

Introduction to Computer Graphics

– Image Processing (1) –

June 10, 2020

Kenshi Takayama

Today's topics

- Edge-aware image processing



- Gradient-domain image processing

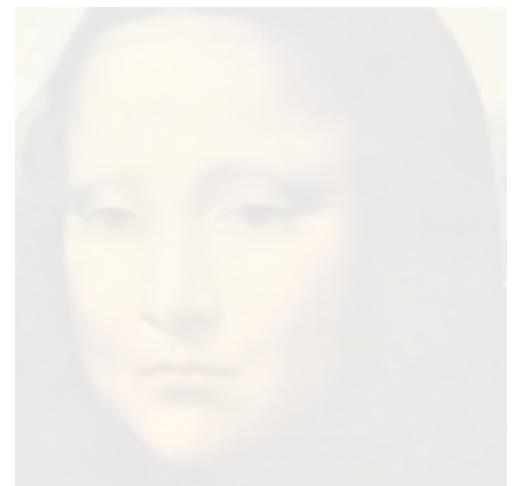
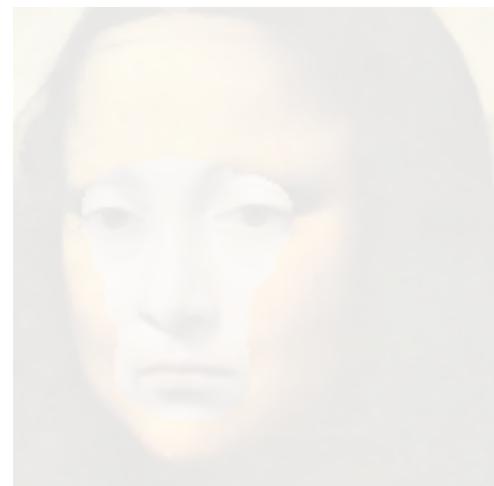


Image smoothing using Gaussian Filter

- Smoothness parameter σ



Original



$\sigma = 2$



$\sigma = 5$



$\sigma = 10$

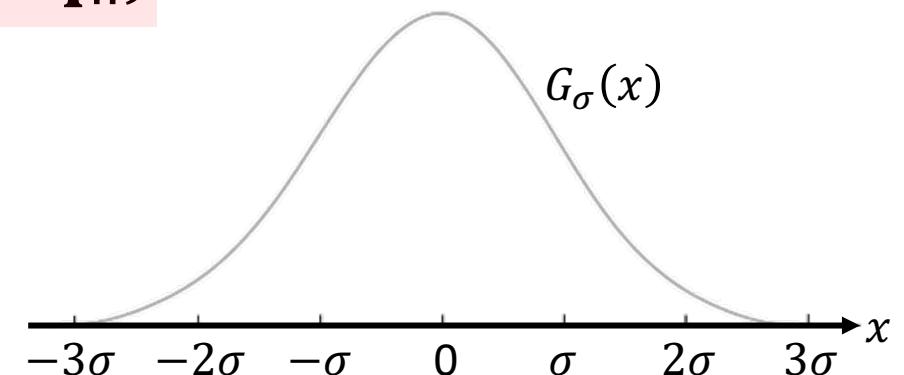
Equation of Gaussian Filter

- $I_{\mathbf{p}}$ represents pixel value of image I at position $\mathbf{p} = (p_x, p_y) \in \Omega$
 - For given resolution e.g. 640×480 , $\Omega := \{1, \dots, 640\} \times \{1, \dots, 480\}$
- $\text{GF}_{\sigma}[I]$ represents filtered image with Gaussian parameter σ :

$$\text{GF}_{\sigma}[I]_{\mathbf{p}} := \frac{\sum_{\mathbf{q} \in \Omega} G_{\sigma}(\|\mathbf{p} - \mathbf{q}\|) I_{\mathbf{q}}}{\sum_{\mathbf{q} \in \Omega} G_{\sigma}(\|\mathbf{p} - \mathbf{q}\|)}$$

$W_{\mathbf{p}}$

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

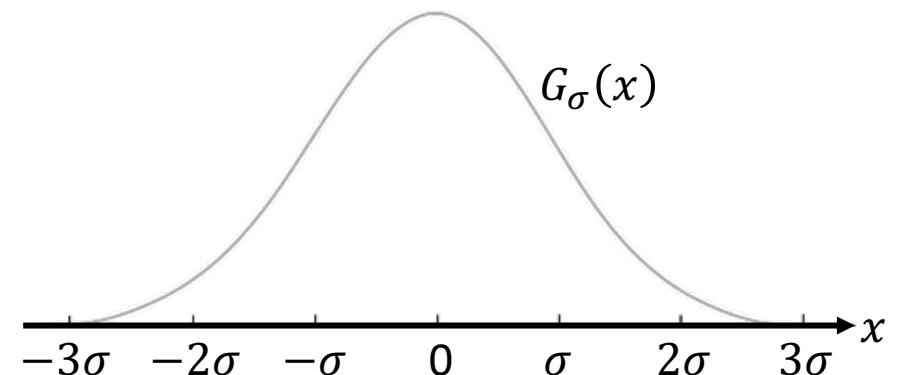


Equation of Gaussian Filter

- $I_{\mathbf{p}}$ represents pixel value of image I at position $\mathbf{p} = (p_x, p_y) \in \Omega$
 - For given resolution e.g. 640×480 , $\Omega := \{1, \dots, 640\} \times \{1, \dots, 480\}$
- $\text{GF}_{\sigma}[I]$ represents filtered image with Gaussian parameter σ :

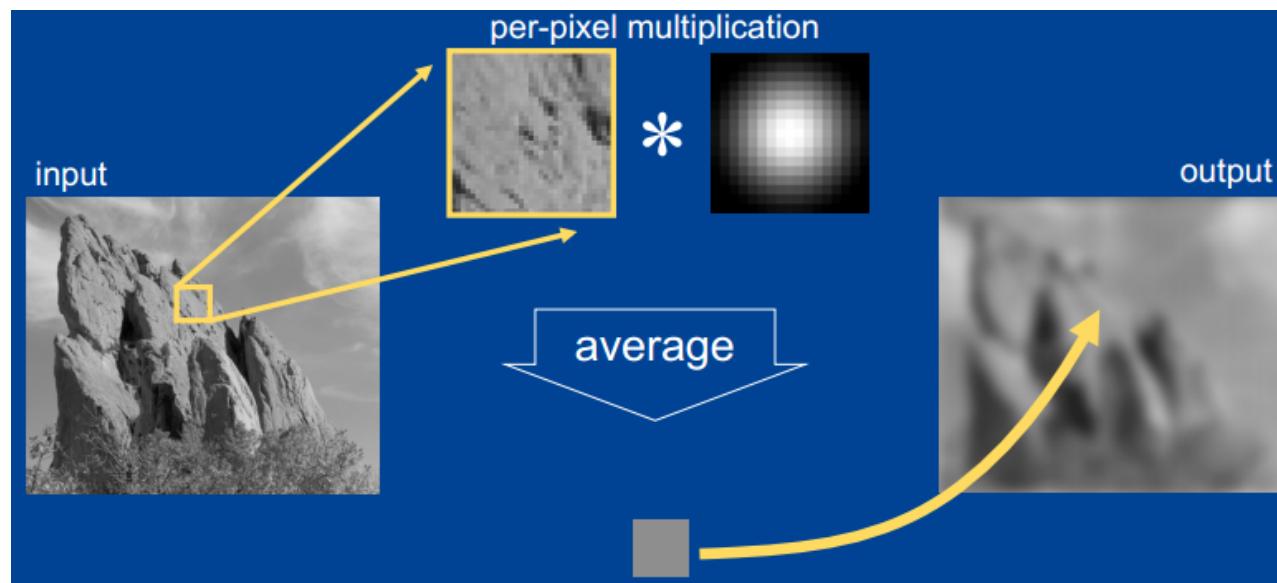
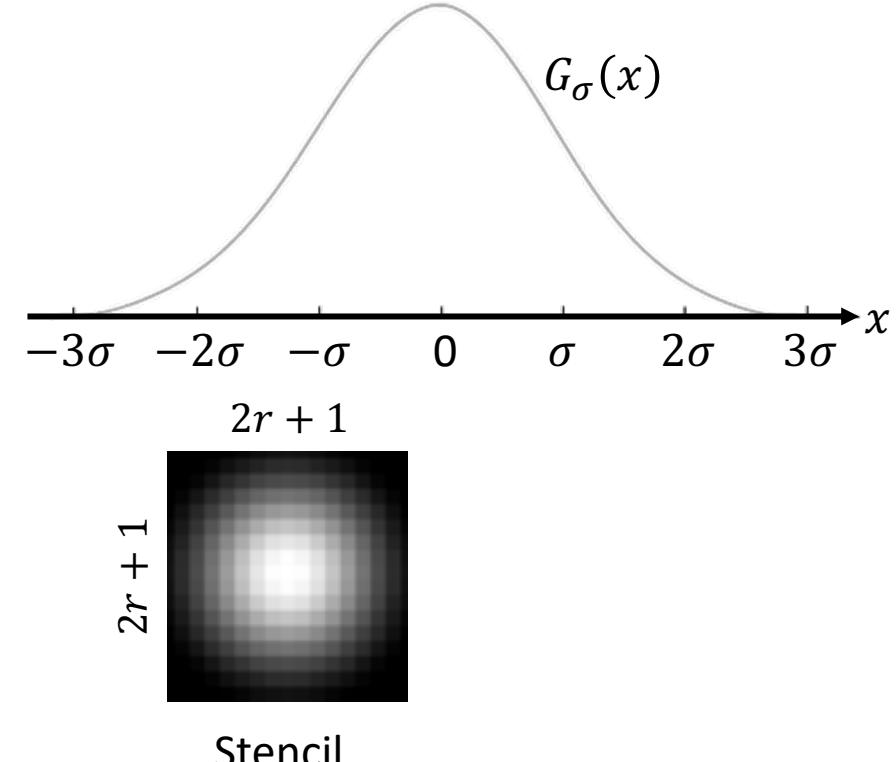
$$\text{GF}_{\sigma}[I]_{\mathbf{p}} := \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in \Omega} G_{\sigma}(\|\mathbf{p} - \mathbf{q}\|) I_{\mathbf{q}}$$

