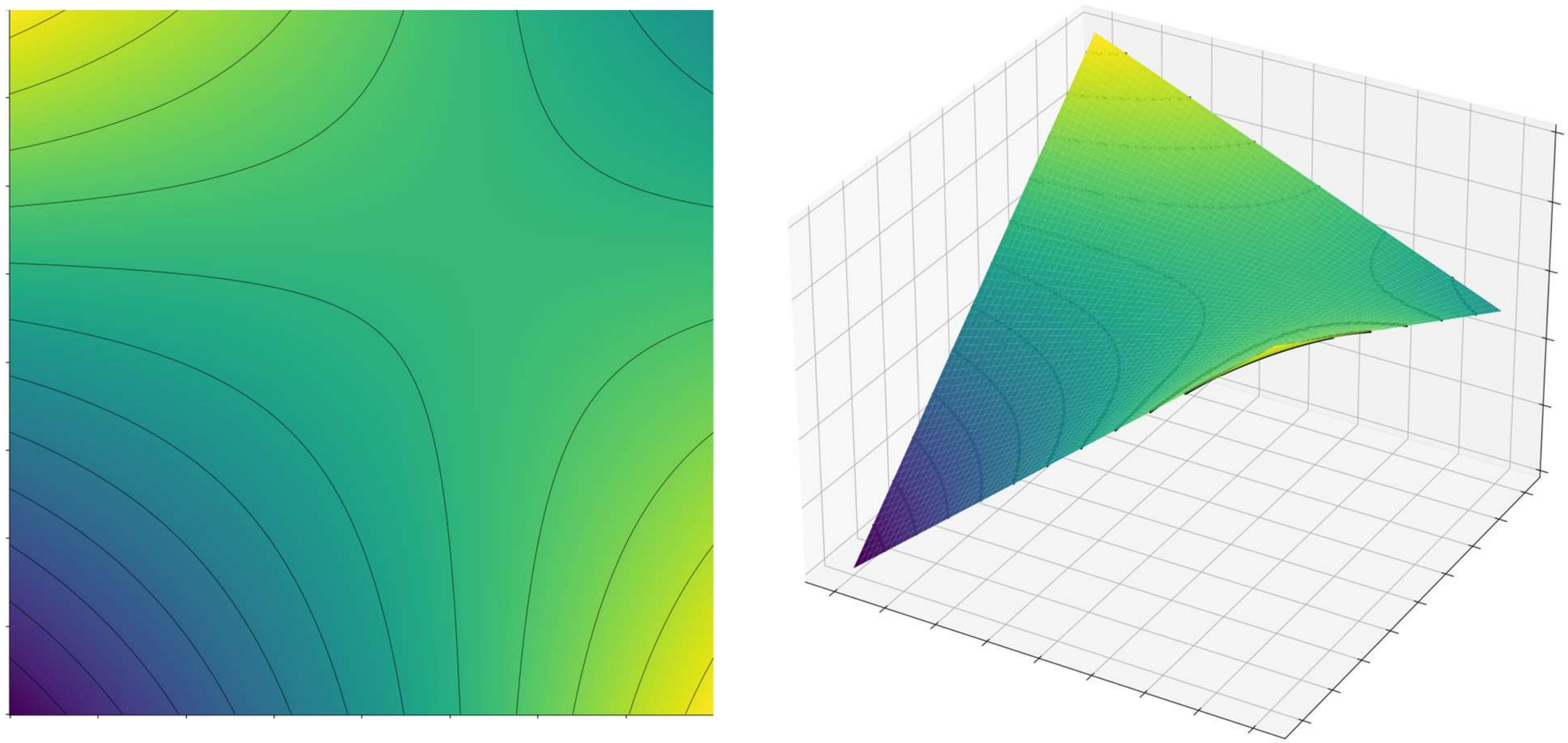




Bi-Linear Interpolation

Consider area between 2x2 adjacent samples (e.g., pixel centers)

Example #2: 1 at top-left and bottom-right, 0 at bottom-left, 0.5 at top-right



Bi-Linear Interpolation

Consider area between 2x2 adjacent samples (e.g., pixel centers):

