コンピュータグラフィクス論

– 画像処理(1) –

2017年6月8日

高山 健志

本日のトピック

- Edge-aware な画像処理


- Gradient-domain の画像処理

Gaussian Filter による画像平滑化

- 「滑らかさ」パラメタ σ

元画像

$\sigma = 2$

$\sigma = 5$

$\sigma = 10$

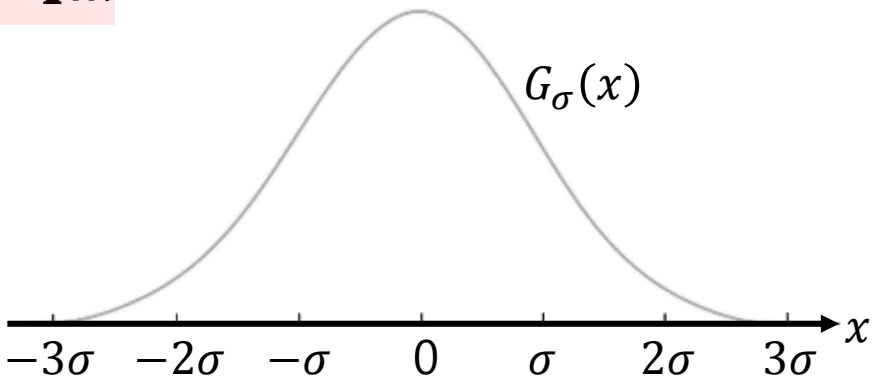
Gaussian Filter の数式

- 画像 I のピクセル位置 $\mathbf{p} = (p_x, p_y) \in \Omega$ における画素値を $I_{\mathbf{p}}$ で表す
  - 解像度640×480の場合、$\Omega := \{1, \cdots, 640\} \times \{1, \cdots, 480\}$

- パラメタ σ による Gaussian Filter 適用後の画像を $\mathrm{GF}_\sigma[I]$ で表す

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

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



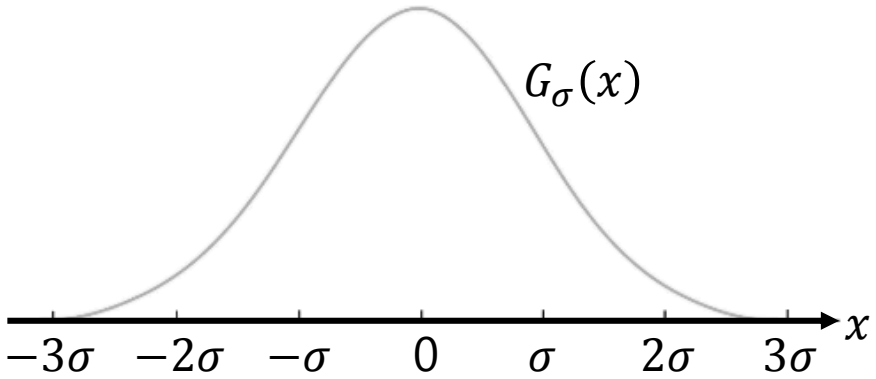
Gaussian Filter の数式

- 画像 I のピクセル位置 $\mathbf{p} = (p_x, p_y) \in \Omega$ における画素値を $I_{\mathbf{p}}$ で表す
  - 解像度640×480の場合、$\Omega \coloneqq \{1, \cdots, 640\} \times \{1, \cdots, 480\}$

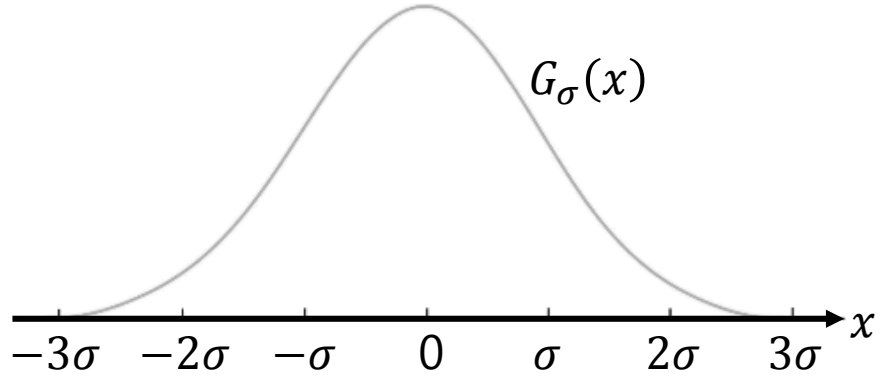
- パラメタ σ による Gaussian Filter 適用後の画像を $\mathrm{GF}_\sigma[I]$ で表す