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



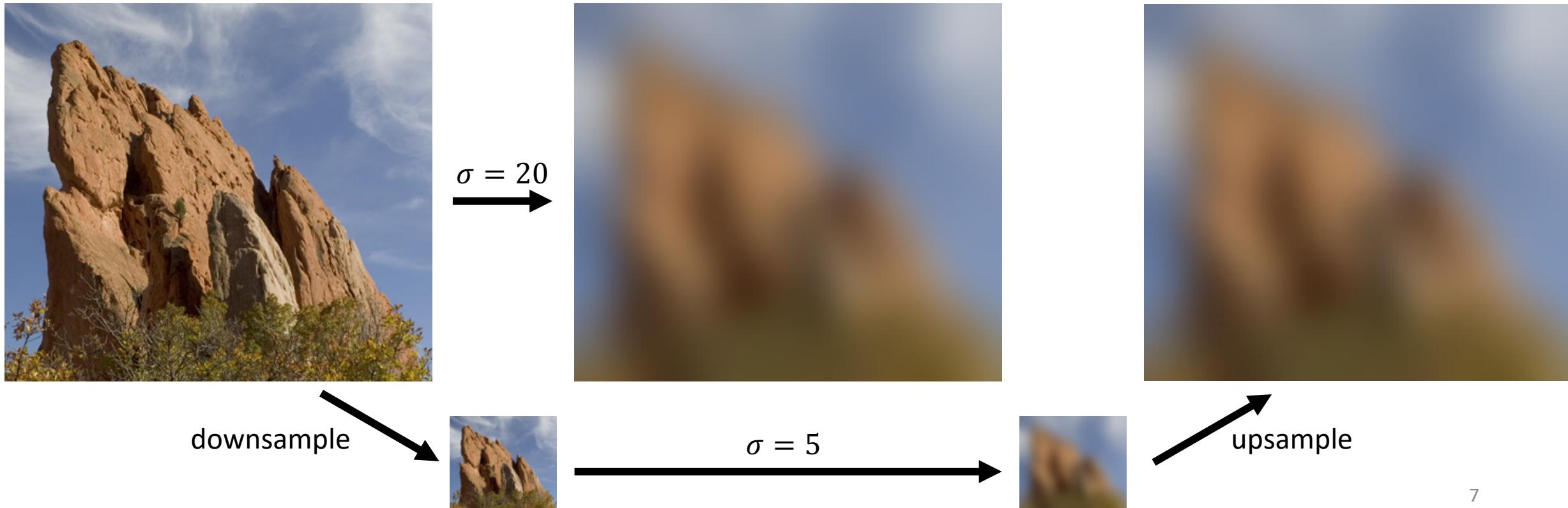
Implementing Gaussian Filter

- $G_\sigma(3\sigma) \approx 0 \rightarrow$ Distant pixels can be ignored
- For fixed size $r := \text{ceil}(3\sigma)$, precompute weights on a $(2r + 1) \times (2r + 1)$ stencil



When kernel radius σ is very large

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



Detail Extraction & Enhancement



-



smoothed

=



detail

halos!

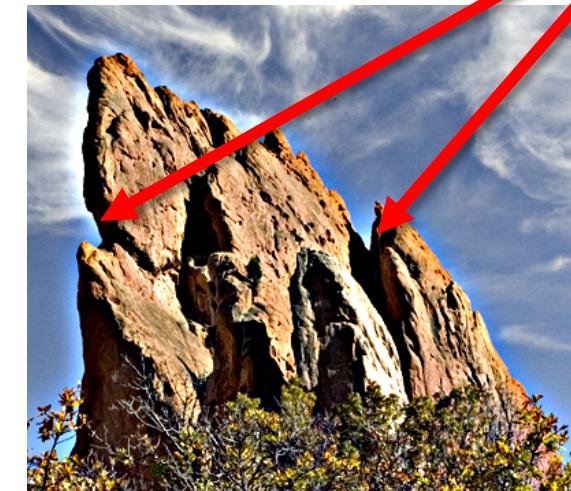


+ 3×



detail

=



enhanced

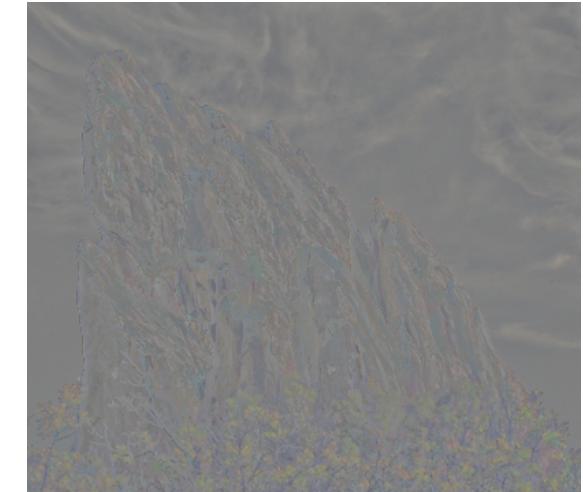
When using edge-aware smoothing, ...



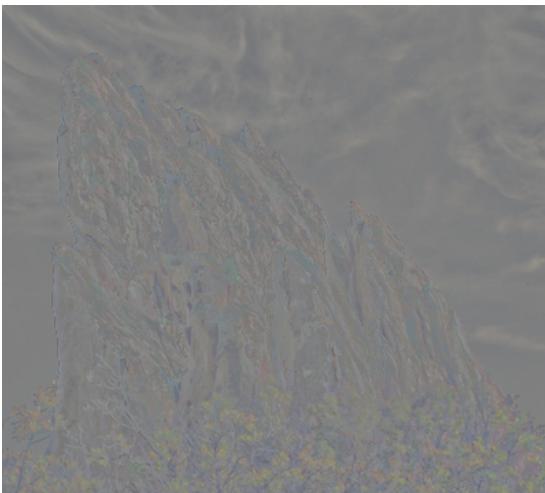
-



=



+ 3×



=



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

$$BF_{\sigma_s, \sigma_r}[I]_p := \frac{1}{W_p} \sum_{q \in \Omega} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(\|I_p - I_q\|) I_q$$

In all cases, $\sigma_s = 10$



Original



$\sigma_r = 32$



$\sigma_r = 128$



$\sigma_r = 512_{10}$

Application of Bilateral Filter: Stylization



Application of Bilateral Filter: Tone Mapping

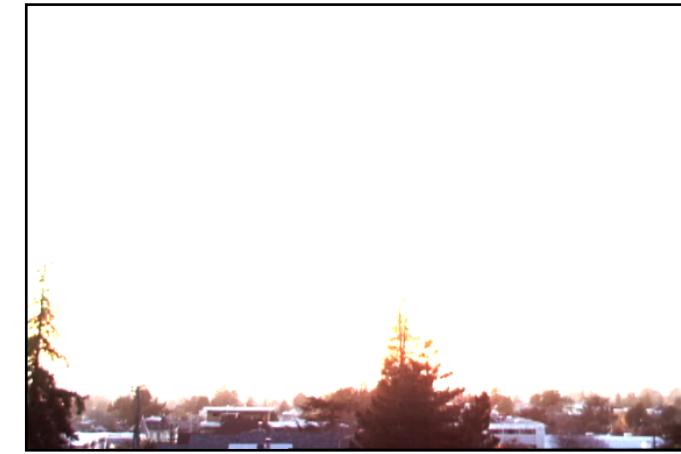
- Range of each channel (24bit color image): 1~255
- Range of light intensity in the real world: 1~ 10^5
 - High Dynamic Range image
 - Can be obtained by photographing with different exposure times



