Introduction to Computer Graphics

– Image Processing (1) –

June 9, 2016

Kenshi Takayama

Today’s topics

- Edge-aware image processing

- Gradient-domain image processing

Image smoothing using Gaussian Filter

- Smoothness parameter σ


Original


$\sigma = 2$


$\sigma = 5$


$\sigma = 10$

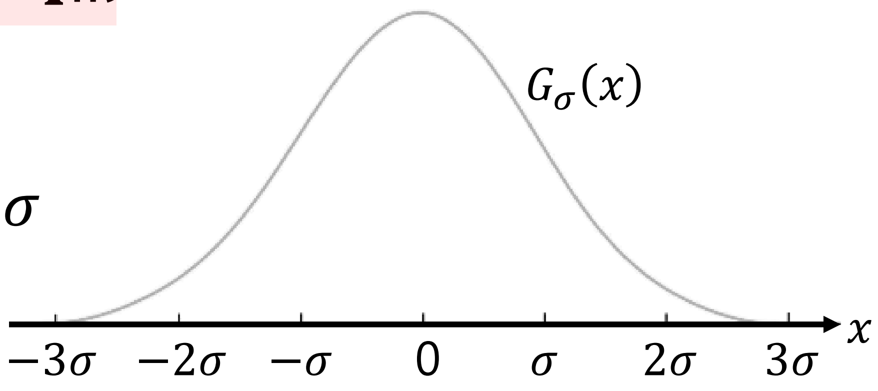
Equation of Gaussian Filter

- $I_{\mathbf{p}}$ represents pixel value of image I at position $\mathbf{p} = (p_{\mathrm{x}}, p_{\mathrm{y}}) \in \Omega$
  - For given resolution e.g. 640×480, $\Omega \coloneqq \{1, \cdots, 640\} \times \{1, \cdots, 480\}$

- $\mathrm{GF}_\sigma[I]$ represents filtered image with Gaussian parameter σ :

$$\mathrm{GF}_\sigma[I]_{\mathbf{p}} \coloneqq \frac{\sum_{\mathbf{q}\in\Omega} G_\sigma(\|\mathbf{p}-\mathbf{q}\|)\, I_{\mathbf{q}}}{\underbrace{\sum_{\mathbf{q}\in\Omega} G_\sigma(\|\mathbf{p}-\mathbf{q}\|)}_{W_{\mathbf{p}}}}$$

  - $G_\sigma(x) \coloneqq \exp\left(-\frac{x^2}{2\sigma^2}\right)$ ← Gaussian Kernel of radius σ



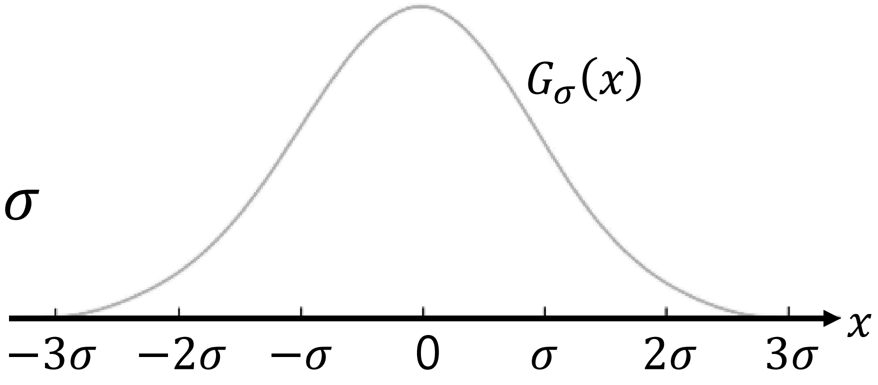
Equation of Gaussian Filter

- $I_{\mathbf{p}}$ represents pixel value of image I at position $\mathbf{p} = \left(p_{\mathrm{x}}, p_{\mathrm{y}}\right) \in \Omega$
  - For given resolution e.g. 640×480, $\Omega \coloneqq \{1, \cdots, 640\} \times \{1, \cdots, 480\}$

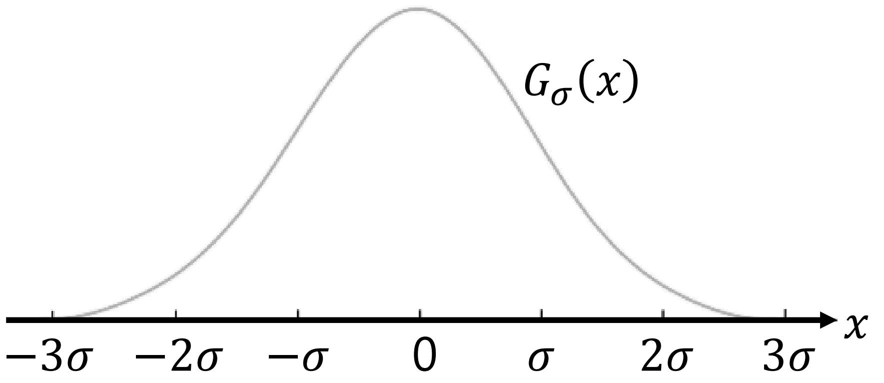
- $\mathrm{GF}_{\sigma}[I]$ represents filtered image with Gaussian parameter σ :

$$\mathrm{GF}_{\sigma}[I]_{\mathbf{p}} \coloneqq \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in \Omega} G_{\sigma}(\|\mathbf{p} - \mathbf{q}\|)\, I_{\mathbf{q}}$$

  - $G_{\sigma}(x) \coloneqq \exp\left(-\frac{x^2}{2\sigma^2}\right)$ 🡐 Gaussian Kernel of radius σ

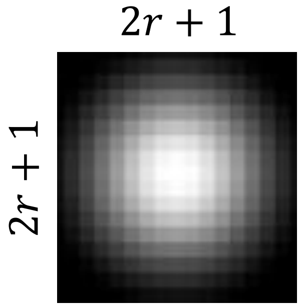


Implementing Gaussian Filter

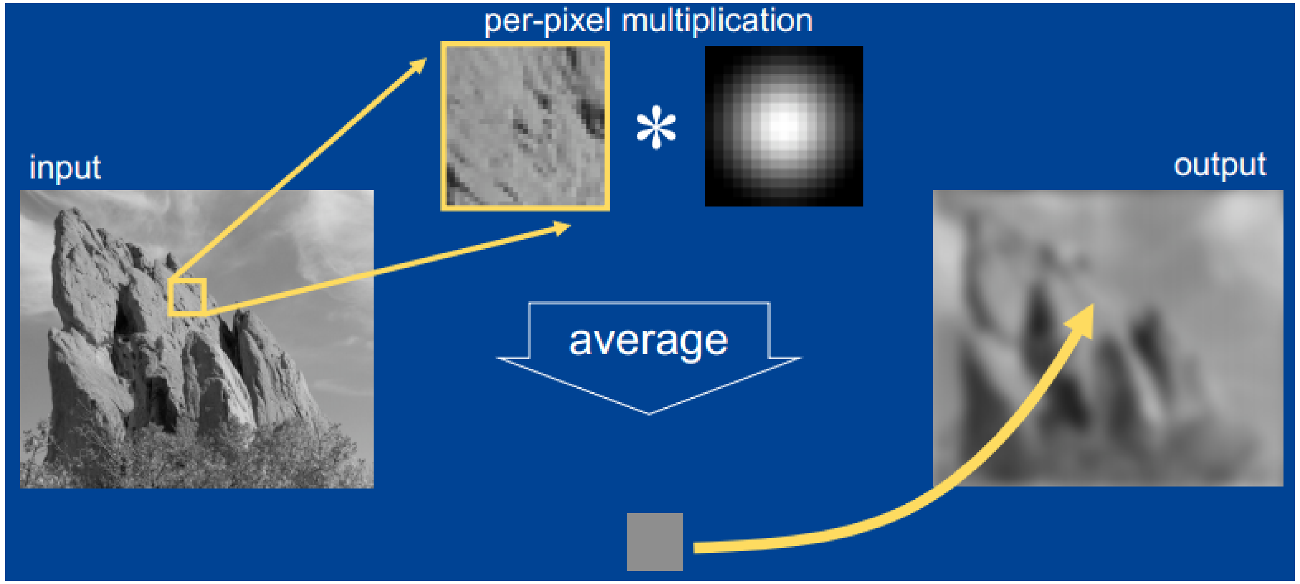


- $G_\sigma(3\sigma) \approx 0$ ➔ Distant pixels can be ignored

- For fixed size $r := \text{ceil}(3\sigma)$, precompute weights on a $(2r+1) \times (2r+1)$ stencil