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

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

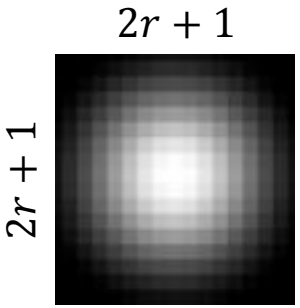


Gaussian Filter の実装

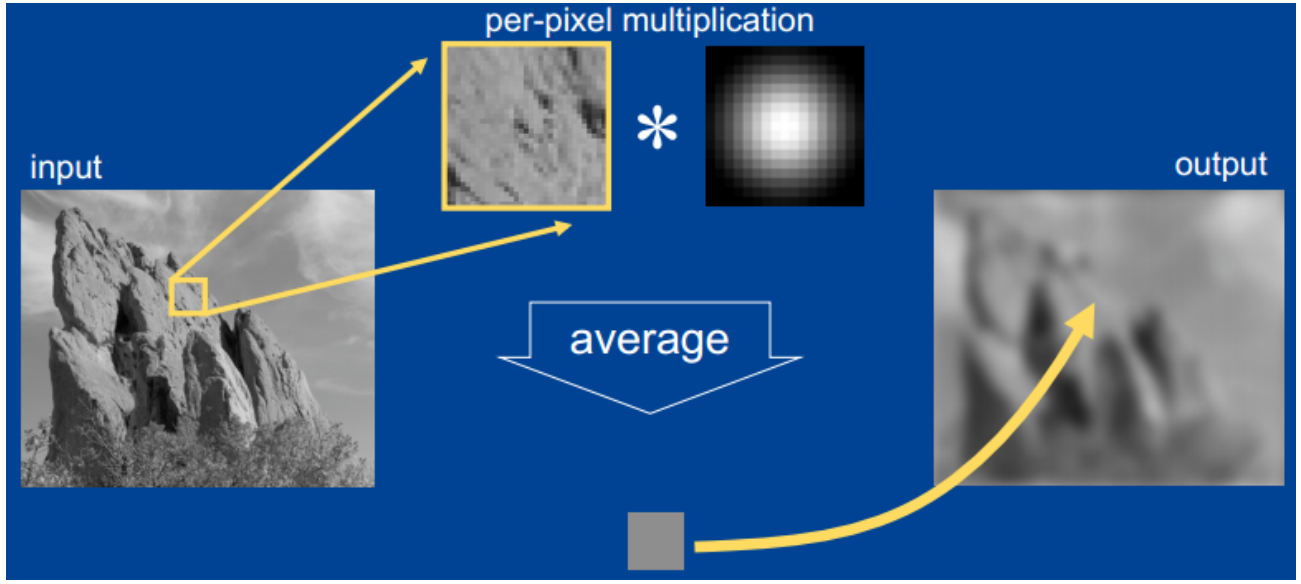


- $G_\sigma(3\sigma) \approx 0$ ➔ 遠くのピクセルは無視できる

- $r := \mathrm{ceil}(3\sigma)$ として $(2r+1) \times (2r+1)$ のステンシル上で重みを前計算



ステンシル

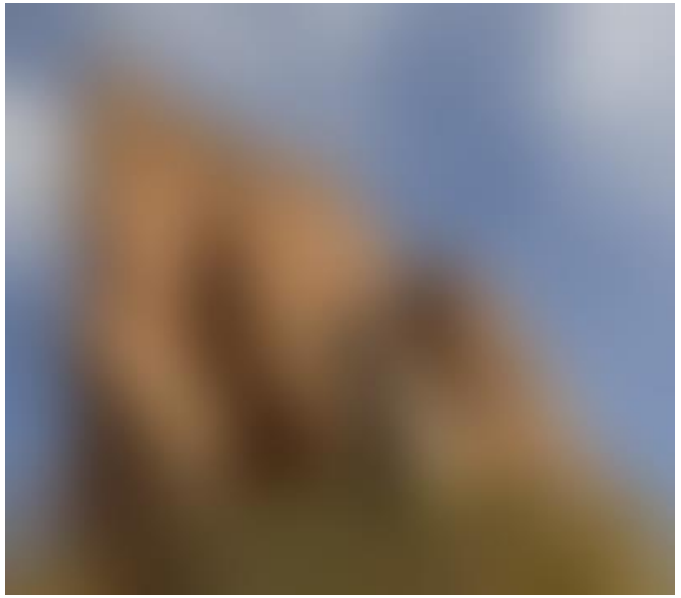


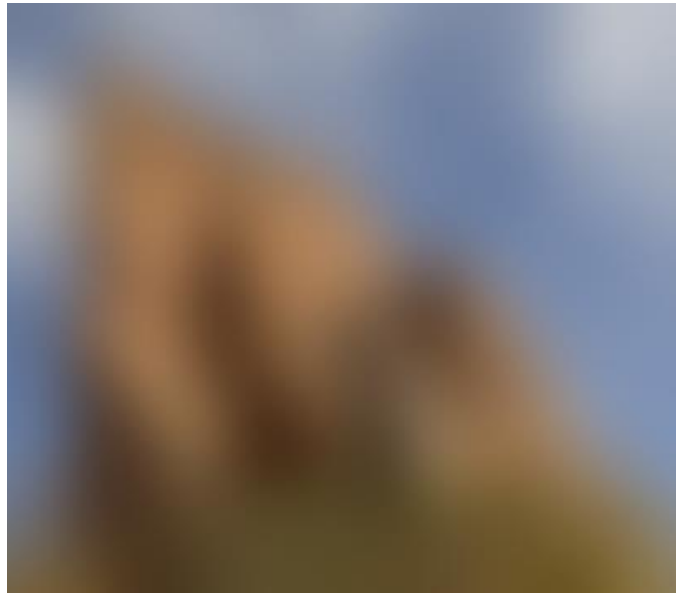
http://people.csail.mit.edu/sparis/bf_course/

Kernel 半径 σ が非常に大きい場合

- そのまま計算すると時間がかかる
- 代替法：downsample ➔ 小さい σ で平滑化 ➔ upsample

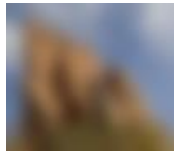
$\sigma = 20$





downsample

$\sigma = 5$



upsample

Detail Extraction & Enhancement

−

smoothed

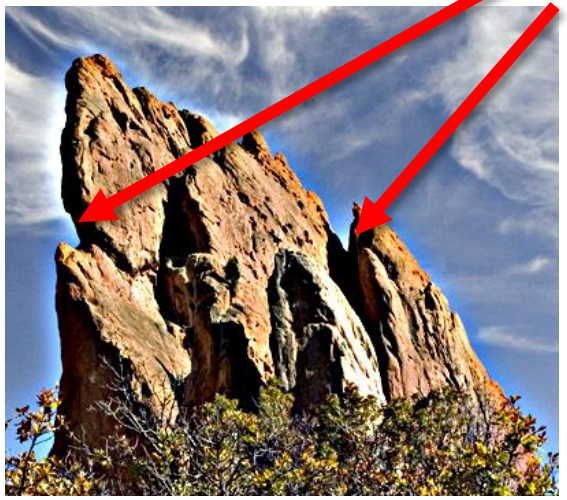
=

detail

+ 3 ×

detail

=



enhanced

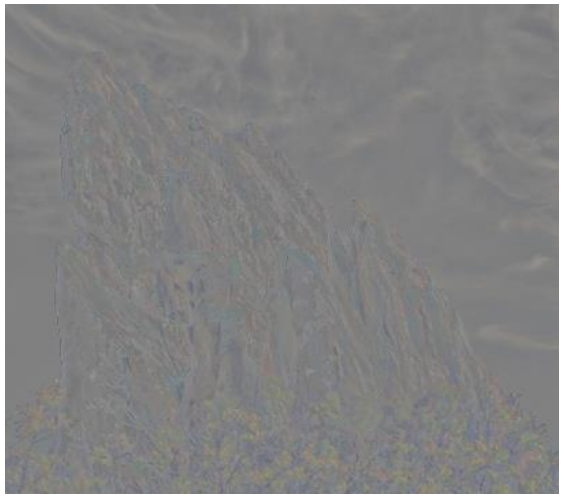
halos!

Edge-aware な画像平滑化を使うと・・・

−

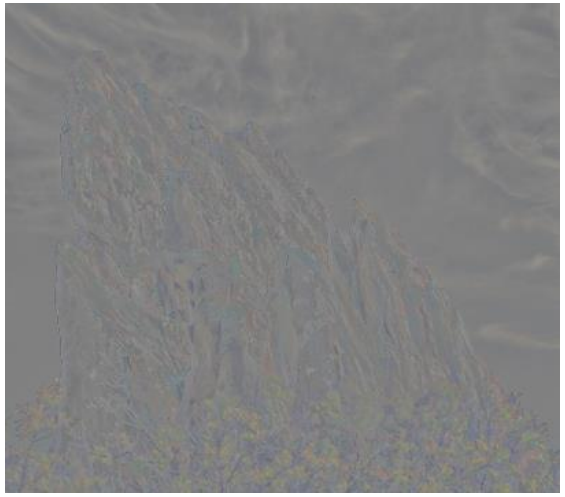
smoothed

=



detail

＋ 3 ×



detail

=

enhanced

Bilateral Filter による edge-aware な平滑化

- 二つのパラメタ
  - σ_s：ピクセルの 位置 に関する平滑化の範囲
  - σ_r：ピクセルの　色　に関する平滑化の範囲

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

すべて $\sigma_s = 10$

元画像

$\sigma_r = 32$

$\sigma_r = 128$

$\sigma_r = 512_{10}$

Bilateral Filter の応用：Stylization

(平滑化した上に輪郭線を描く)