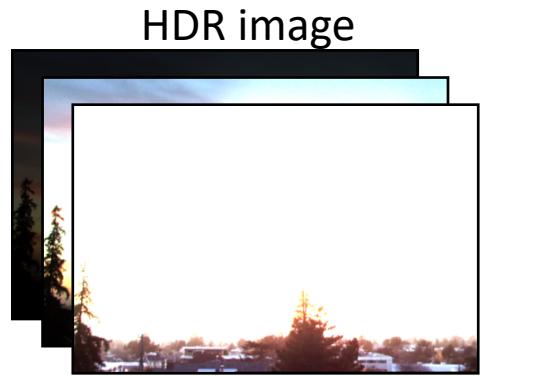
Short exposure



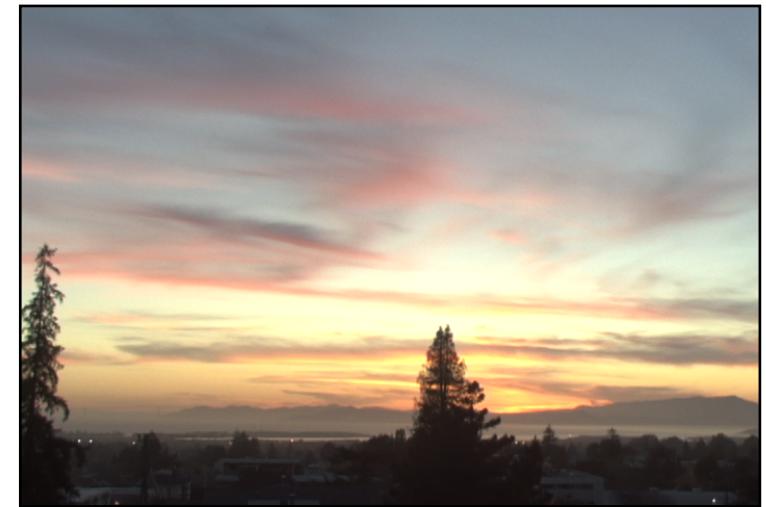
Long exposure



Application of Bilateral Filter: Tone Mapping



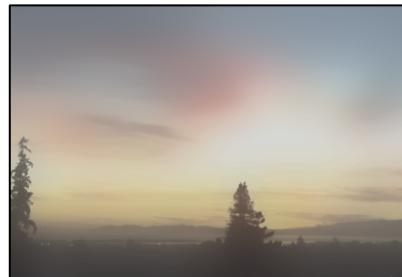
γ correct. ($X \rightarrow X^\gamma$)



Bilateral Filter



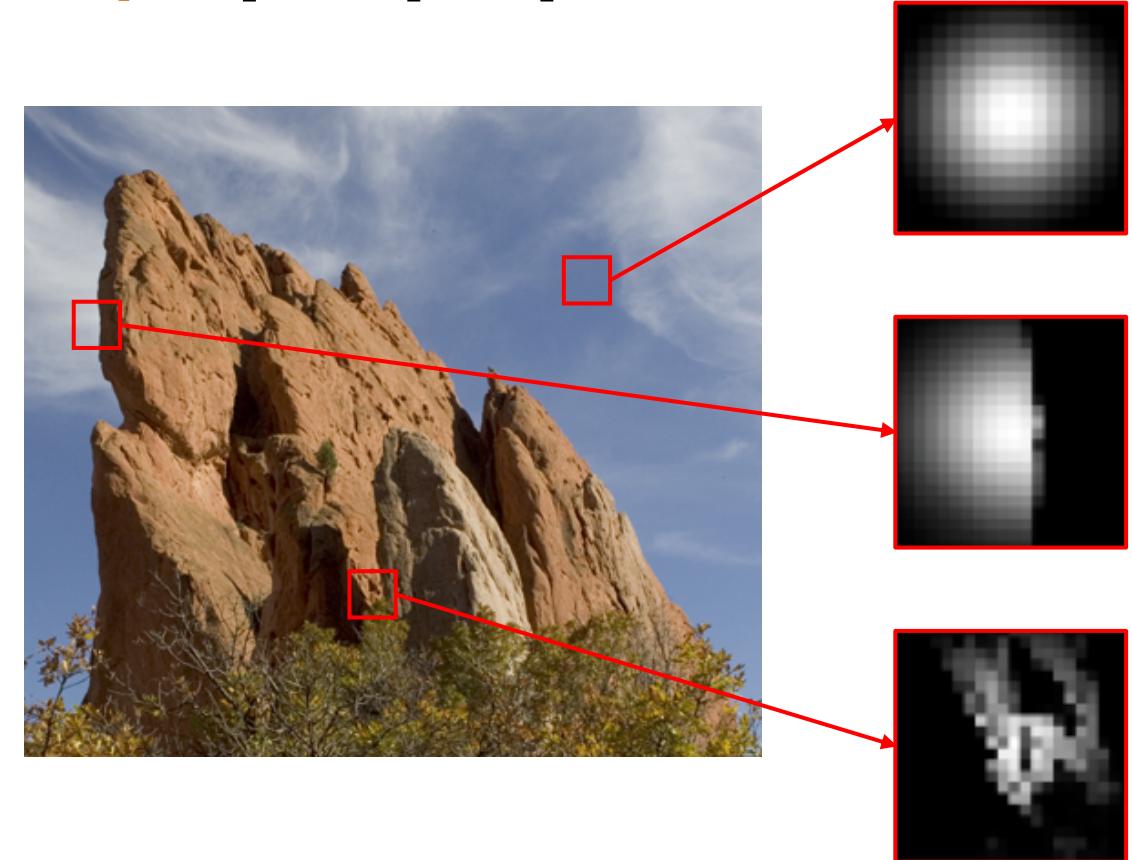
γ correct.



Naïve implementation of Bilateral Filter

$$\frac{1}{W_p} \sum_{q \in \Omega} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(\|I_p - I_q\|) I_q$$

- Recompute stencil for every pixel location $p \in \Omega$
→ slow
- (Basic Assignment)



Another view toward Bilateral Filter

- Define **feature vector** $\mathbf{f}_p := \left(\frac{p}{\sigma_s}, \frac{I_p}{\sigma_r} \right)$ for pixel location p and intensity I_p

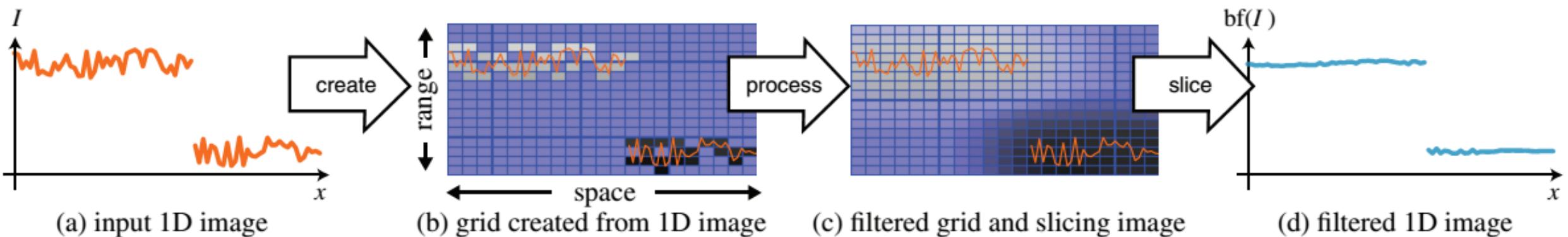
- Weight of Bilateral Filter is equivalent to Gaussian kernel applied to Euclidean distance in the feature space

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

$$\begin{aligned} & G_{\sigma_s}(\|p - q\|)G_{\sigma_r}(\|I_p - I_q\|) \\ &= \exp\left(-\frac{\|p - q\|^2}{2\sigma_s^2}\right)\exp\left(-\frac{\|I_p - I_q\|^2}{2\sigma_r^2}\right) \\ &= \exp\left(-\frac{\|\mathbf{f}_p - \mathbf{f}_q\|^2}{2}\right) \\ &= G_1(\|\mathbf{f}_p - \mathbf{f}_q\|) \end{aligned}$$

Bilateral Grid [Paris06; Chen07]

- Define 3D feature space as (X-coord, Y-coord, intensity), map sample points $\{f_p\}$ to 3D grid
- The larger σ_s & σ_r , the coarser the grid \rightarrow lower comput. cost



Weight map generation using feature space



White scribble → constraint of weight=1

Black scribble → constraint of weight=0

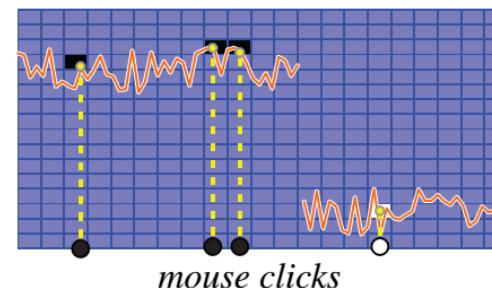
Weight map



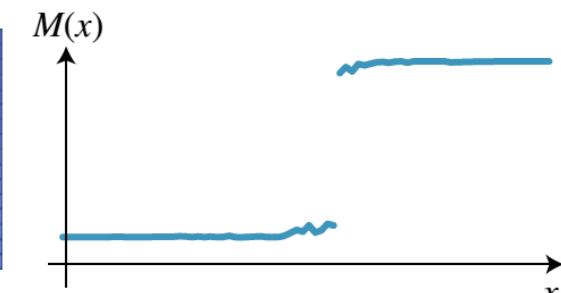
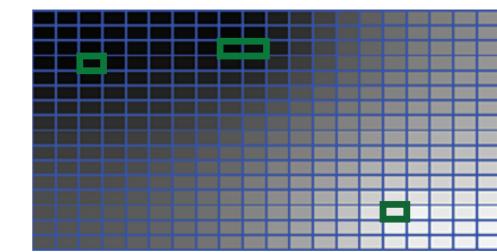
Application: color adjustment