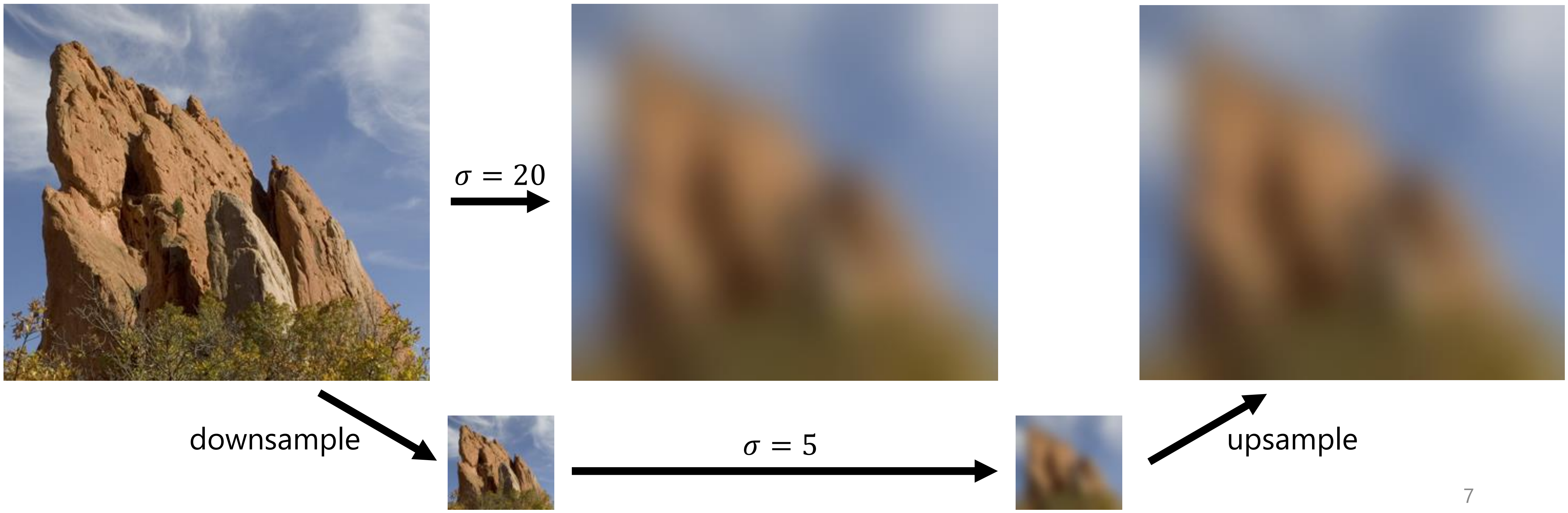
Stencil



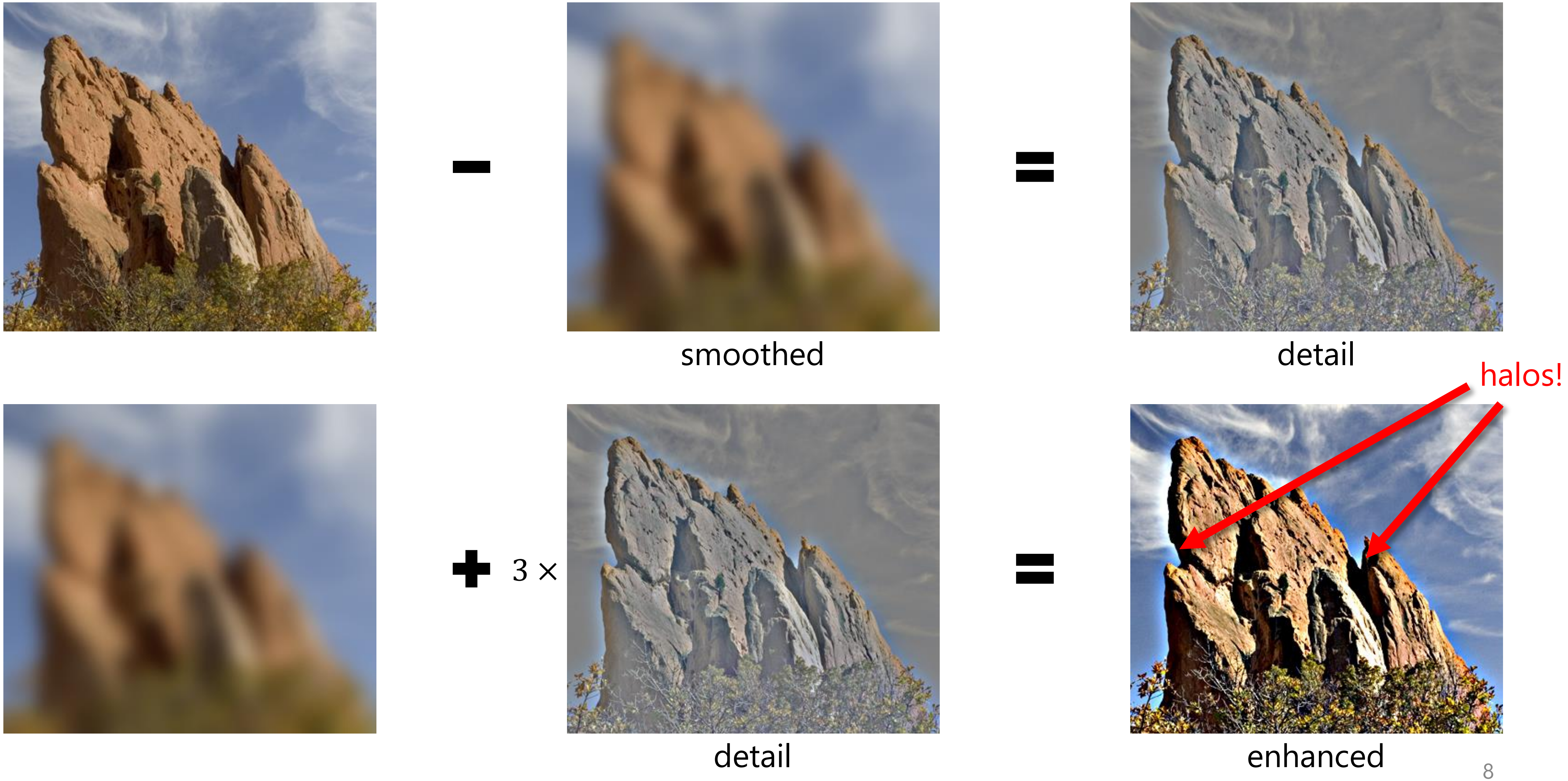
http://people.csail.mit.edu/sparis/bf_course/

When kernel radius σ is very large

- Direct computation takes a lot of time
- Alternative: downsample ➔ smooth with small σ ➔ upsample



Detail Extraction & Enhancement

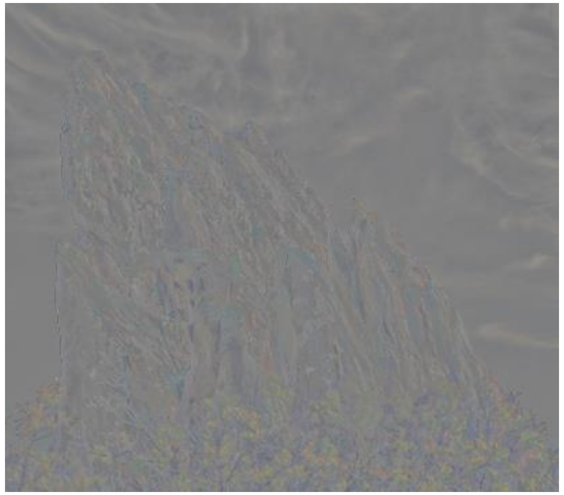


When using edge-aware smoothing, …

−

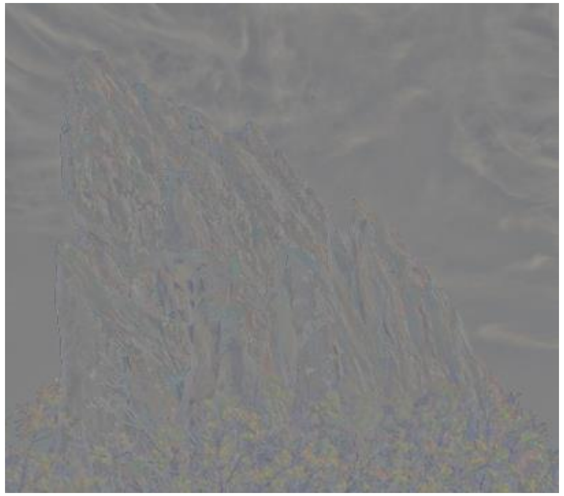
smoothed

=



detail

+ $3 \times$



detail

=

enhanced

Edge-aware smoothing using Bilateral Filter

- Two parameters
  - σ_s : Range of smoothing w.r.t. pixel's location
  - σ_r : Range of smoothing w.r.t. pixel's color

$$\mathrm{BF}_{\sigma_s,\sigma_r}[I]_{\mathbf{p}} \coloneqq \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in\Omega} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(\|I_{\mathbf{p}}-I_{\mathbf{q}}\|) I_{\mathbf{q}}$$