Real-time video abstraction [Winnemöller SIGGRAPH06]

Bilateral Filter の応用：Tone Mapping

- 24bitカラー画像の各成分の範囲：1~255
- 現実世界の光の強さの範囲：$1\sim10^5$
  - **H**igh **D**ynamic **R**ange 画像
  - 露光時間を変えて撮影することで計測可能

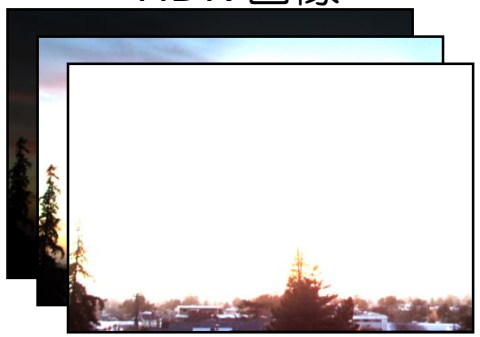
オリジナルの撮影データ
(Tone mapping無し)

Tone mapping処理結果

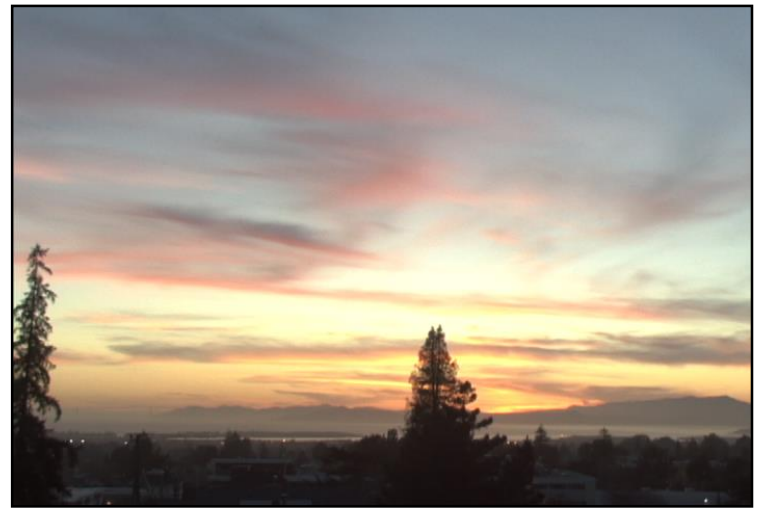
https://en.wikipedia.org/wiki/Tone_mapping

Bilateral Filter の応用：Tone Mapping

HDR 画像



γ補正 ($X \rightarrow X^{\gamma}$)



detail が失われる

Bilateral Filter

＋

γ補正

＋

＝

detail を保つ

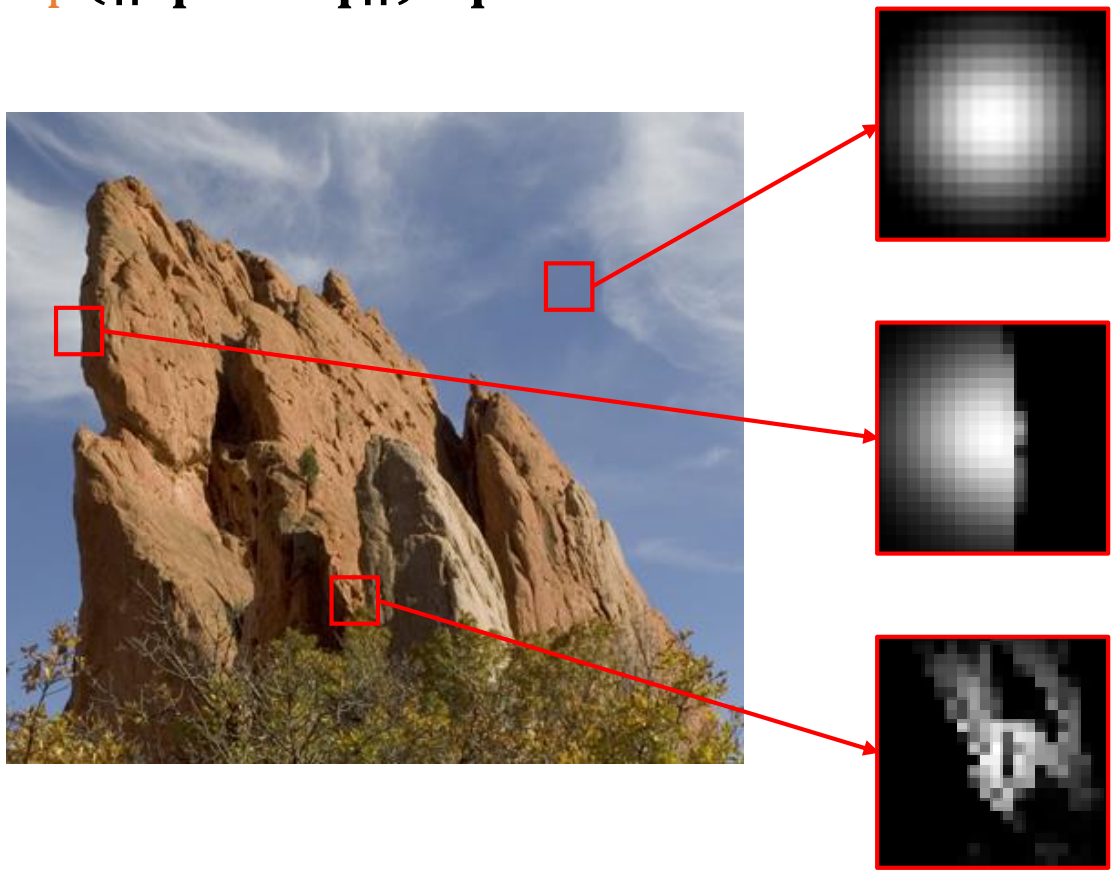
Fast bilateral filtering for the display of high-dynamic-range images [Durand SIGGRAPH02]

Bilateral Filter のナイーブな実装

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

- ピクセル位置 $\mathbf{p} \in \Omega$ ごとにステンシルの再計算が必要
  ➔ 遅い

- (基本課題)



Bilateral Filter に対するもう一つの見方

- ピクセル位置 $\mathbf{p}$ と画素値 $I_{\mathbf{p}}$ から特徴ベクトル $\mathbf{f}_{\mathbf{p}} \coloneqq \left(\frac{\mathbf{p}}{\sigma_{\mathrm{s}}}, \frac{I_{\mathbf{p}}}{\sigma_{\mathrm{r}}}\right)$ を定義