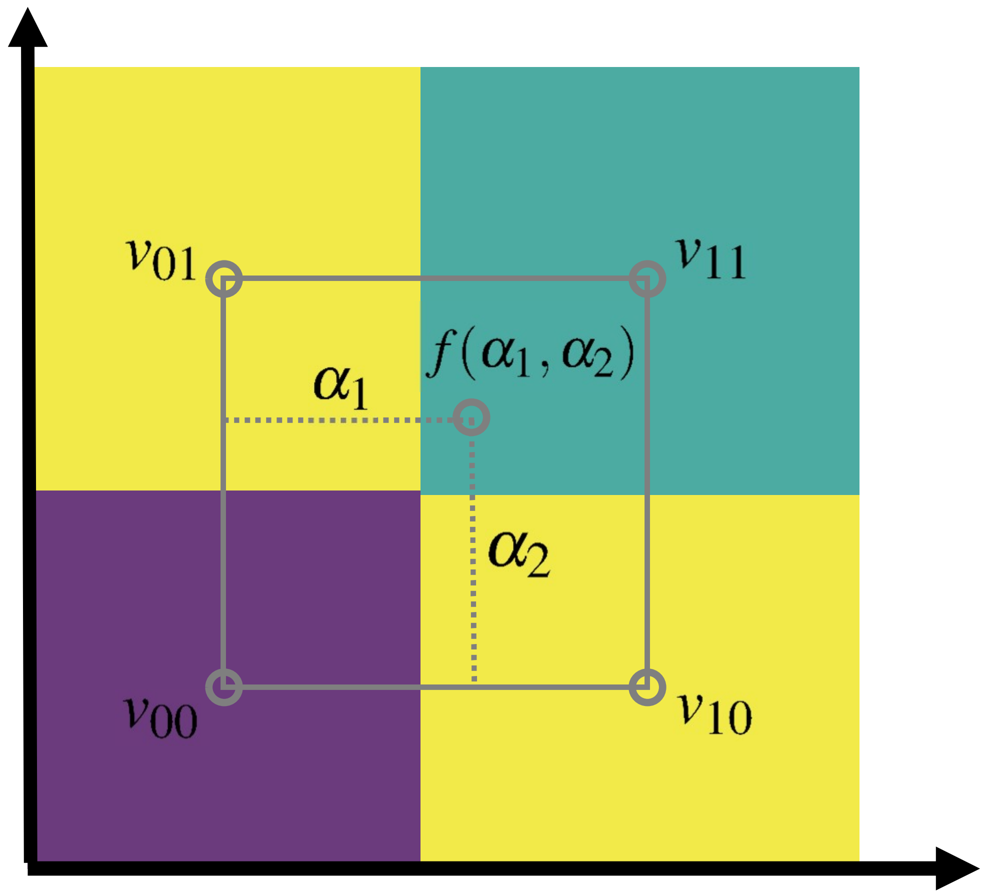
Given any (fractional) position

$$\alpha_1 := x_1 - \lfloor x_1 \rfloor \qquad \alpha_1 \in [0.0, 1.0)$$
$$\alpha_2 := x_2 - \lfloor x_2 \rfloor \qquad \alpha_2 \in [0.0, 1.0)$$

and 2x2 sample values

$$\begin{bmatrix} v_{01} & v_{11} \\ v_{00} & v_{10} \end{bmatrix}$$

Compute: $f(\alpha_1, \alpha_2)$



Bi-Linear Interpolation

Consider area between 2x2 adjacent samples (e.g., pixel centers):