In all cases, $\sigma_s = 10$

Original

$\sigma_r = 32$

$\sigma_r = 128$

$\sigma_r = 512$

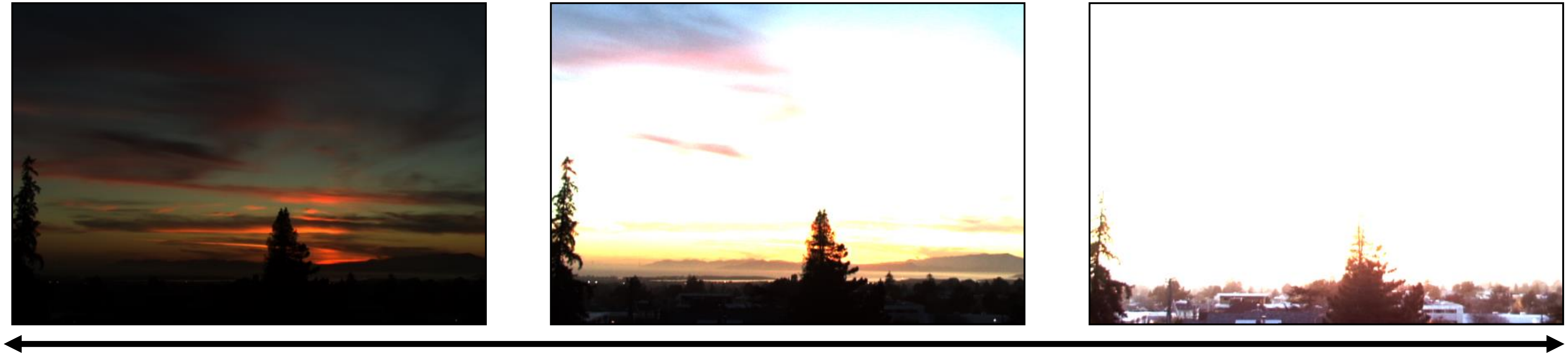
Application of Bilateral Filter: Stylization

Real-time video abstraction [Winnemöller SIGGRAPH06]

Application of Bilateral Filter: Tone Mapping

- Range of each channel (24bit color image): 1~255

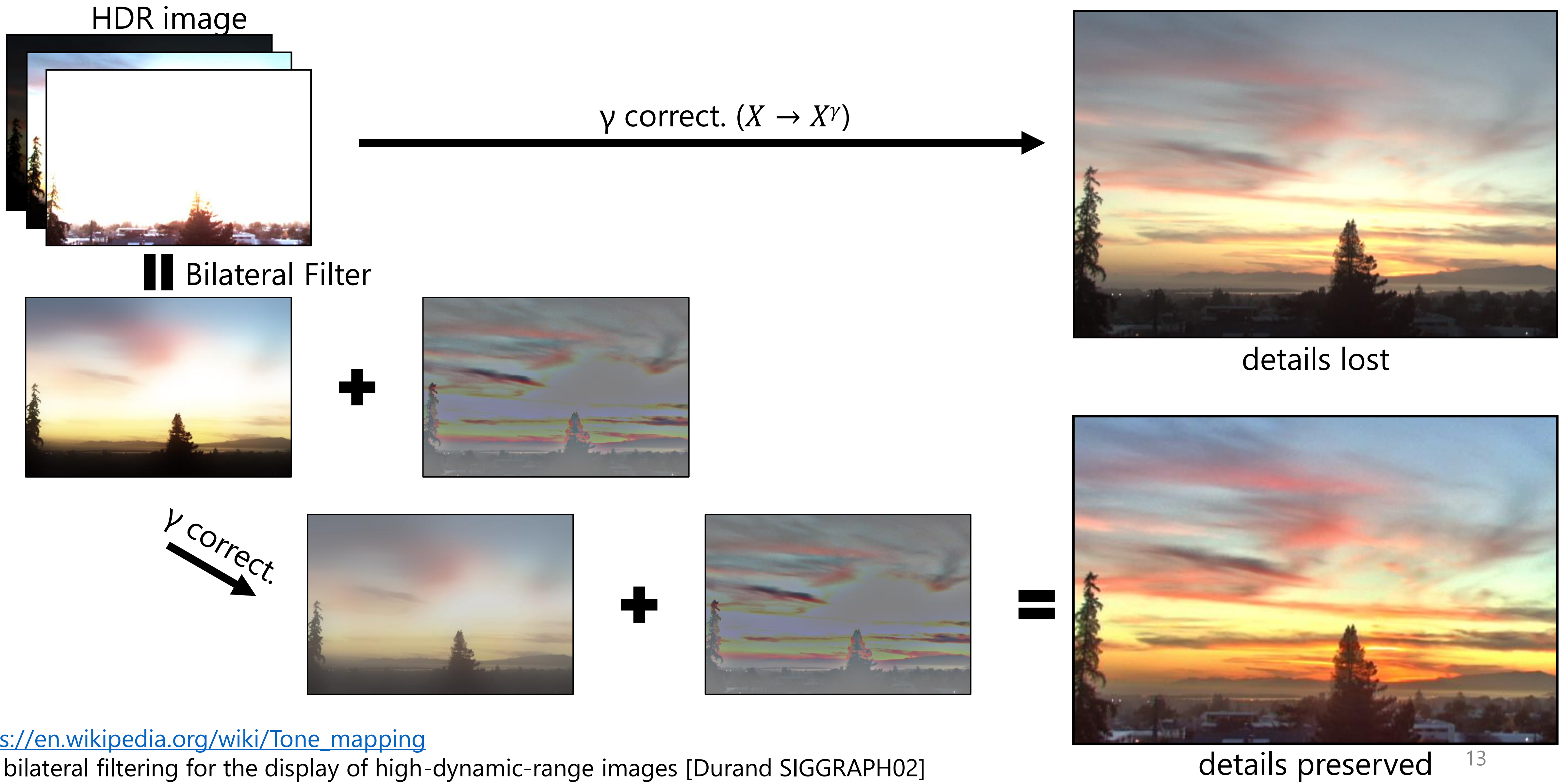
- Range of light intensity in the real world: $1 \sim 10^5$
  - **H**igh **D**ynamic **R**ange image
  - Can be obtained by photographing with different exposure times



Short exposure Long exposure

https://en.wikipedia.org/wiki/Tone_mapping
Fast bilateral filtering for the display of high-dynamic-range images [Durand SIGGRAPH02]

Application of Bilateral Filter: Tone Mapping

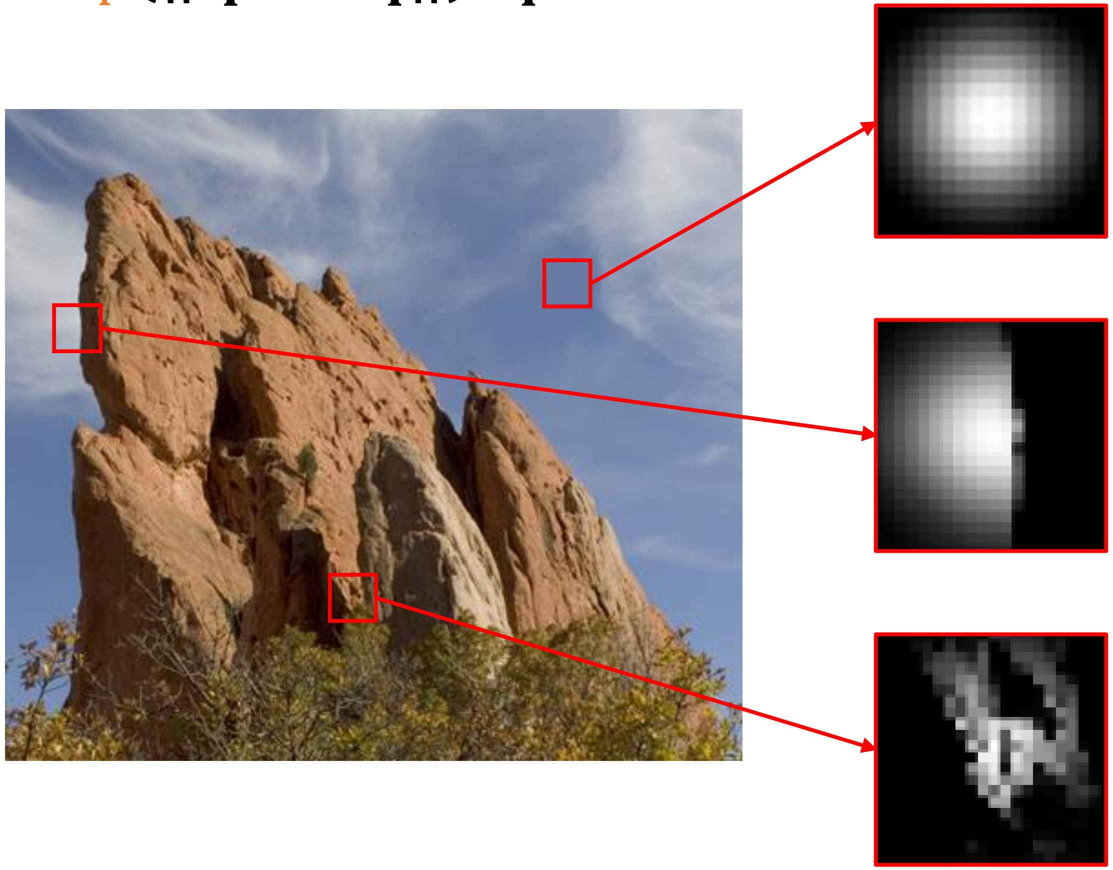


https://en.wikipedia.org/wiki/Tone_mapping
Fast bilateral filtering for the display of high-dynamic-range images [Durand SIGGRAPH02]