- Bilateral Filter の重みは、特徴ベクトル同士の Euclid 距離を Gaussian Kernel に代入したものに等しい

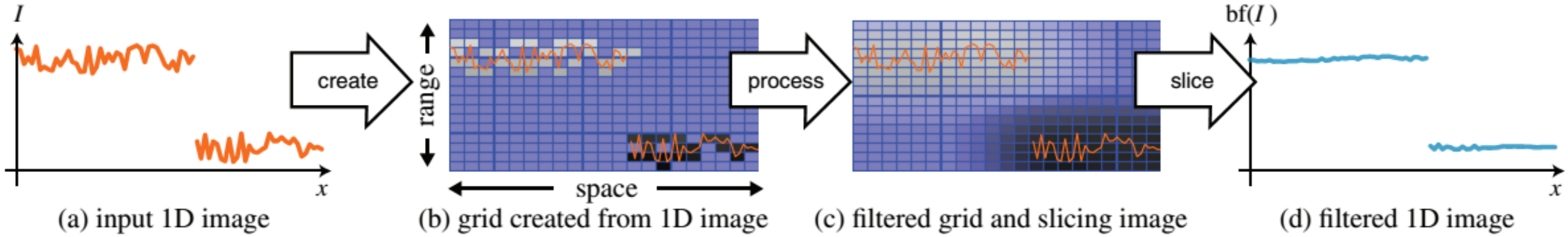
$$
\begin{aligned}
& G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_{\mathrm{r}}}\left(\left\|I_{\mathbf{p}}-I_{\mathbf{q}}\right\|\right) \\
&= \exp\left(-\frac{\|\mathbf{p}-\mathbf{q}\|^2}{2\sigma_{\mathrm{s}}^2}\right) \exp\left(-\frac{\left\|I_{\mathbf{p}}-I_{\mathbf{q}}\right\|^2}{2\sigma_{\mathrm{r}}^2}\right)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

- Bilateral Filter は、特徴空間におけるサンプル集合 $\{\mathbf{f}_{\mathbf{p}}\}$ に対して半径 1 の Gaussian Filter をかけるのと同義
  ➔ 計算が単純化

Bilateral Grid [Paris06; Chen07]

- 3D 特徴ベクトルを (X座標, Y座標, 輝度) として定義し、
  サンプル集合 $\{\mathbf{f_p}\}$ を 3D 配列上にマッピング

- σ_s と σ_r が大きいほど、配列の解像度を低くできる ➔ 計算コスト低減



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]

特徴空間を介した重みマップの生成

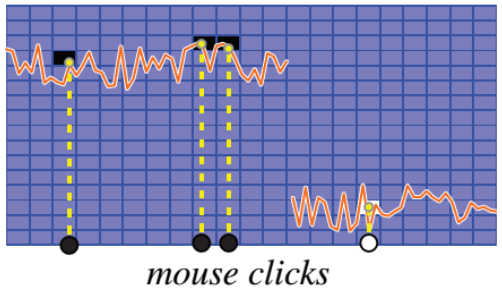
白い scribble ➔ 重み=1 の制約
黒い scribble ➔ 重み=0 の制約

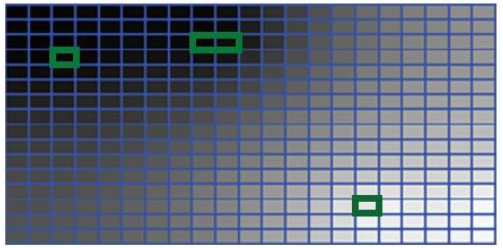
重みマップ

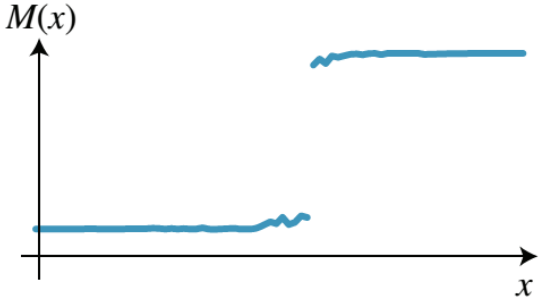
利用例：色味の変更

- 様々な呼ばれ方：Edit Propagation, Matting, Segmentation

- Bilateral Grid 上で
  Laplace 方程式を解く





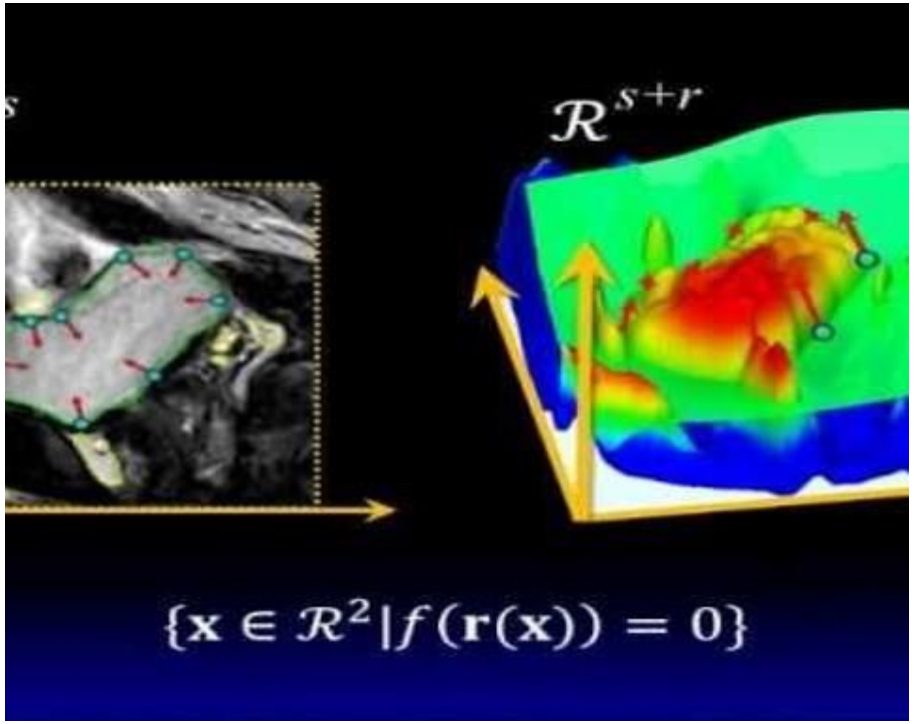


Real-time edge-aware image processing with the bilateral grid [Chen SIGGRAPH07]

特徴空間を介した重みマップの生成

RBF で補間 [Li10]
(目的：画像と動画の編集)

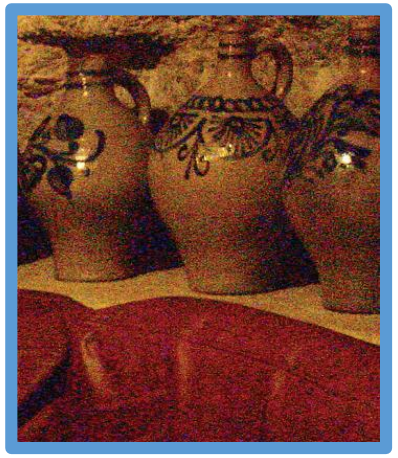
Hermite RBF で補間 [Ijiri13]
(目的：CT volume の領域分割)



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]