- Various names: Edit Propagation, Matting, Segmentation

- Solve Laplace equation on Bilateral Grid

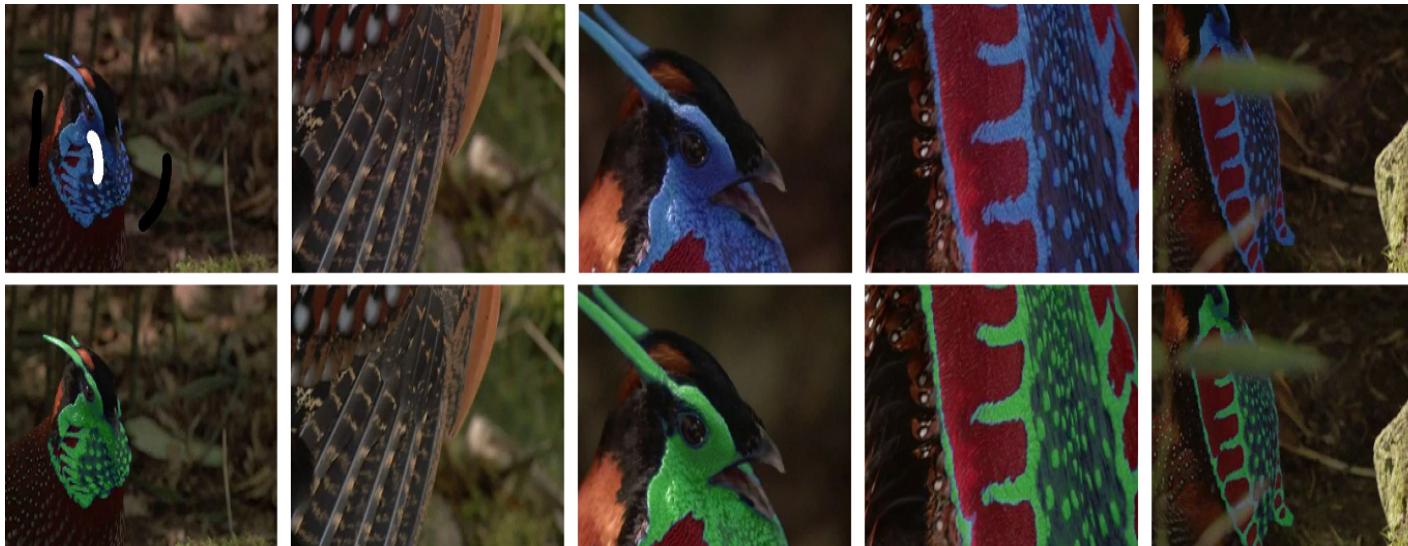


mouse clicks



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)

Eurographics 2013
May 6-10, Girona (Spain)

Bilateral Hermite Radial Basis Functions for Contour-based Volume Segmentation

T. Ijiri¹, S. Yoshizawa¹, Y. Sato², M. Ito², H. Yokota¹
¹RIKEN, ²National Cancer Center Hospital East

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

Extension to BF: Joint (Cross) Bilateral Filter

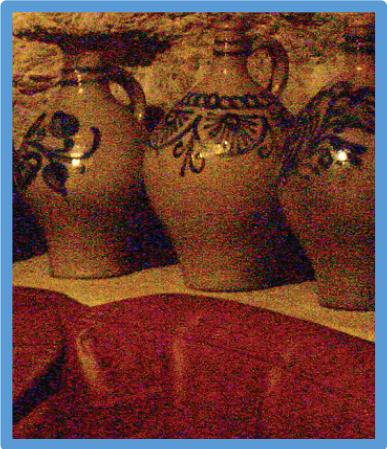


Photo A: without flash

😊 Correct color

😢 Noisy, blurred

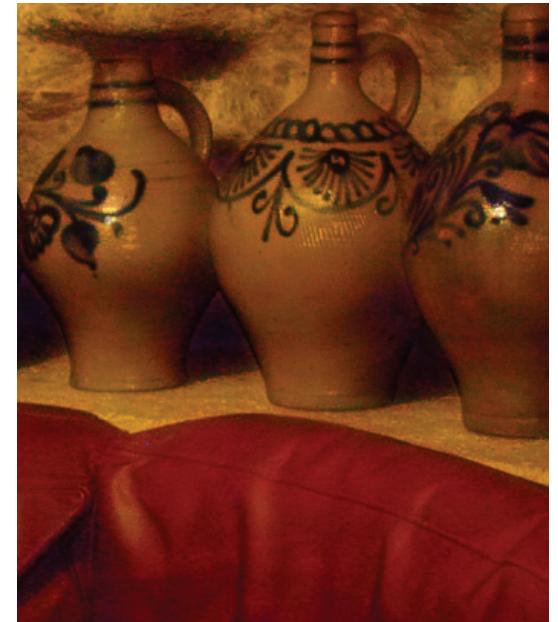


Photo F: with flash

😢 Incorrect color

😊 Less noisy, sharp

After applying JBF



$$\text{JBF}_{\sigma_s, \sigma_r}(A, F)_p := \frac{1}{W_p} \sum_{q \in \Omega} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(\|F_p - F_q\|) A_q$$

Extension to BF: Non-Local Means Filter

- Define feature space by **neighborhood vector \mathbf{n}_p** representing 7×7 sub-image centered at \mathbf{p}

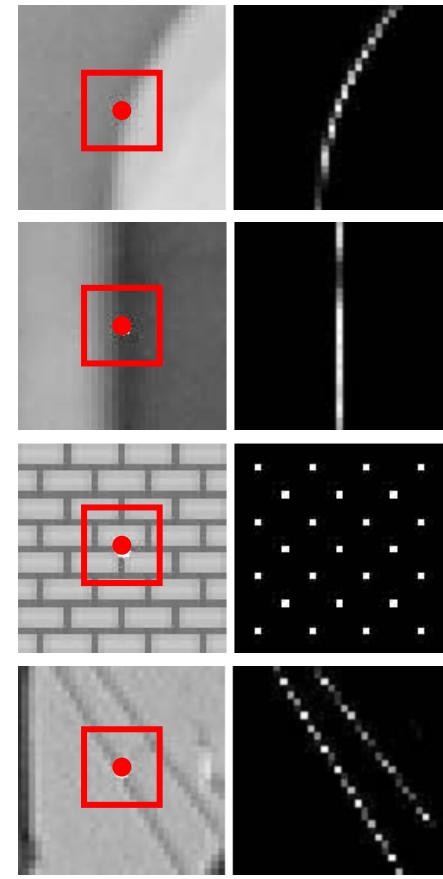
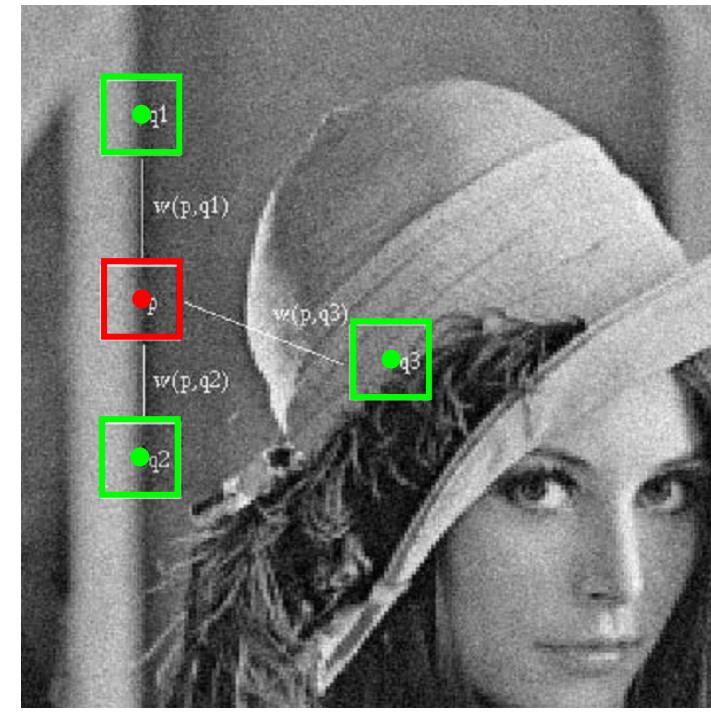
$$\text{NLMF}_{\sigma}(I)_{\mathbf{p}} := \frac{1}{W_p} \sum_{\mathbf{q} \in \Omega} G_{\sigma}(\|\mathbf{n}_p - \mathbf{n}_q\|) I_q$$