Naïve implementation of Bilateral Filter

$$\frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in \Omega} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(\|I_{\mathbf{p}} - I_{\mathbf{q}}\|) I_{\mathbf{q}}$$

- Recompute stencil for every pixel location $\mathbf{p} \in \Omega$
  ➔ slow



- (Basic Assignment)

Another view toward Bilateral Filter

- Define feature vector $\mathbf{f}_\mathbf{p} \coloneqq \left(\frac{\mathbf{p}}{\sigma_\mathrm{s}}, \frac{I_\mathbf{p}}{\sigma_\mathrm{r}}\right)$ for pixel location $\mathbf{p}$ and intensity $I_\mathbf{p}$

- Weight of Bilateral Filter is equivalent to Gaussian kernel applied to Euclidean distance in the feature space

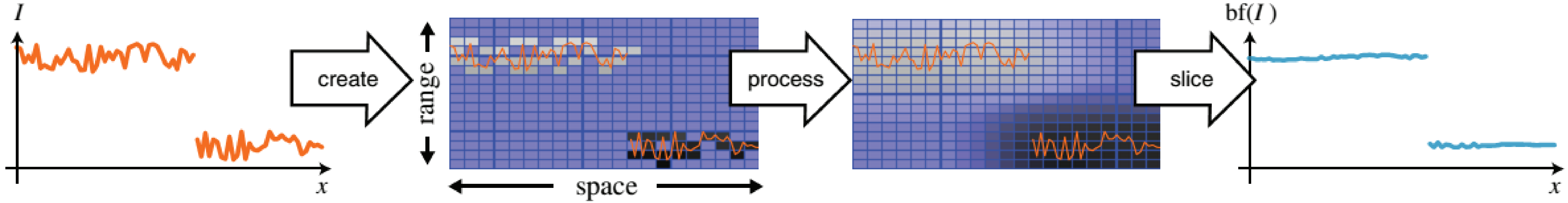
$$
\begin{aligned}
&G_{\sigma_\mathrm{s}}(\|\mathbf{p}-\mathbf{q}\|)G_{\sigma_\mathrm{r}}\left(\left\|I_\mathbf{p}-I_\mathbf{q}\right\|\right) \\
&= \exp\left(-\frac{\|\mathbf{p}-\mathbf{q}\|^2}{2\sigma_\mathrm{s}^2}\right)\exp\left(-\frac{\left\|I_\mathbf{p}-I_\mathbf{q}\right\|^2}{2\sigma_\mathrm{r}^2}\right) \\
&= \exp\left(-\frac{\left\|\mathbf{f}_\mathbf{p}-\mathbf{f}_\mathbf{q}\right\|^2}{2}\right) \\
&= G_1\left(\left\|\mathbf{f}_\mathbf{p}-\mathbf{f}_\mathbf{q}\right\|\right)
\end{aligned}
$$

- Bilateral Filter is equivalent to applying Gaussian Filter of radius 1 to sample points $\{\mathbf{f}_\mathbf{p}\}$ in the feature space
  ➔ Simpler computation

Bilateral Grid [Paris06; Chen07]

- Define 3D feature space as (X-coord, Y-coord, intensity), map sample points $\{\mathbf{f_p}\}$ to 3D grid

- The larger σ_s & σ_r, the coarser the grid ➔ lower comput. cost



(a) input 1D image (b) grid created from 1D image (c) filtered grid and slicing image (d) filtered 1D image

A Fast Approximation of the Bilateral Filter using a Signal Processing Approach [Paris ECCV06]
Real-time edge-aware image processing with the bilateral grid [Chen SIGGRAPH07]

Weight map generation using feature space

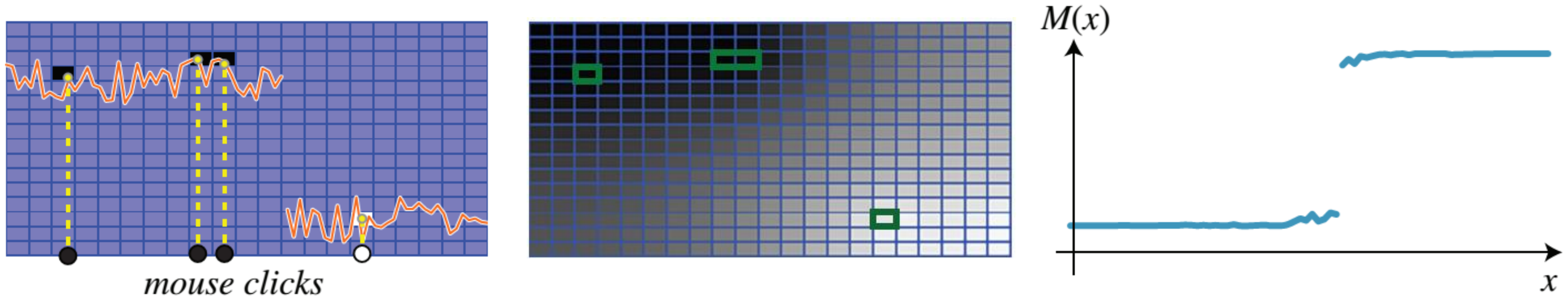
White scribble ➔ constraint of weight=1
Black scribble ➔ constraint of weight=0

Weight map

Application: color adjustment

- Various names: Edit Propagation, Matting, Segmentation

- Solve Laplace equation on Bilateral Grid

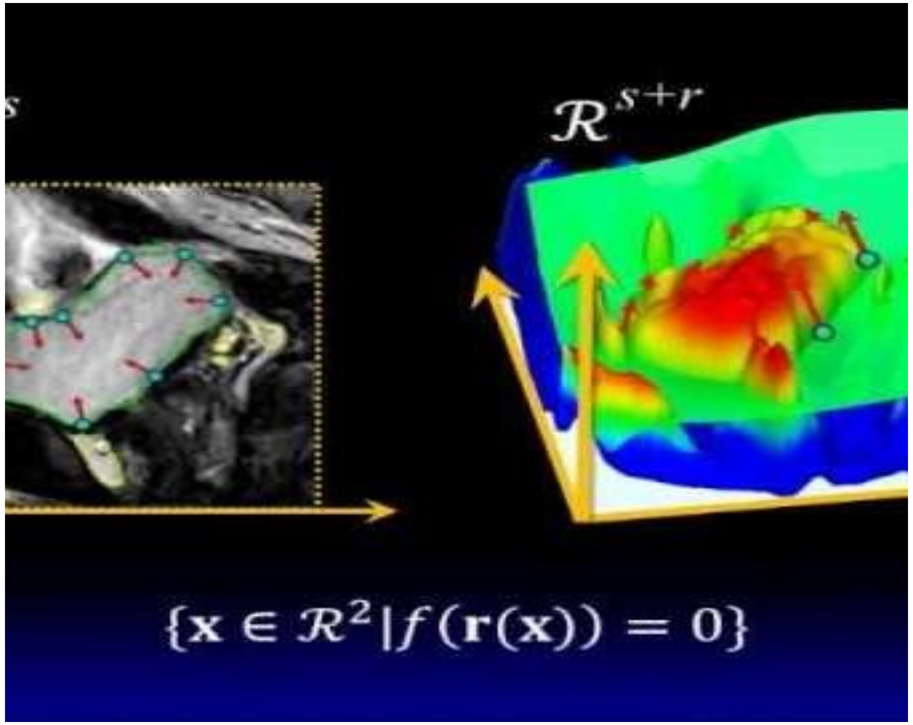


Real-time edge-aware image processing with the bilateral grid [Chen SIGGRAPH07]

Weight map generation using feature space

Interpolation using RBF [Li10]
(Purpose: edit propagation for images/videos)

Interpolation using Hermite RBF [Ijiri13]
(Purpose: segmentation of CT volume)



https://www.youtube.com/watch?v=mL6ig_OaQAA

Instant Propagation of Sparse Edits on Images and Videos [Li PG10]
Bilateral Hermite Radial Basis Functions for Contour-based Volume Segmentation [Ijiri EG13]