Bilateral Filter の拡張：Joint (Cross) Bilateral Filter



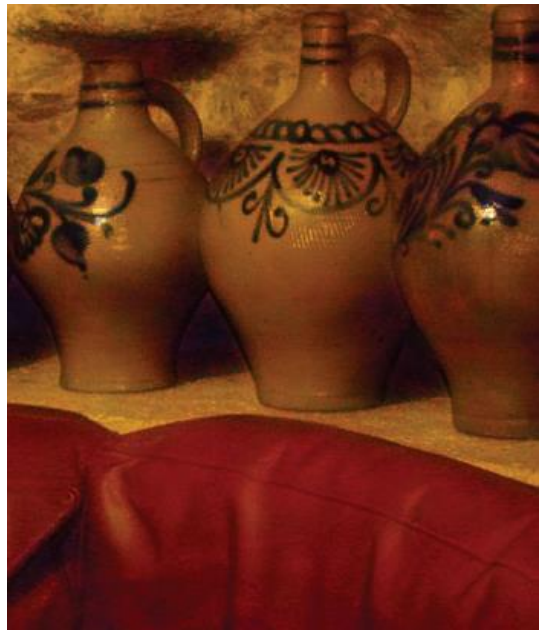
フラッシュ無し写真 **A**
☺ 色味は良い
☹ ノイズが大きい、ボケ気味

フラッシュ有り写真 **F**
☹ 色味は悪い
☺ ノイズが小さい、クッキリ

考え方:
色味に関する情報は**A**から取ってきて、
構造に関する情報は**F**から取ってくる

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

JBF 適用結果



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

Bilateral Filter の拡張：Non-Local Means Filter

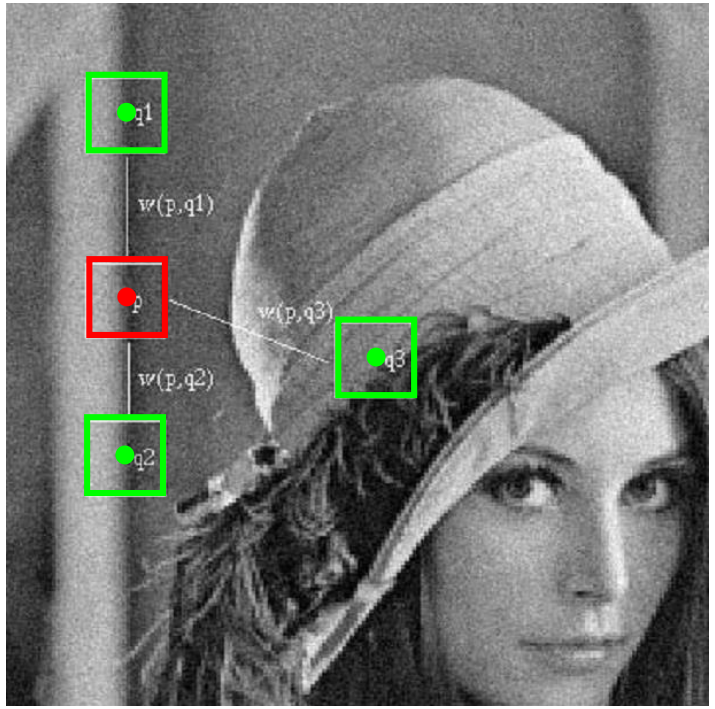
- ピクセル $\mathbf{p}$ を中心とする 7×7 領域の画素値から成る近傍ベクトル $\mathbf{n_p}$ によって、特徴空間を定義

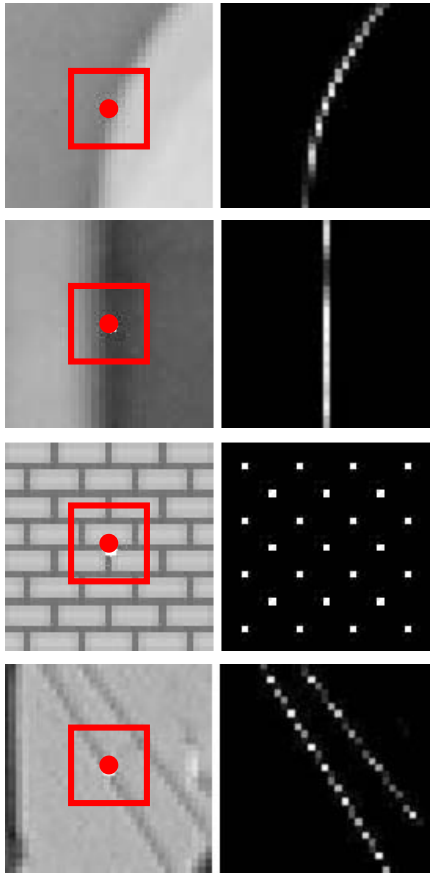
$$\mathrm{NLMF}_\sigma(I)_\mathbf{p} \coloneqq \frac{1}{W_\mathbf{p}} \sum_{\mathbf{q}\in\Omega} G_\sigma(\|\mathbf{n_p} - \mathbf{n_q}\|)\, I_\mathbf{q}$$

Noisy input

Bilateral

NL Means





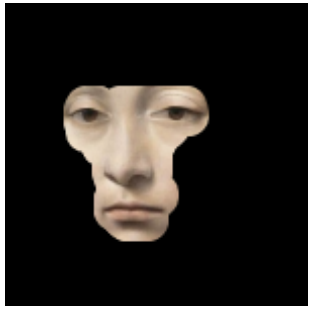
A non local algorithm for image denoising [Buades CVPR05]

本日のトピック

- Edge-aware な画像処理

- Gradient-domain の画像処理

シナリオ：Source 画像を Dest. 画像へ挿入



Source

Dest.