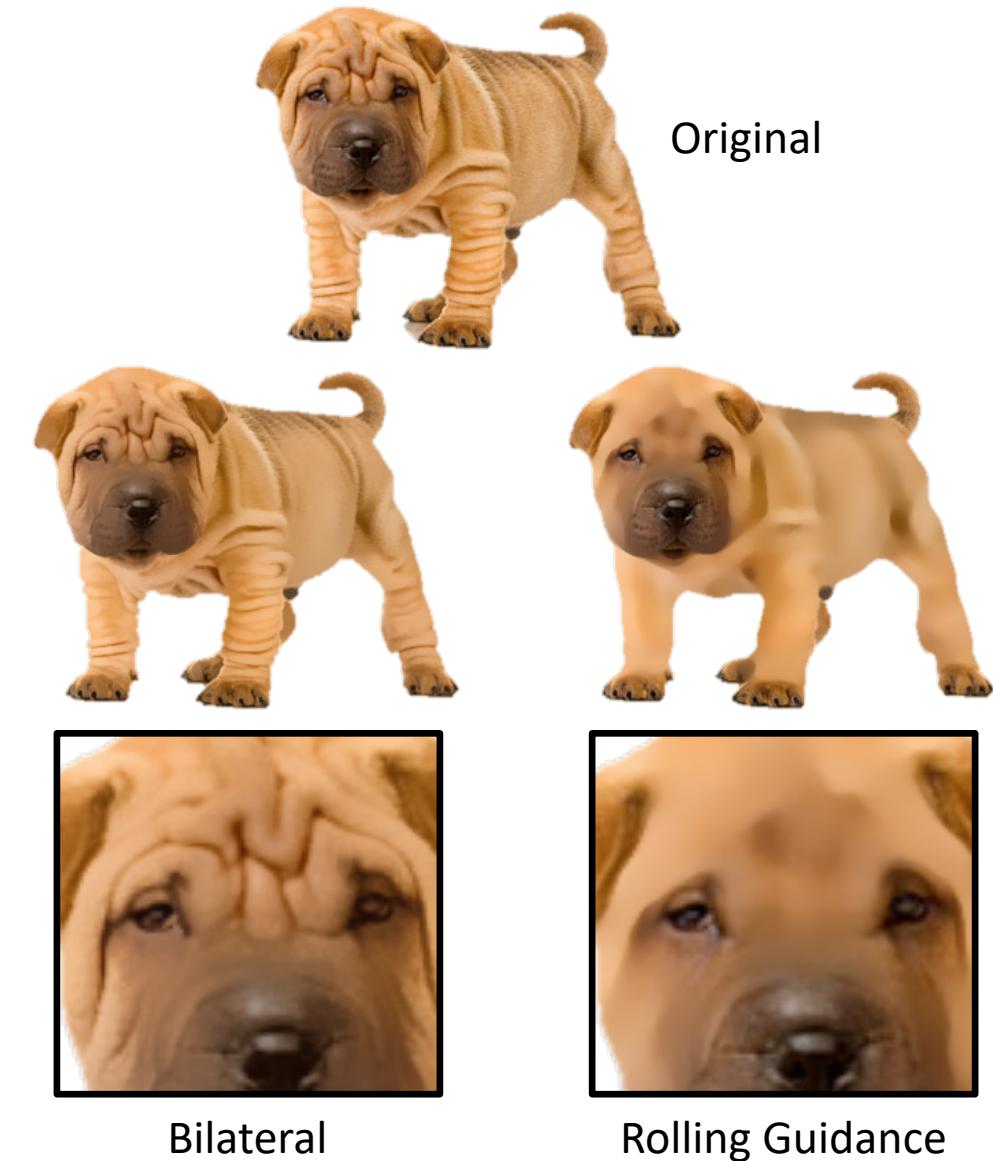
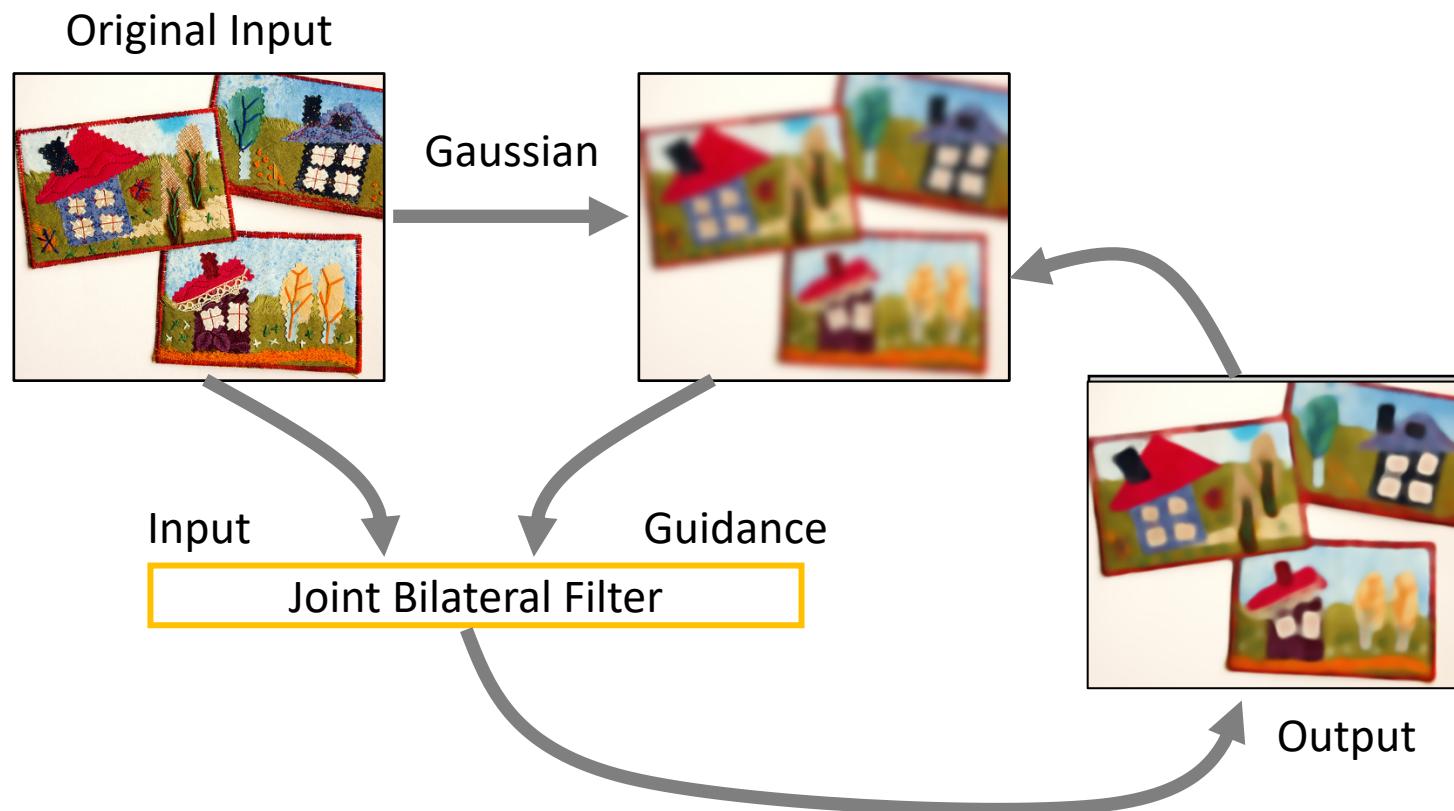
Noisy input

Bilateral

NL Means



Extension to BF: Rolling Guidance Filter

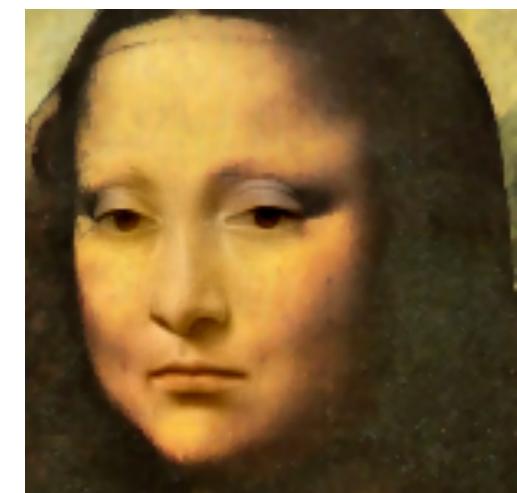
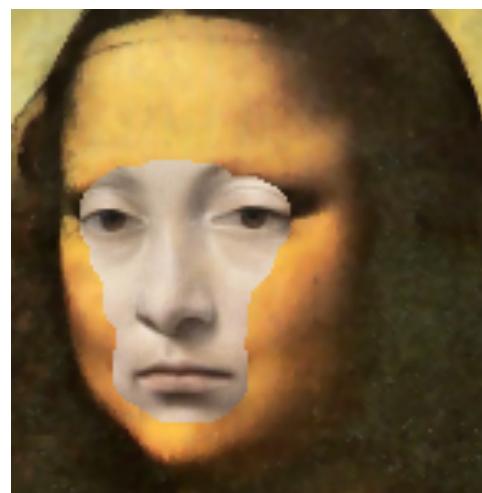


Today's topics

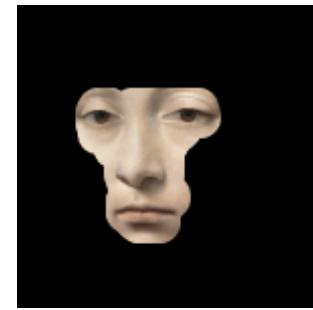
- Edge-aware image processing



- Gradient-domain image processing



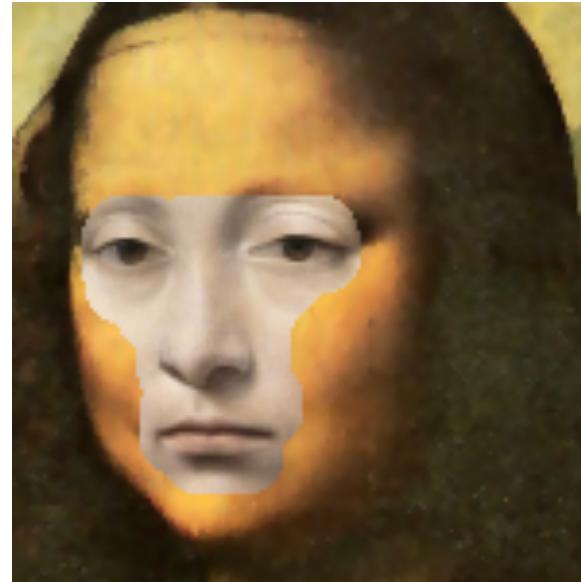
Scenario: insert source img. into destination img.