Extension of Bilateral Filter: Joint (Cross) Bilateral Filter

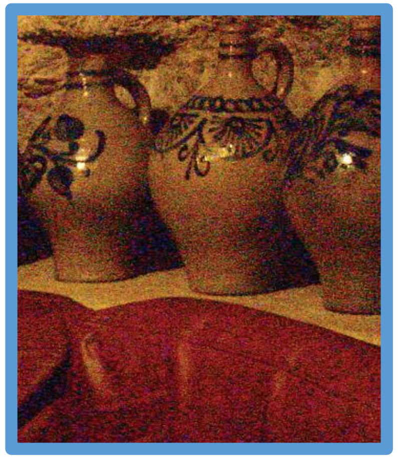
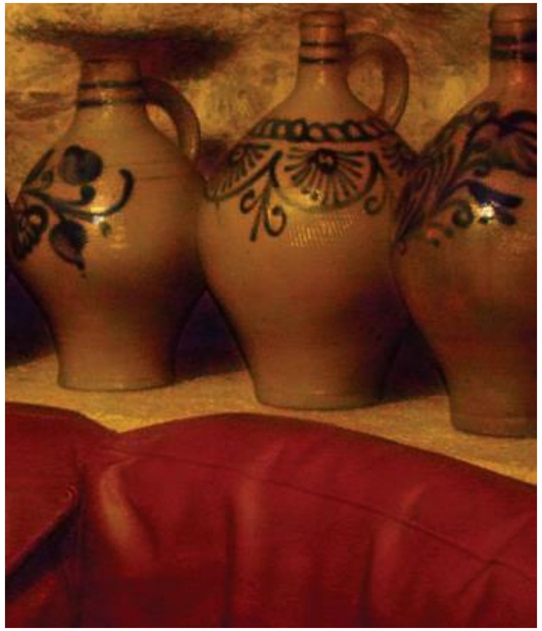


Photo A: without flash
☺ Correct color
☹ Noisy, blurred

Photo F: with flash
☹ Incorrect color
☺ Less noisy, sharp

After applying JBF



$$\mathrm{JBF}_{\sigma_s,\,\sigma_r}(A, F)_{\mathbf{p}} := \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in\Omega} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(\|F_{\mathbf{p}} - F_{\mathbf{q}}\|)\, A_{\mathbf{q}}$$

Digital Photography with Flash and No-Flash Image Pairs [Petschnigg SIGGRAPH04]
Flash Photography Enhancement via Intrinsic Relighting [Eisemann SIGGRAPH04]

Extension of Bilateral Filter: Non-Local Means Filter

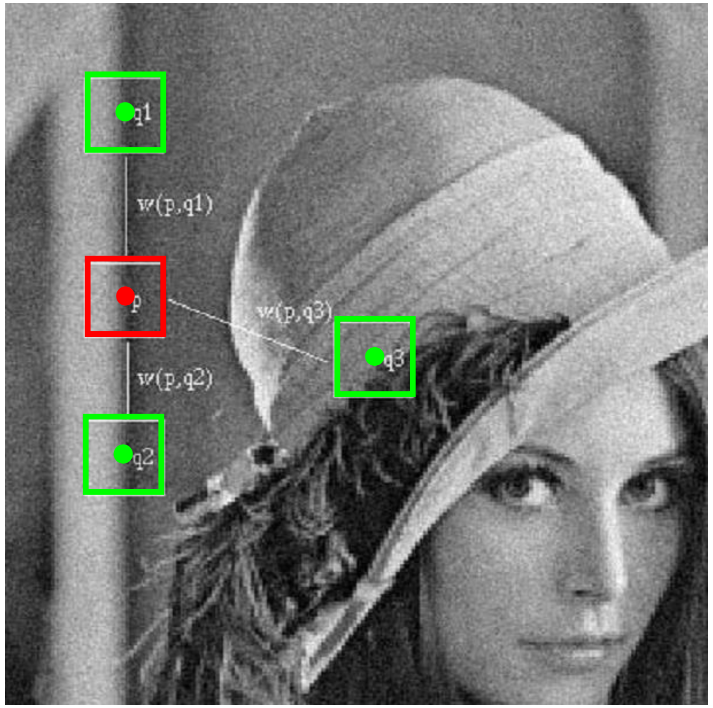
- Define feature space by neighborhood vector $\mathbf{n_p}$ representing 7×7 sub-image centered at $\mathbf{p}$

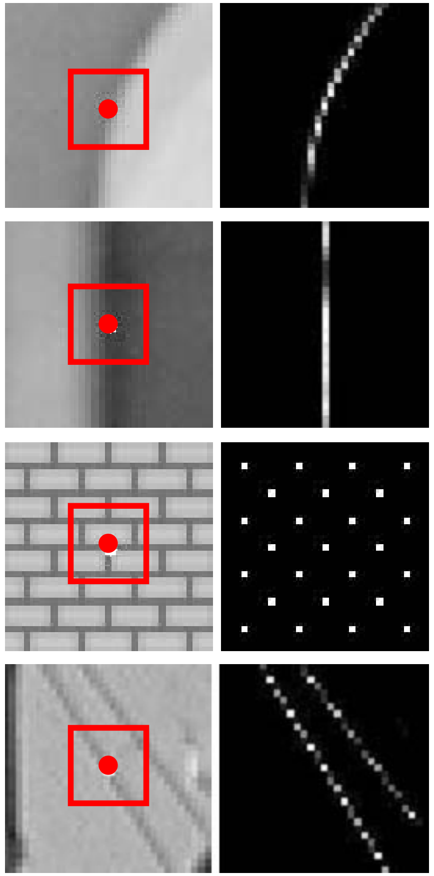
$$\mathrm{NLMF}_{\sigma}(I)_{\mathbf{p}} \coloneqq \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in \Omega} G_{\sigma}(\|\mathbf{n_p} - \mathbf{n_q}\|)\, I_{\mathbf{q}}$$

Noisy input

Bilateral

NL Means





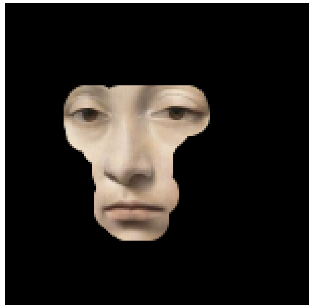
A non local algorithm for image denoising [Buades CVPR05]

Today’s topics

- Edge-aware image processing

- Gradient-domain image processing

Scenario: insert source img. into destination img.



Source

Dest.

Simple copying
☹

Boundary blurred
☹

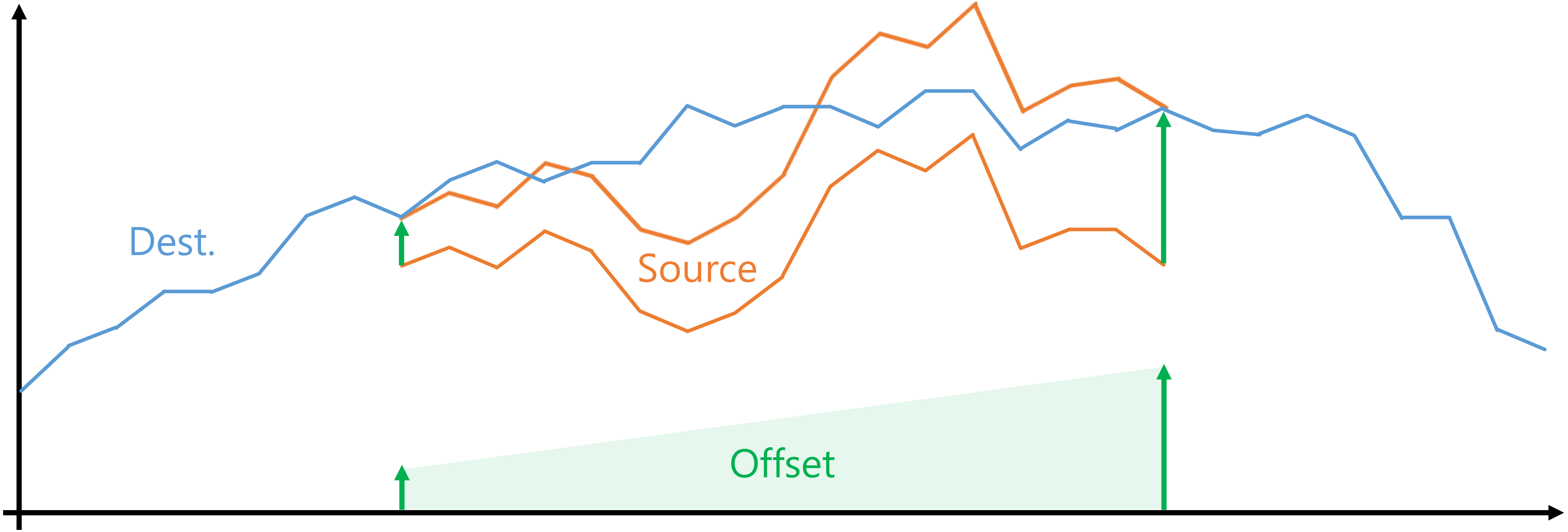
Gradient-domain processing
☺

Poisson image editing [Perez SIGGRAPH03]
Gradient Domain Manipulation Techniques in Vision and Graphics [Agrawal ICCV07 Course]