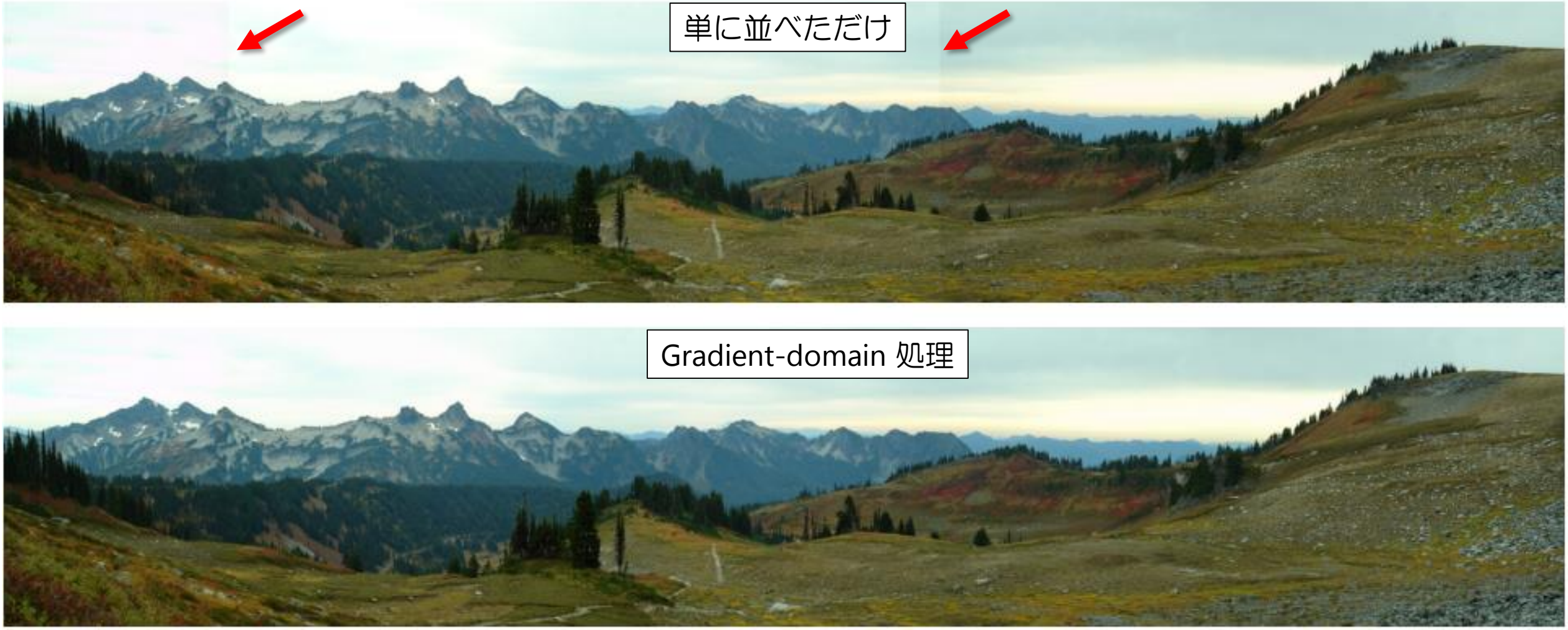
単純な上書き

境界をぼかしてみる

Gradient-domain 処理

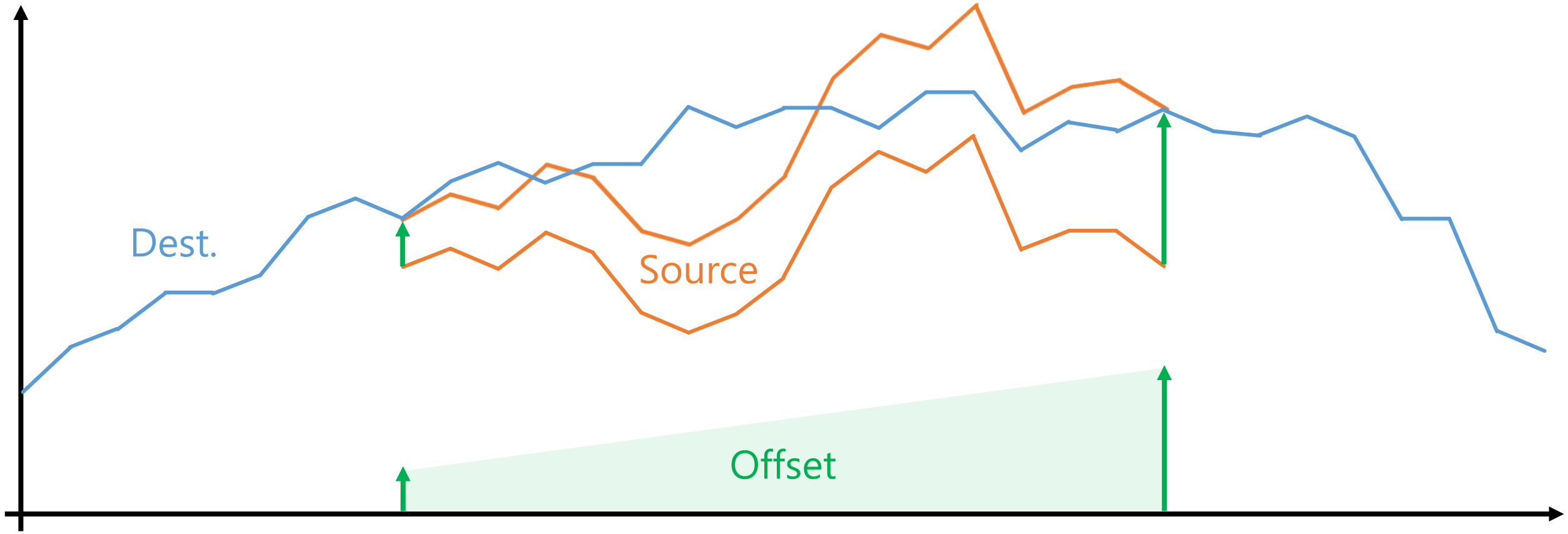
Poisson image editing [Perez SIGGRAPH03]
Gradient Domain Manipulation Techniques in Vision and Graphics [Agrawal ICCV07 Course]

シナリオ：複数写真からパノラマ合成



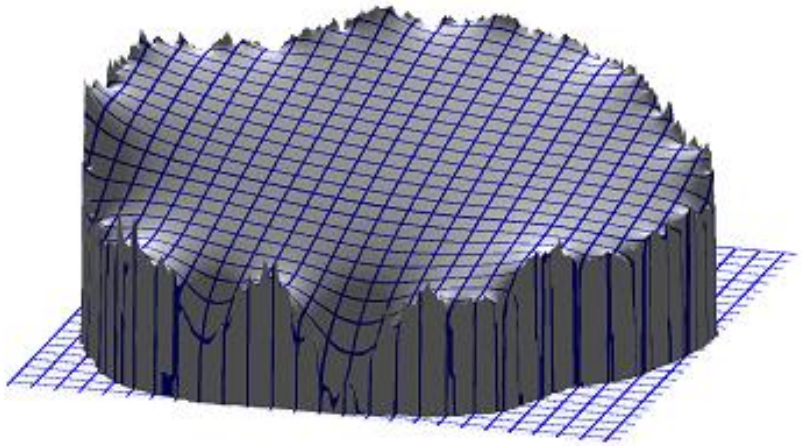
Efficient gradient-domain compositing using quadtrees [Agarwala SIGGRAPH07]