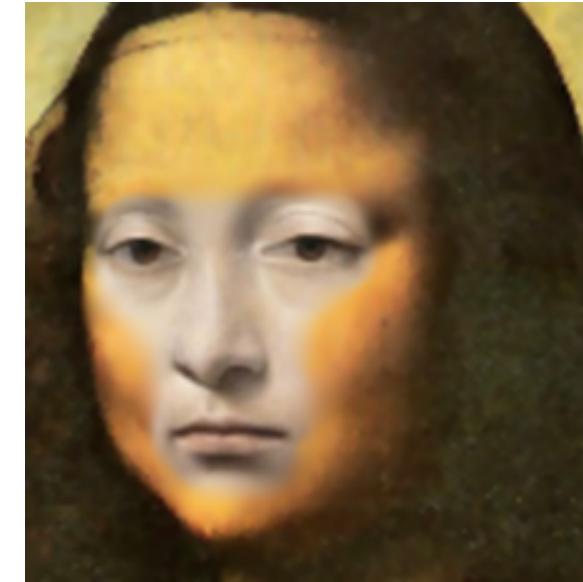
Source



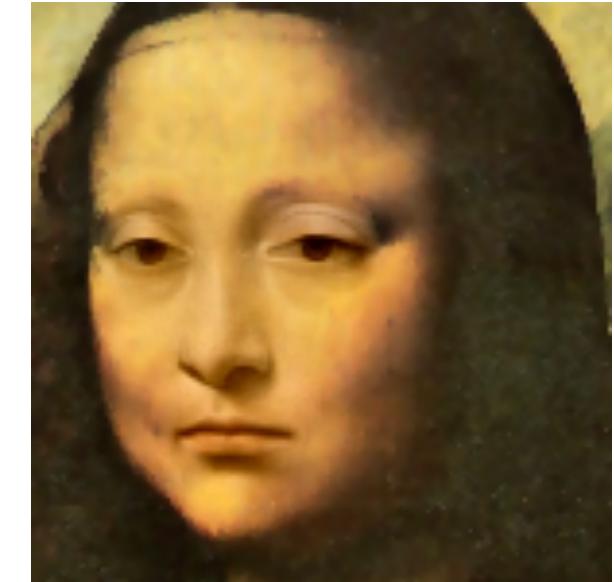
Dest.