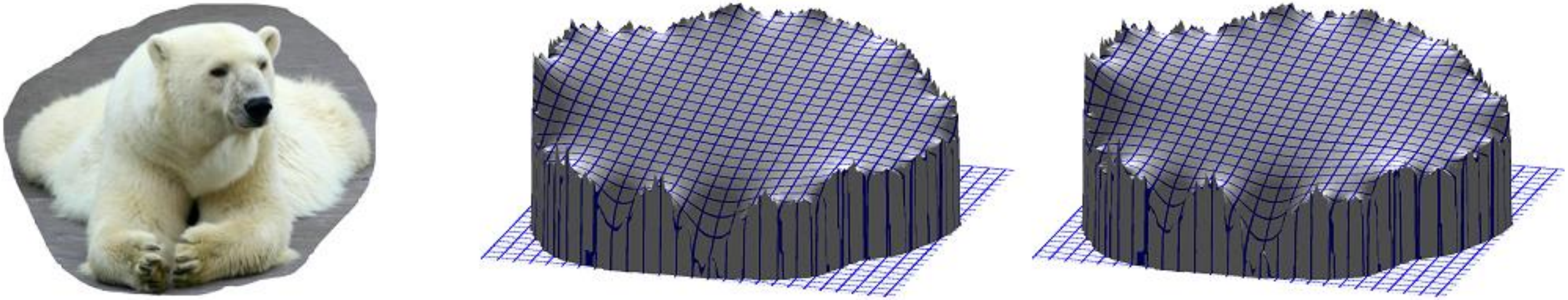
Scenario: generating panorama from several shots

Efficient gradient-domain compositing using quadtrees [Agarwala SIGGRAPH07]

Simple case of 1D grayscale image



2D case: offset by Laplace Membrane



(a) Source patch (b) Laplace membrane (c) Mean-value membrane

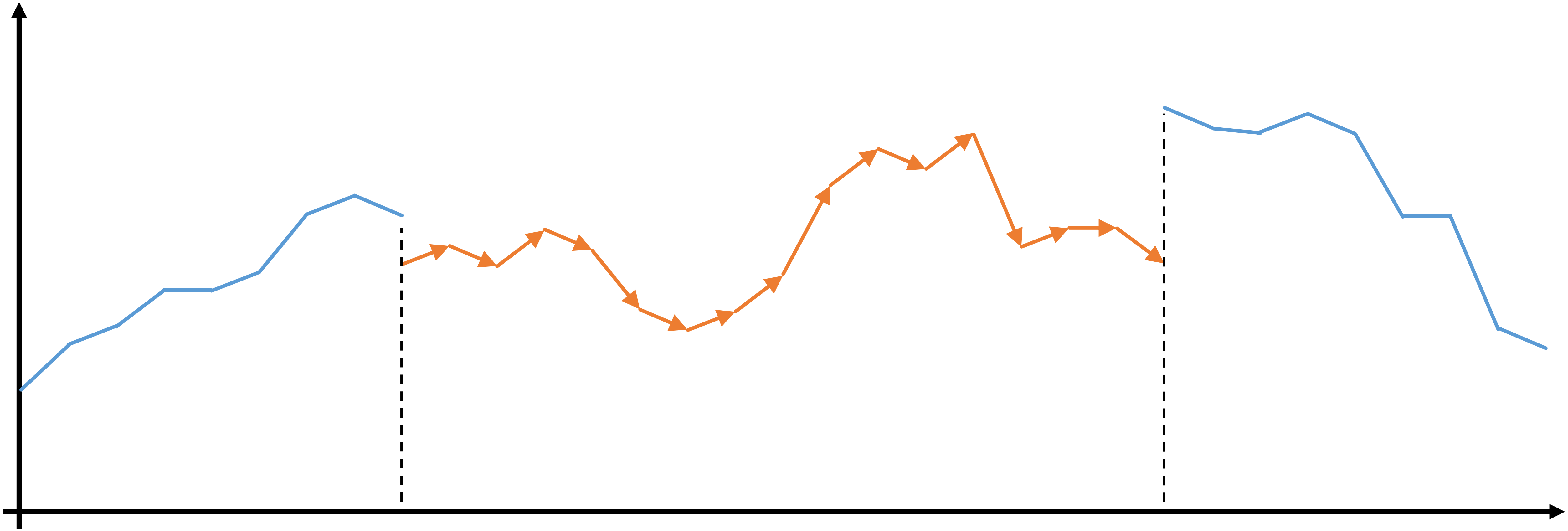
- Solve Laplace equation under Dirichlet boundary condition

- Fast approximation using Mean Value Coordinates

https://www.youtube.com/watch?v=AXvPeuc-wRw

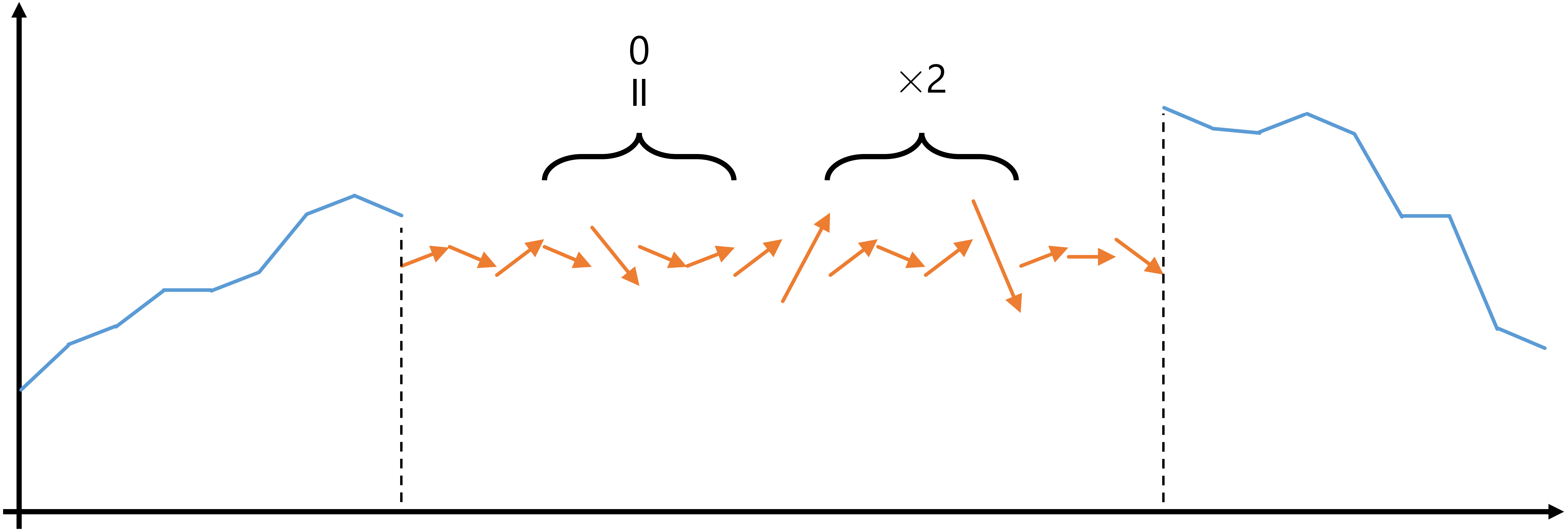
Mean value coordinates [Floater CAGD03]
Coordinates for instant image cloning [Farbman SIGGRAPH09]

Gradient-domain processing in general form (not just simple cloning)



Gradient-domain processing in general form (not just simple cloning)

Modify gradients arbitrarily!



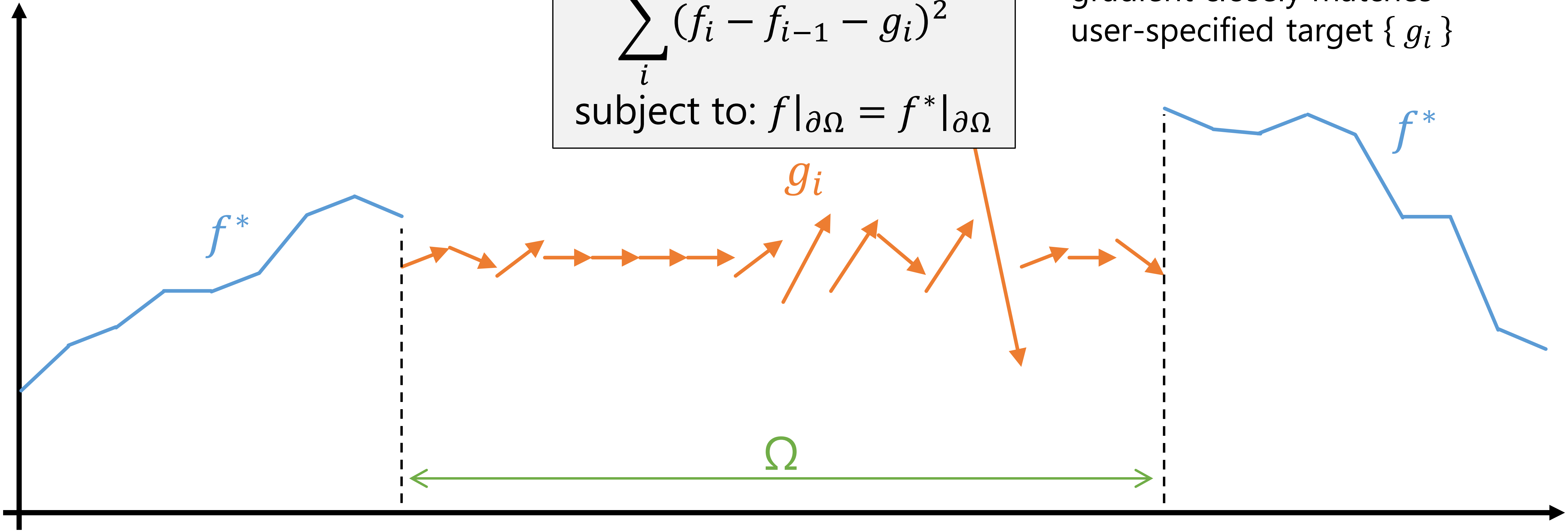
Gradient-domain processing in general form (not just simple cloning)