1D grayscale 画像の場合の考察

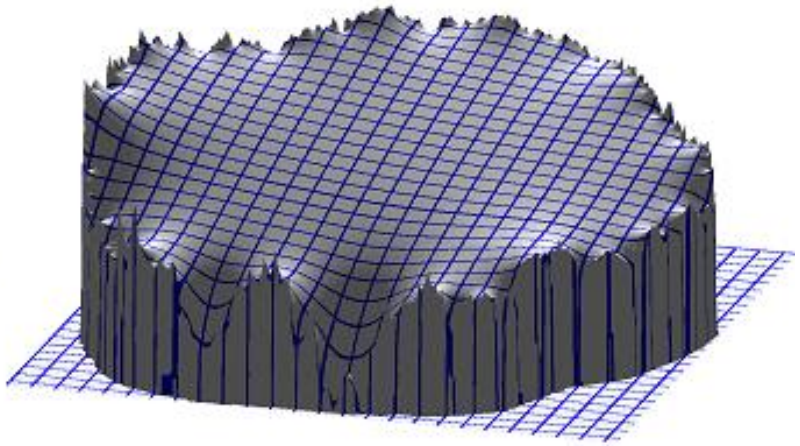


2D の場合：Offset by Laplace Membrane

(a) Source patch



(b) Laplace membrane



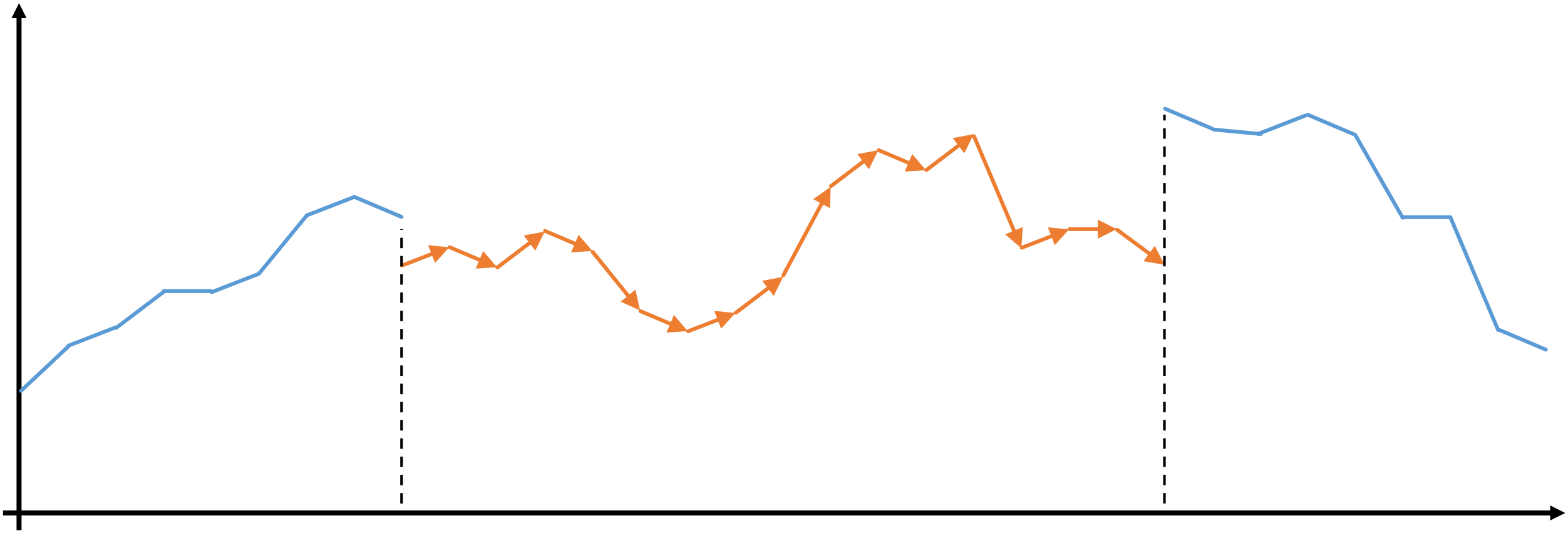
(c) Mean-value membrane

- ディリクレ境界条件の下で Laplace 方程式を解く
- Mean Value Coordinates を用いた高速な近似

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

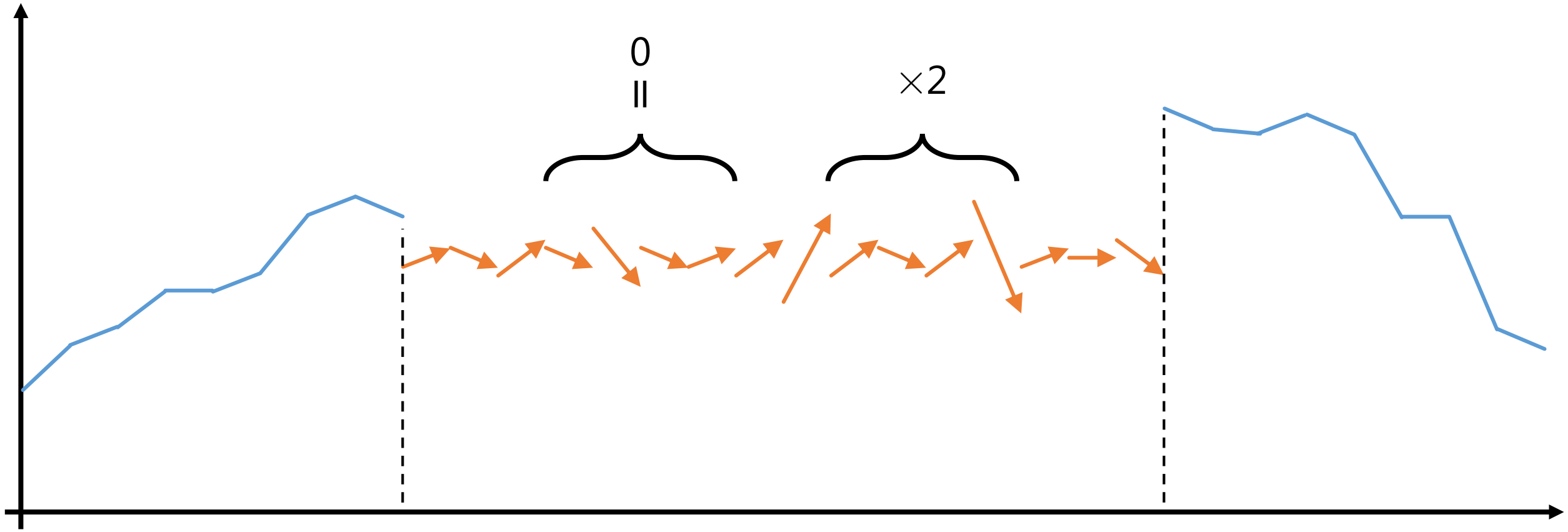
Mean value coordinates [Floater CAGD03]
Coordinates for instant image cloning [Farbman SIGGRAPH09]