Simple copying



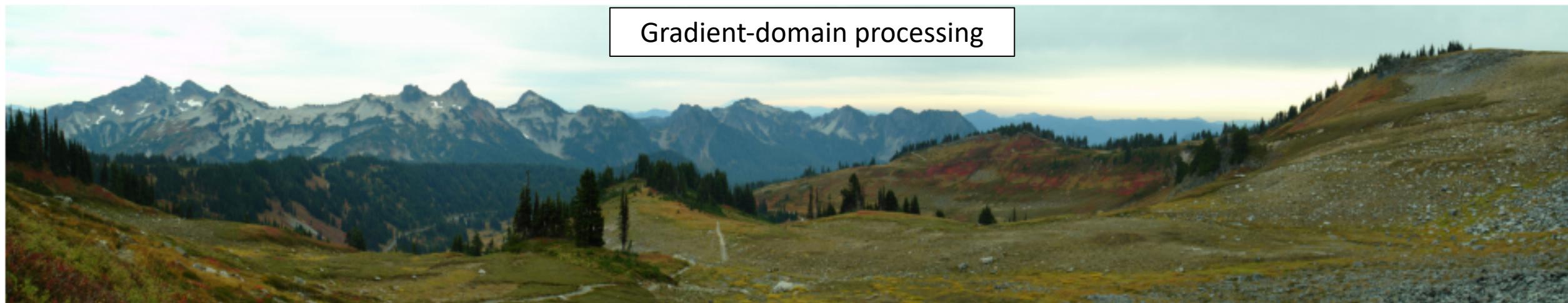
Boundary blurred



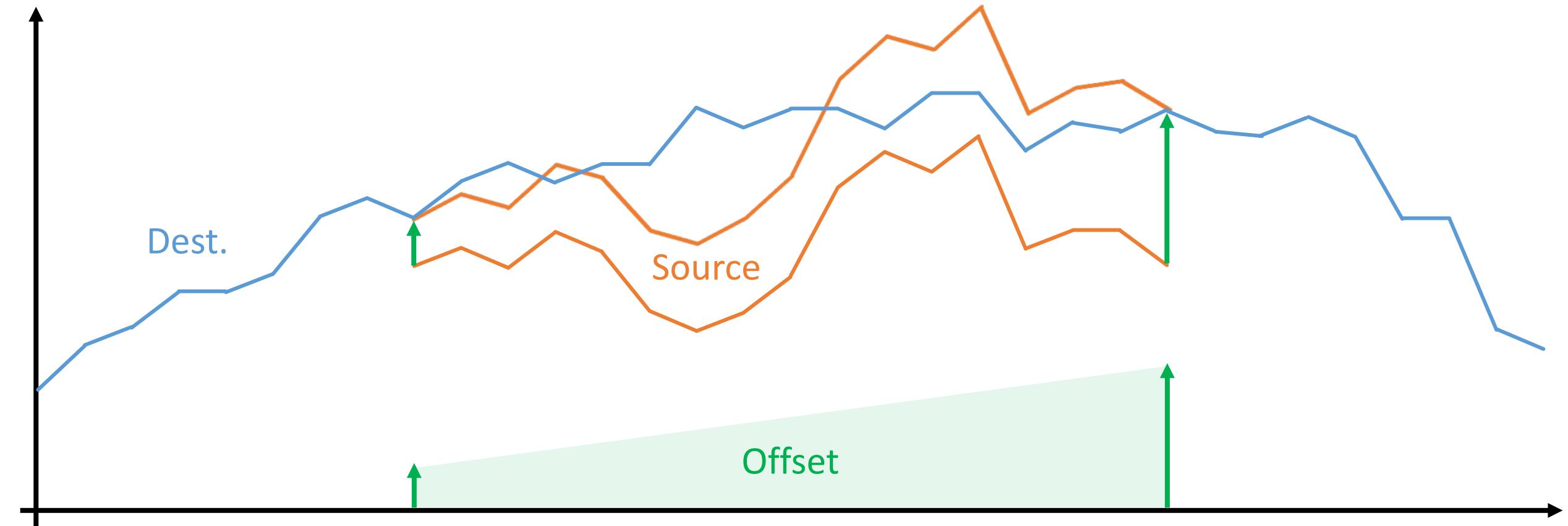
Gradient-domain processing



Scenario: generating panorama from several shots



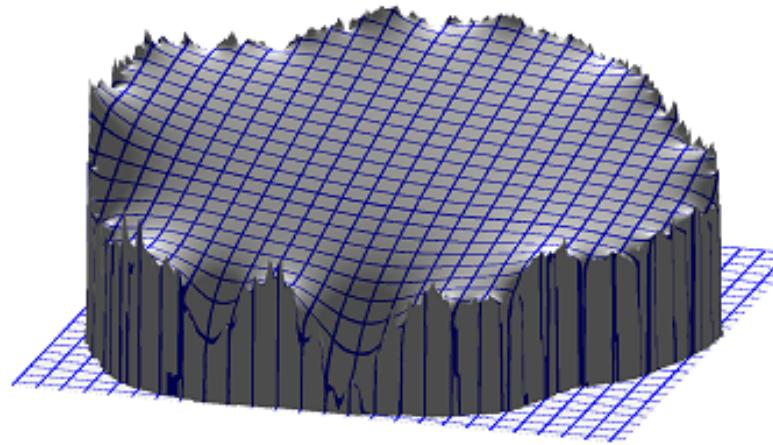
Simple case of 1D grayscale image



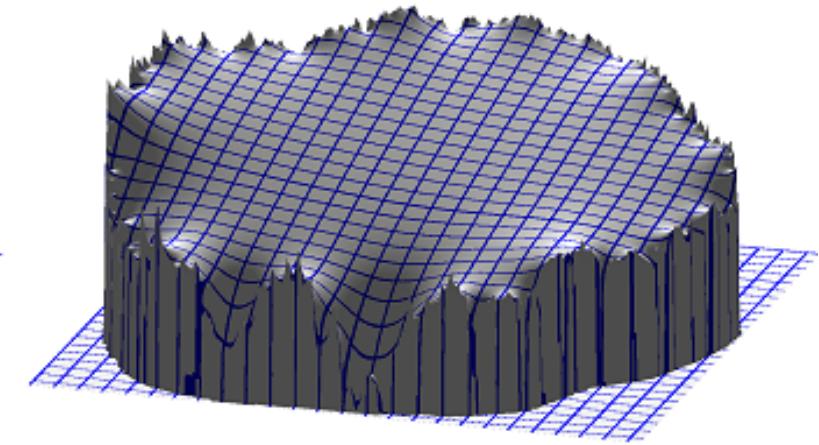
2D case: offset by Laplace Membrane



(a) Source patch



(b) Laplace membrane



(c) Mean-value membrane

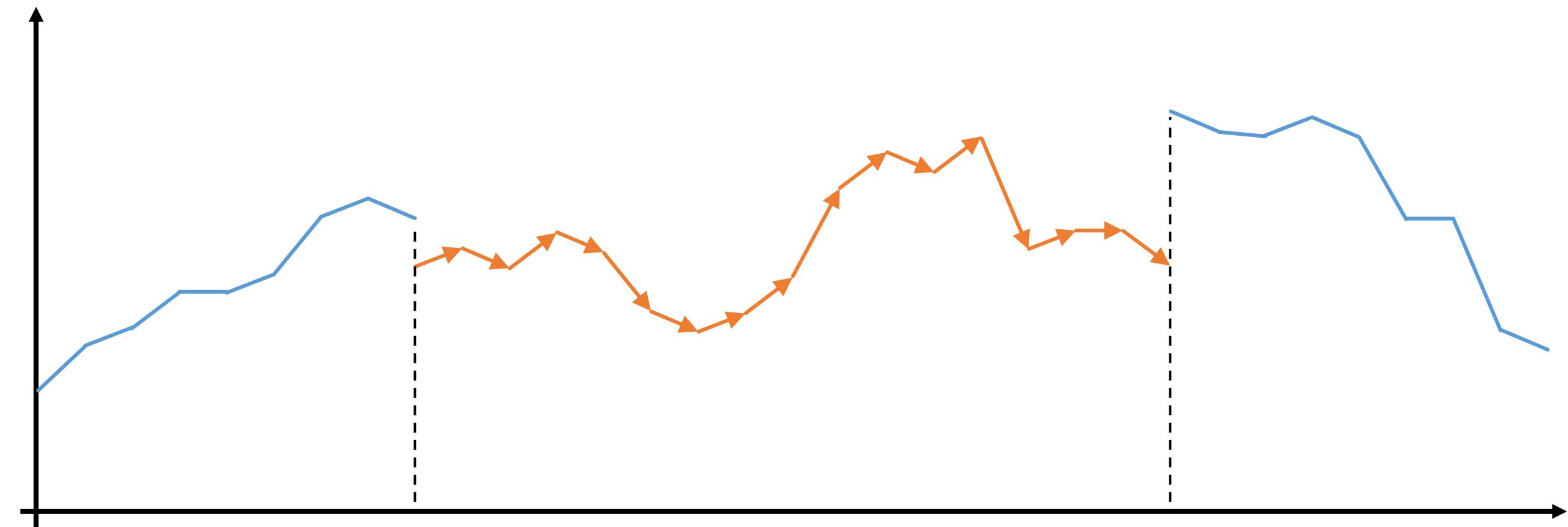
- Solve Laplace equation under Dirichlet boundary condition
- Fast approximation using Mean Value Coordinates

Coordinates for
Instant Image Cloning

Zeev Farbman
Gil Hoffer
Yaron Lipman
Daniel Cohen-Or
Dani Lischinski

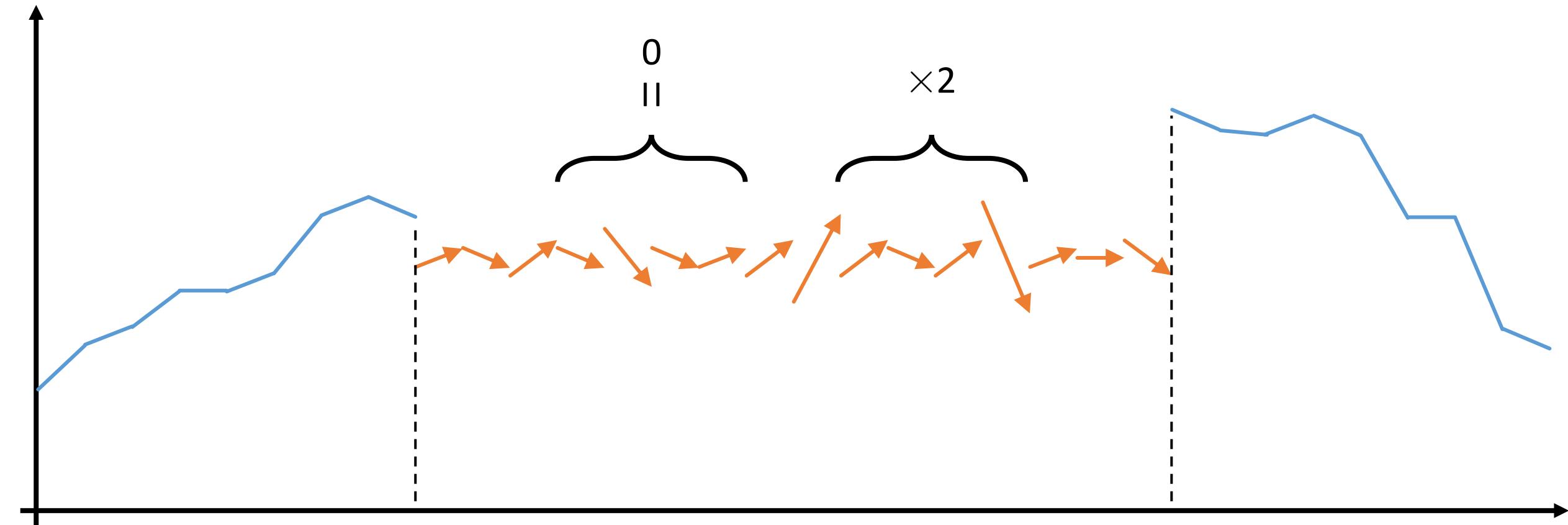
ACM Transactions on Graphics
(Proceedings of ACM SIGGRAPH 2009)

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

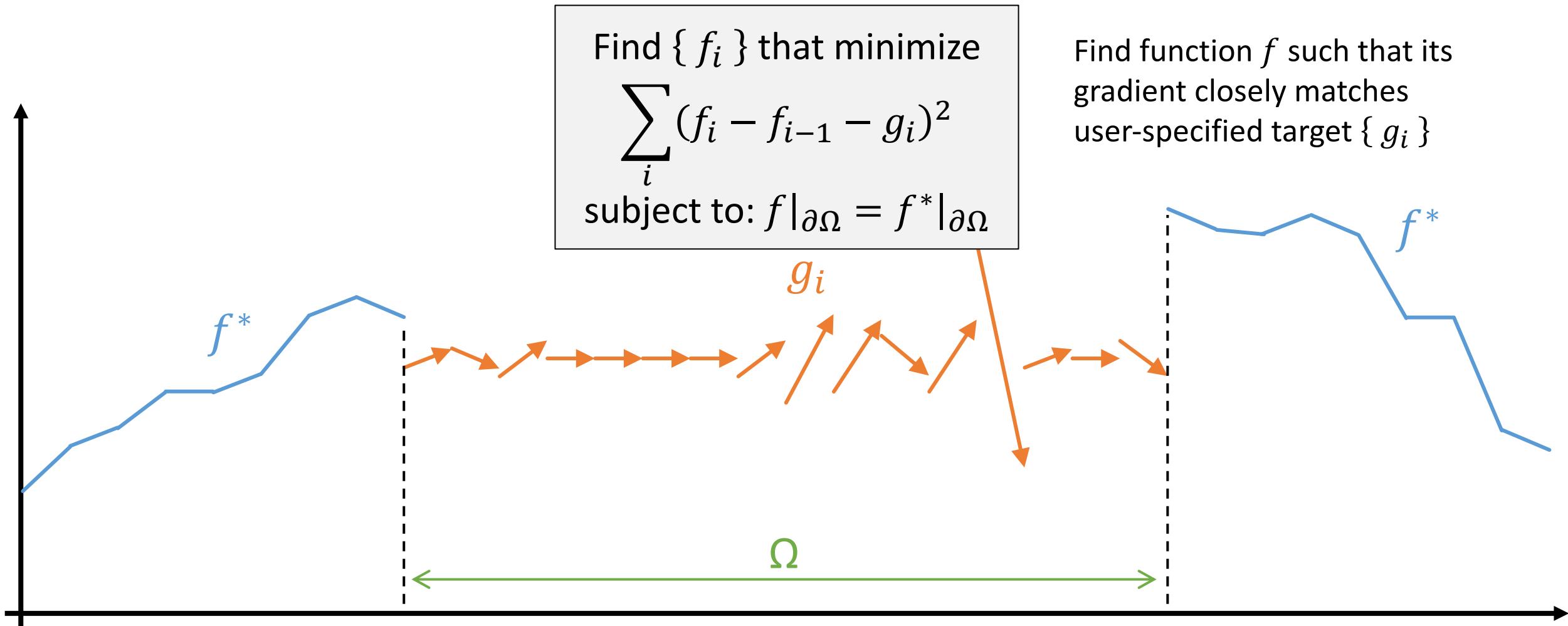


Gradient-domain processing in general form (not just simple cloning)

Modify gradients arbitrarily!