Find $\{ f_i \}$ that minimize

$$\sum_i (f_i - f_{i-1} - g_i)^2$$

subject to: $f|_{\partial\Omega} = f^*|_{\partial\Omega}$

Find function f such that its gradient closely matches user-specified target $\{ g_i \}$



1D case

Find $\{ f_i \}$ that minimize

$$\sum_i (f_i - f_{i-1} - g_i)^2$$

subject to: $f|_{\partial\Omega} = f^*|_{\partial\Omega}$

2D case

Find $f(x,y)$ that minimizes

$$\int_{(x,y)\in\Omega} \|\nabla f(x,y) - \mathbf{g}(x,y)\|^2$$

subject to: $f|_{\partial\Omega} = f^*|_{\partial\Omega}$

$\Updownarrow$

Solve Poisson equation:

$$\Delta f = \nabla \cdot \mathbf{g}$$

subject to: $f|_{\partial\Omega} = f^*|_{\partial\Omega}$

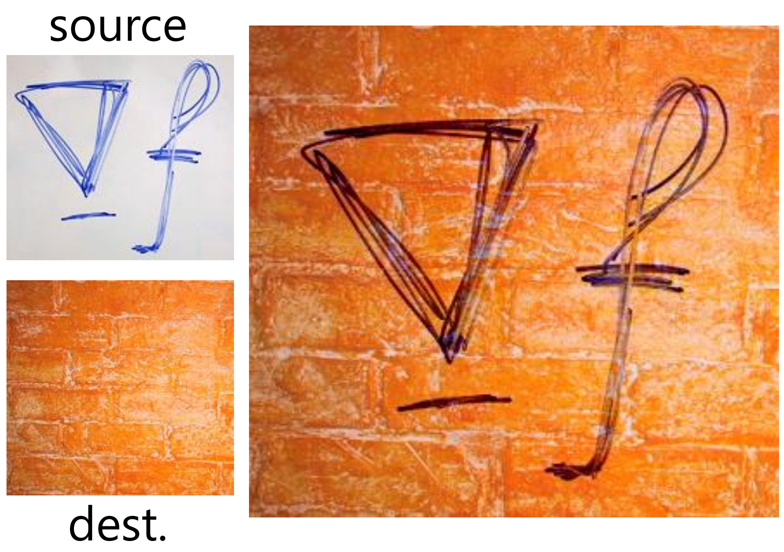
- Basics of Gradient-domain image processing:

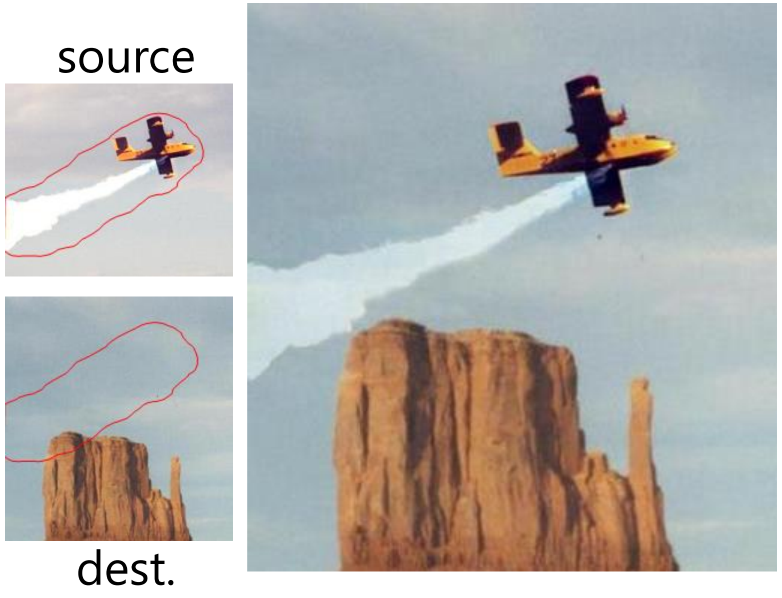
  Find image f whose gradient best matches user-specified target gradient field $\mathbf{g}$ by solving Poisson equation

How to give target gradients: mixing

- Copy source's gradient only when its magnitude is larger

  ➔ smooth part of source won't be copied





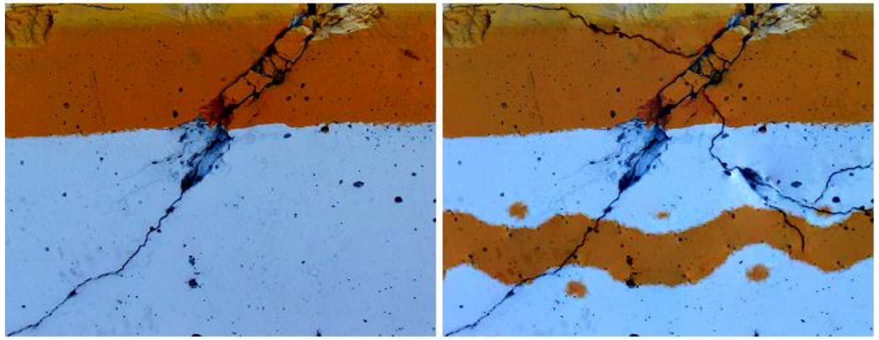
Poisson image editing [Perez SIGGRAPH03]

How to give target gradient: Edge Brush

- Copy gradients along object silhouette, paste along brush stroke
- Real-time Poisson solver implemented on GPU

Before After

Before After



https://www.youtube.com/watch?v=9MGjrsPzFc4

Real-time gradient-domain painting [McCann SIGGRAPH08]
Code: http://graphics.cs.cmu.edu/projects/gradient-paint/gradient_paint.r2403.tar.gz

How to give target gradient: modify original

Amplify/suppress within selected region
➔ Local Tone Mapping

Set to zero except where detected as edges
➔ Stylization

Poisson image editing [Perez SIGGRAPH03]

Extra:
Gradient-domain geometry processing

Gradient-domain geometry processing

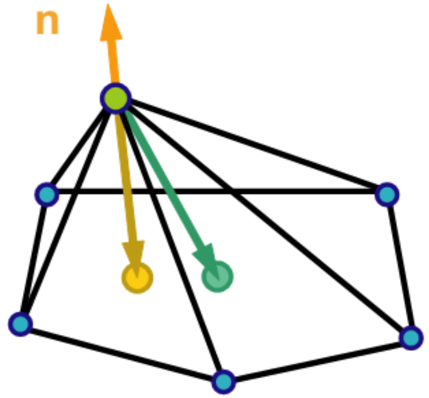
Find $\{ \mathbf{v}_i \}$ that minimize

$$\sum_{(i,j)\in E} w_{ij} \left\| \mathbf{v}_i - \mathbf{v}_j - \overline{\mathbf{e}_{ij}} \right\|^2$$

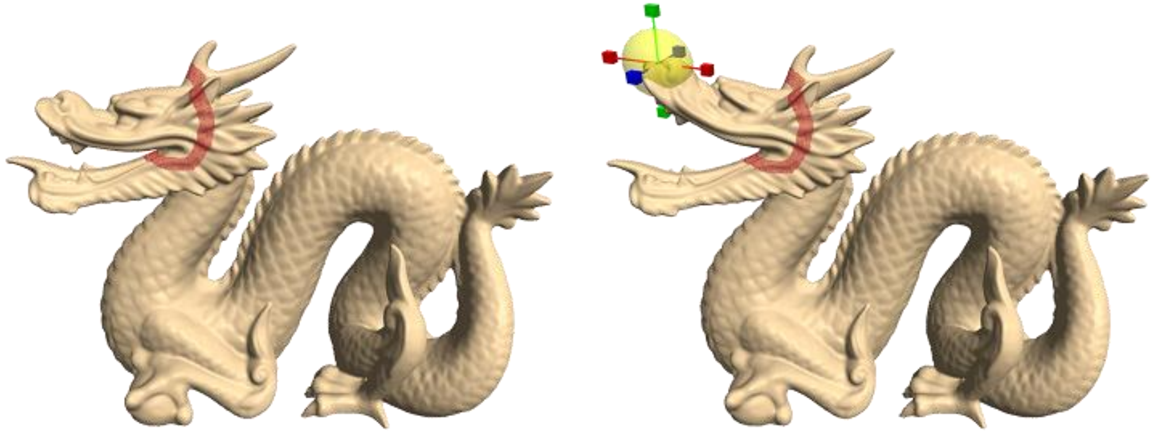
subject to: $\mathbf{v}_c = \mathbf{v}_c^*, c \in I_C$