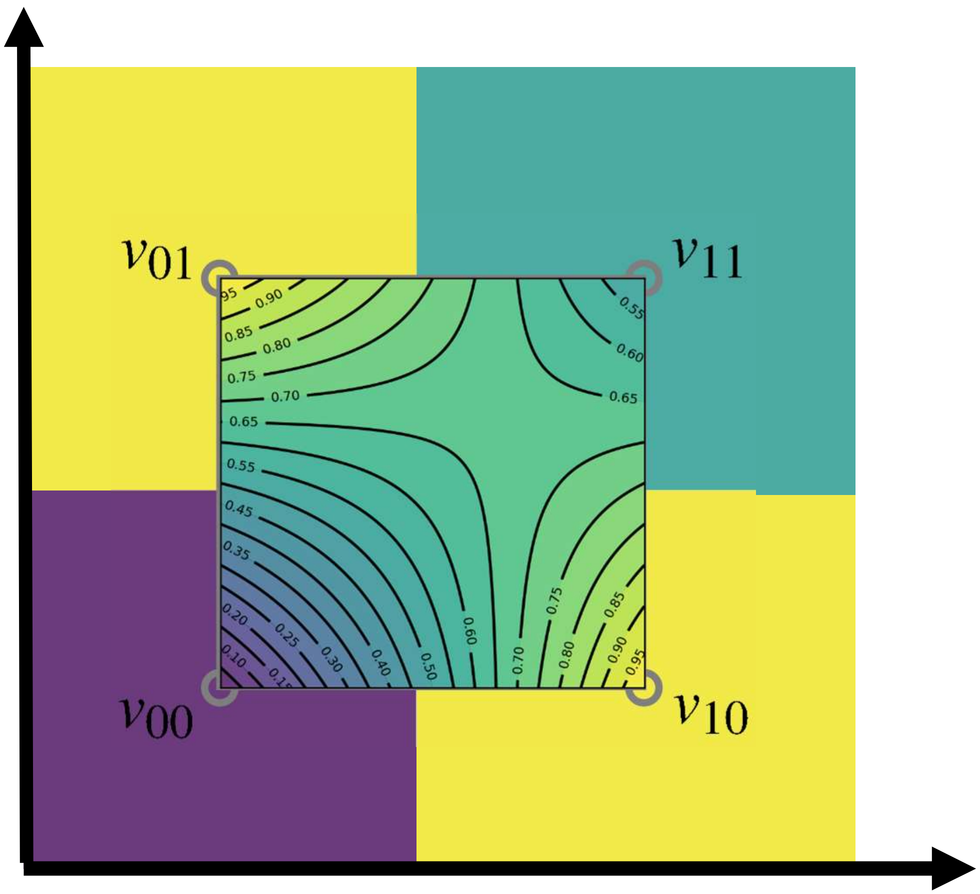
Given any (fractional) position

$$\alpha_1 := x_1 - \lfloor x_1 \rfloor \qquad \alpha_1 \in [0.0, 1.0)$$
$$\alpha_2 := x_2 - \lfloor x_2 \rfloor \qquad \alpha_2 \in [0.0, 1.0)$$

and 2x2 sample values

$$\begin{bmatrix} v_{01} & v_{11} \\ v_{00} & v_{10} \end{bmatrix}$$

Compute: $f(\alpha_1, \alpha_2)$



Bi-Linear Interpolation

Weights in 2x2 format:

$$\begin{bmatrix} \alpha_2 \\ (1-\alpha_2) \end{bmatrix} \begin{bmatrix} (1-\alpha_1) & \alpha_1 \end{bmatrix} = \begin{bmatrix} (1-\alpha_1)\alpha_2 & \alpha_1\alpha_2 \\ (1-\alpha_1)(1-\alpha_2) & \alpha_1(1-\alpha_2) \end{bmatrix}$$

Interpolate function at (fractional) position (α_1, α_2) :

$$f(\alpha_1, \alpha_2) = \begin{bmatrix} \alpha_2 & (1-\alpha_2) \end{bmatrix} \begin{bmatrix} v_{01} & v_{11} \\ v_{00} & v_{10} \end{bmatrix} \begin{bmatrix} (1-\alpha_1) \\ \alpha_1 \end{bmatrix}$$

Bi-Linear Interpolation

Interpolate function at (fractional) position (α_1, α_2) :

$$f(\alpha_1, \alpha_2) = \begin{bmatrix} \alpha_2 & (1-\alpha_2) \end{bmatrix} \begin{bmatrix} v_{01} & v_{11} \\ v_{00} & v_{10} \end{bmatrix} \begin{bmatrix} (1-\alpha_1) \\ \alpha_1 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_2 & (1-\alpha_2) \end{bmatrix} \begin{bmatrix} (1-\alpha_1)v_{01} + \alpha_1 v_{11} \\ (1-\alpha_1)v_{00} + \alpha_1 v_{10} \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_2 v_{01} + (1-\alpha_2)v_{00} & \alpha_2 v_{11} + (1-\alpha_2)v_{10} \end{bmatrix} \begin{bmatrix} (1-\alpha_1) \\ \alpha_1 \end{bmatrix}$$

Bi-Linear Interpolation

Interpolate function at (fractional) position (α_1, α_2) :

$$f(\alpha_1, \alpha_2) = \begin{bmatrix} \alpha_2 & (1-\alpha_2) \end{bmatrix} \begin{bmatrix} v_{01} & v_{11} \\ v_{00} & v_{10} \end{bmatrix} \begin{bmatrix} (1-\alpha_1) \\ \alpha_1 \end{bmatrix}$$

$$= (1-\alpha_1)(1-\alpha_2)v_{00} + \alpha_1(1-\alpha_2)v_{10} + (1-\alpha_1)\alpha_2 v_{01} + \alpha_1\alpha_2 v_{11}$$

$$= v_{00} + \alpha_1(v_{10} - v_{00}) + \alpha_2(v_{01} - v_{00}) + \alpha_1\alpha_2(v_{00} + v_{11} - v_{10} - v_{01})$$

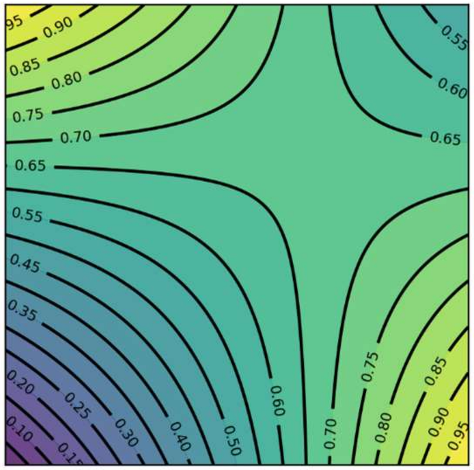
Bi-Linear Interpolation: Contours

Find one specific iso-contour (can of course do this for any/all isovalues):

$$f(\alpha_1, \alpha_2) = c$$

Find all (α_1, α_2) where:

$$v_{00} + \alpha_1 (v_{10} - v_{00}) + \alpha_2 (v_{01} - v_{00}) + \alpha_1 \alpha_2 (v_{00} + v_{11} - v_{10} - v_{01}) = c$$



Thank you.

Thanks for material

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