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



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



- Basics of Gradient-domain image processing:

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

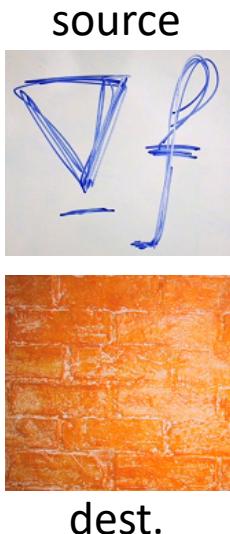
Solve **Poisson equation**:

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

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

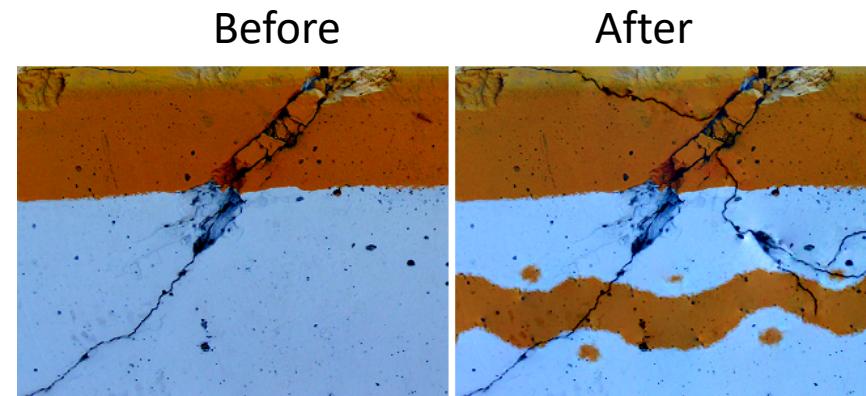
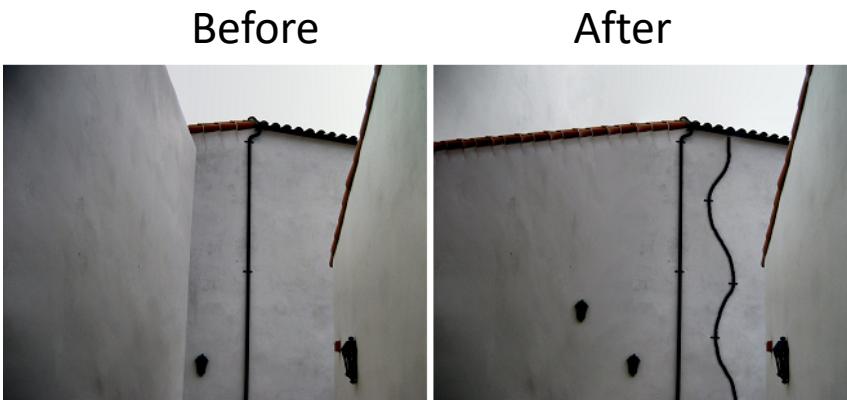
How to give target gradients: mixing

- Copy source's gradient only when its magnitude is larger
→ smooth part of source won't be copied



How to give target gradient: Edge Brush

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



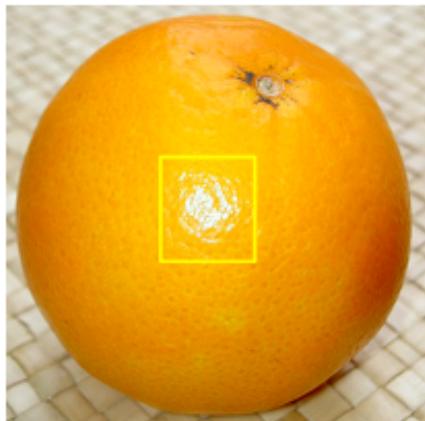
Real-Time
Gradient-Domain
Painting

James McCann Nancy Pollard
Carnegie Mellon University

(With Audio)

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

How to give target gradient: modify original



Amplify/suppress within selected region
→ Local Tone Mapping



Set to zero except where detected as edges
→ Stylization

Extra:
Gradient-domain geometry processing

Gradient-domain geometry processing

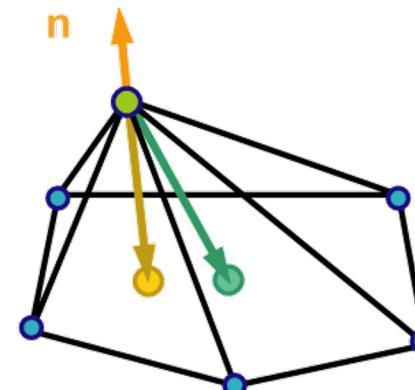
Find $\{ \mathbf{v}_i \}$ that minimize

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

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

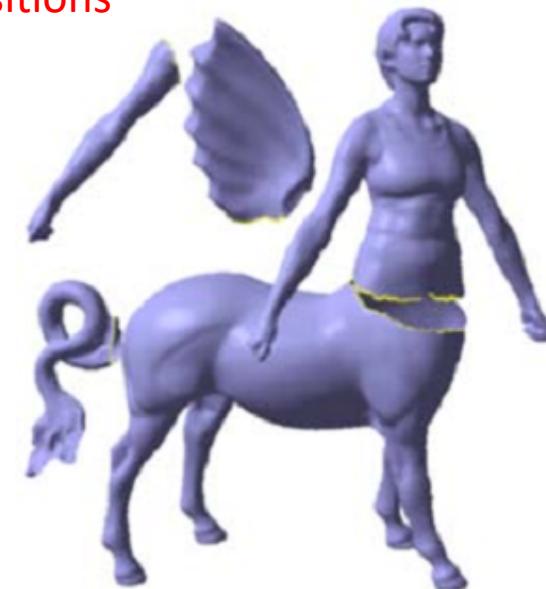
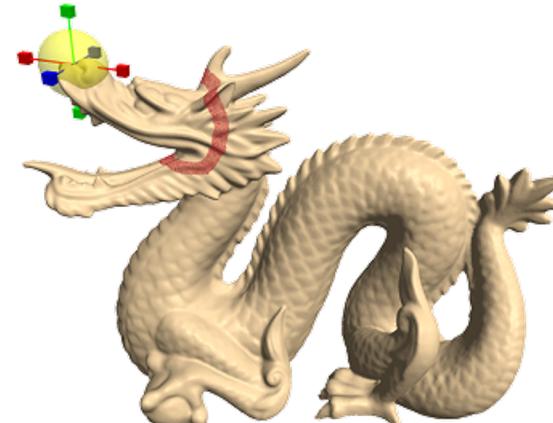
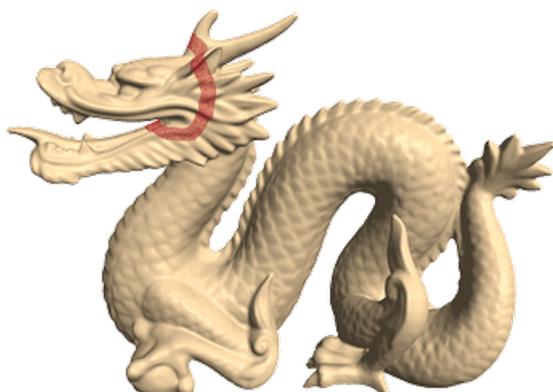
Edge vector of original shape
→ target gradient

A few constraints of vertex positions
→ boundary condition



Poisson equation

$$\boxed{\mathbf{L}} \quad \boxed{\mathbf{x}} = \boxed{\boldsymbol{\delta}}$$



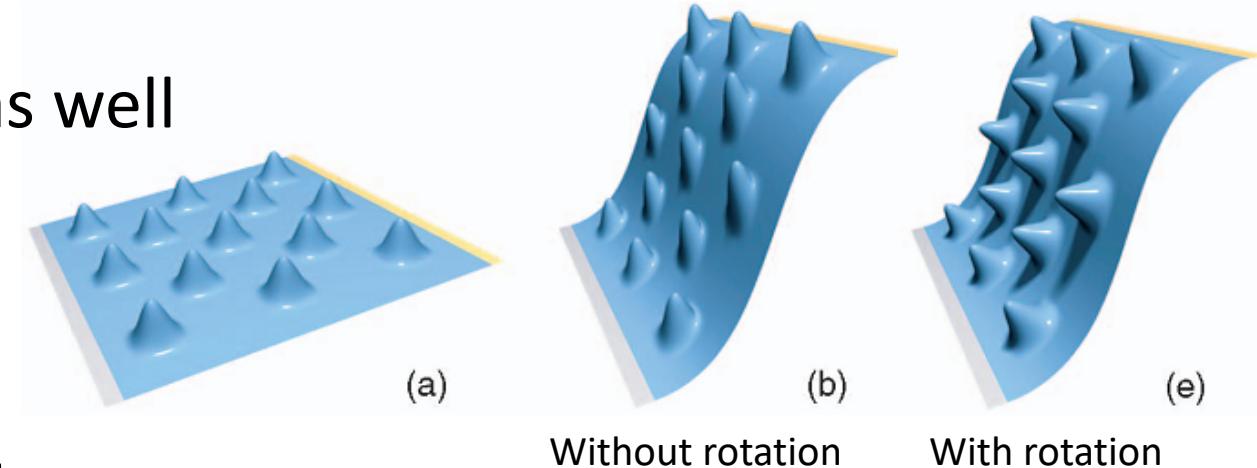
Mesh editing with poisson-based gradient field manipulation [Yu SIGGRAPH04]

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



- 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

**As-Rigid-As-Possible
Surface Modeling**

Olga Sorkine Marc Alexa
TU Berlin



<https://www.youtube.com/watch?v=ItX-qUjbkdc>

GeoBrush: Cloning brush for surface meshes

GeoBrush:
Interactive Mesh Geometry Cloning

Kenshi Takayama^{1,2}

Ryan Schmidt³

Karan Singh³

Takeo Igarashi¹

Tamy Boubekeur⁴

Olga Sorkine²

¹The University of Tokyo

²New York University

³University of Toronto

⁴Telecom ParisTech - CNRS LTCI

https://www.youtube.com/watch?v=FPscn_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)