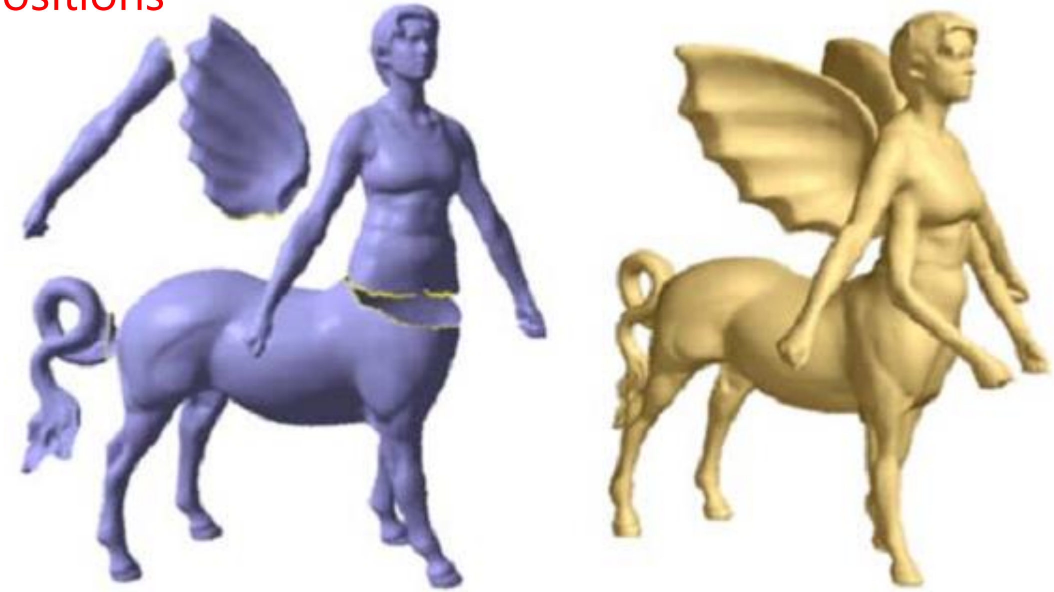
Edge vector of original shape
➔ target gradient

A few constraints of vertex positions
➔ boundary condition



Poisson equation

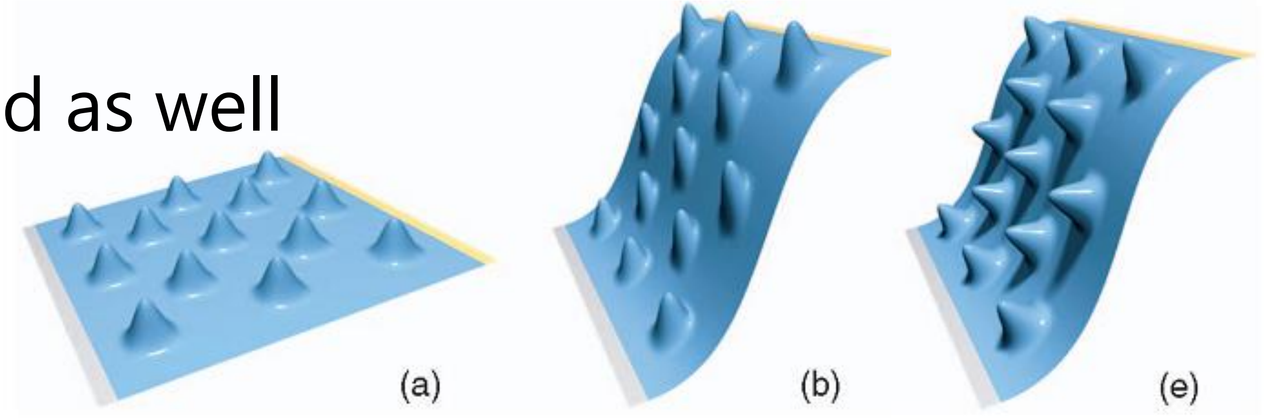




Mesh editing with poisson-based gradient field manipulation [Yu SIGGRAPH04]
Laplacian surface editing [Sorkine SGP04]
Interfaces and algorithms for the creation, modification, and optimization of surface meshes [Nealen PhD07]

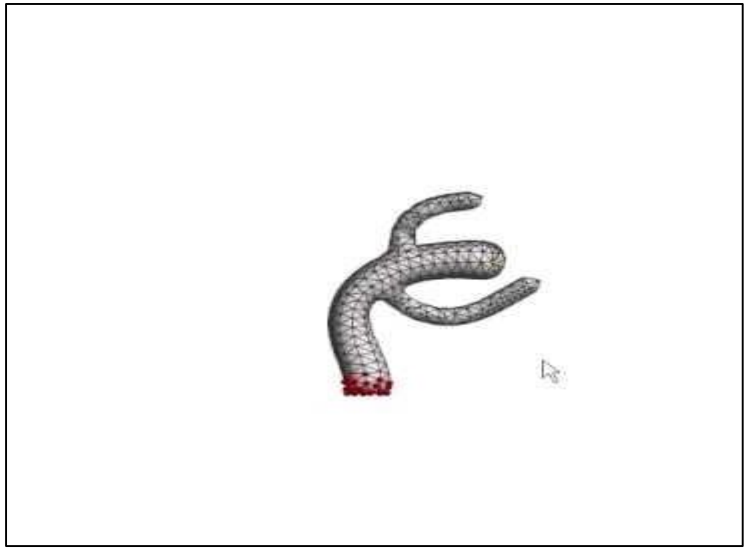
Rotation of local region due to large deformation

- Target gradient needs to be rotated as well
  - Non-linear relation
  - Optimal rotation difficult to find



Without rotation With rotation

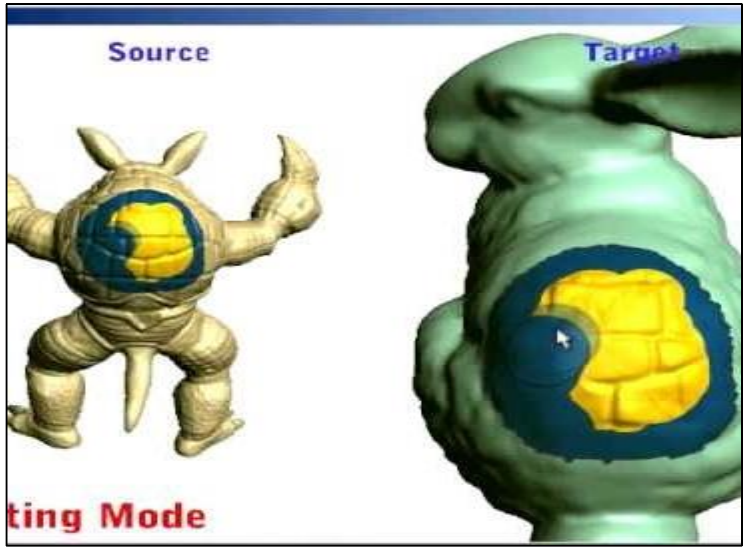
- Local-global optimization [Sorkine07]
  - **Local step**:
    Fix vertex positions,
    compute local rotation using SVD
  - **Global step**:
    Fix local rotations,
    compute vertex positions via Poisson equation



https://www.youtube.com/watch?v=ltX-qUjbkdc

On linear variational surface deformation methods [Botsch TVCG08]
As-rigid-as-possible surface modeling [Sorkine SGP07]

GeoBrush: Cloning brush for surface meshes

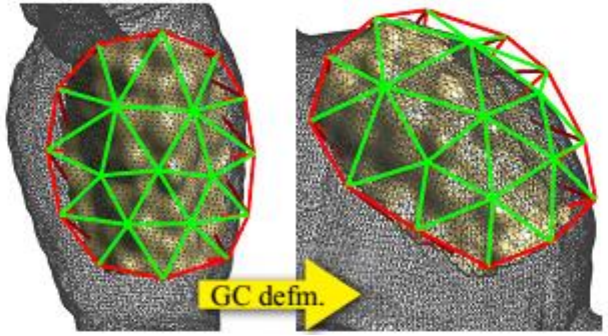


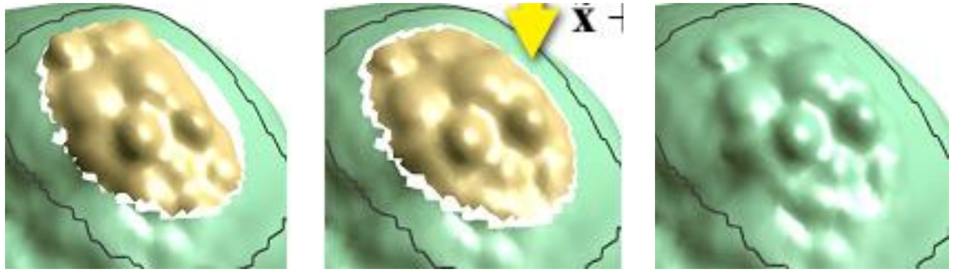
https://www.youtube.com/watch?v=FPsccn_gG8E

- Split deformation into two steps:

  1. Rotation of local region
     ➔ Fast & approx. computation using cage-based method

  2. Accurate offset
     ➔ Adapt GPU-based Poisson solver (originally for image processing)





GeoBrush: Interactive Mesh Geometry Cloning [Takayama EG11]