単純な cloning 以外の gradient-domain 処理



単純な cloning 以外の gradient-domain 処理

Gradient を好き勝手に操作する！



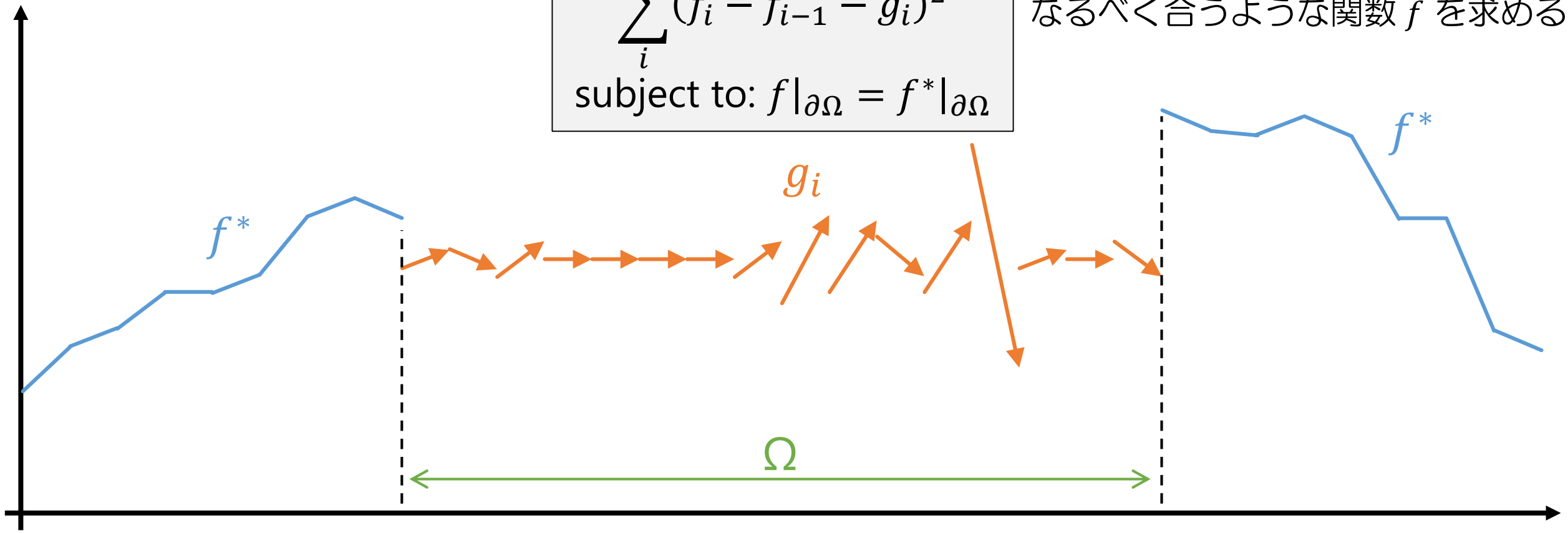
単純な cloning 以外の gradient-domain 処理

Find $\{ f_i \}$ that minimize

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

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

ユーザが与えた目標勾配 $\{ g_i \}$ に
なるべく合うような関数 f を求める



1D の場合

Find $\{ f_i \}$ that minimize

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

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

2D の場合

Find $f(x, y)$ that minimizes

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

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

⇕

Solve Poisson equation:

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

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

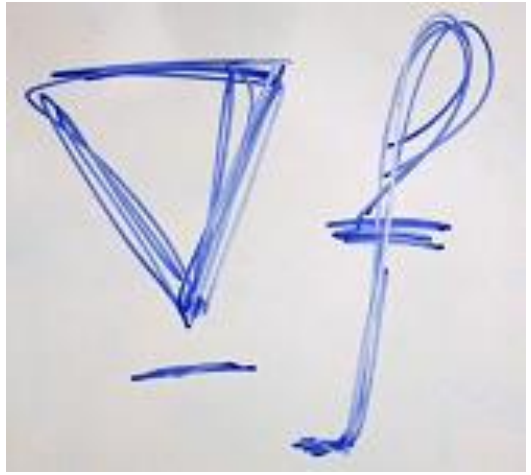
- Gradient-domain 画像処理の基本：

  ユーザが好き勝手に与えた目標勾配
  ベクトル場 $\mathbf{g}$ になるべく合うような
  画像 f を、Poisson 方程式を解いて求める

Target gradient の与え方：Mixing Gradients

- Source 勾配と Dest. 勾配のうち大きい方を使う

  ➔ 平坦な部分は clone されない



Source

Destination

Sourceの勾配を
そのまま使った場合

Source / Dest. 勾配のうち
大きい方を使った場合

Poisson image editing [Perez SIGGRAPH03]

Target gradient の与え方：Mixing Gradients

- Source 勾配と Dest. 勾配のうち大きい方を使う

  ➔ 平坦な部分は clone されない

Source

Destination

Sourceの勾配を
そのまま使った場合

Source / Dest. 勾配のうち
大きい方を使った場合

Poisson image editing [Perez SIGGRAPH03]

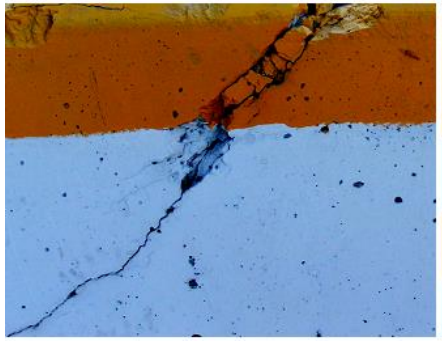
Target gradient の与え方：Edge Brush

- 物体輪郭に沿った勾配をコピーし、ストロークに沿って貼り付け
- GPU 実装の Poisson solver によってリアルタイム動作

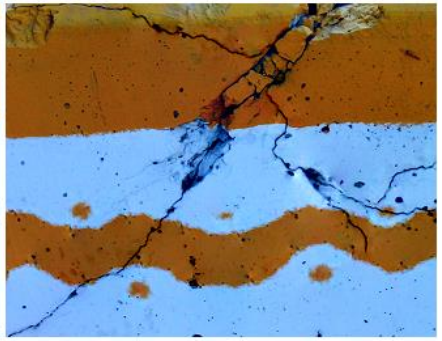
Before

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

Target gradient の与え方：元の gradient を操作

選択範囲内でのみ増幅・減衰
➔ Local Tone Mapping

エッジ検出された場所以外ではゼロにする
➔ Stylization

Poisson image editing [Perez SIGGRAPH03]

おまけ：Gradient-domain の形状処理

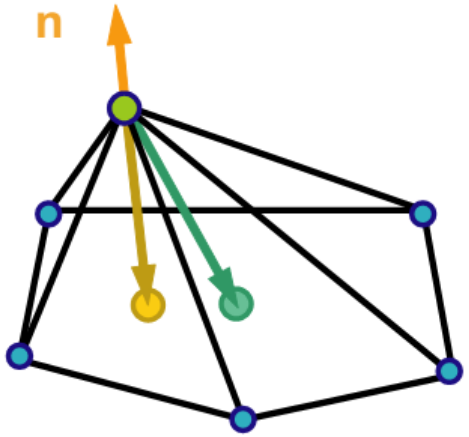
Gradient-domain 形状処理

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

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

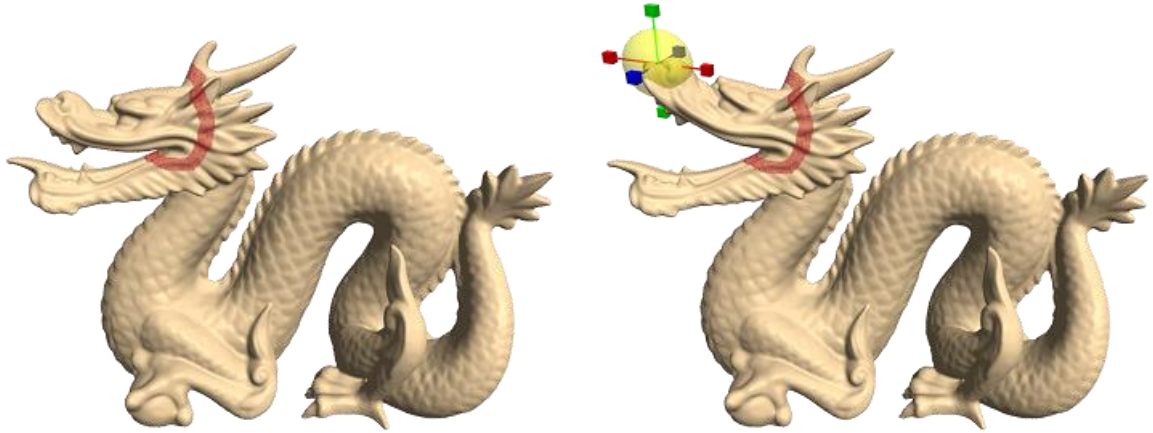
subject to: $\mathbf{v}_c = \mathbf{v}_c^*, c \in I_C$

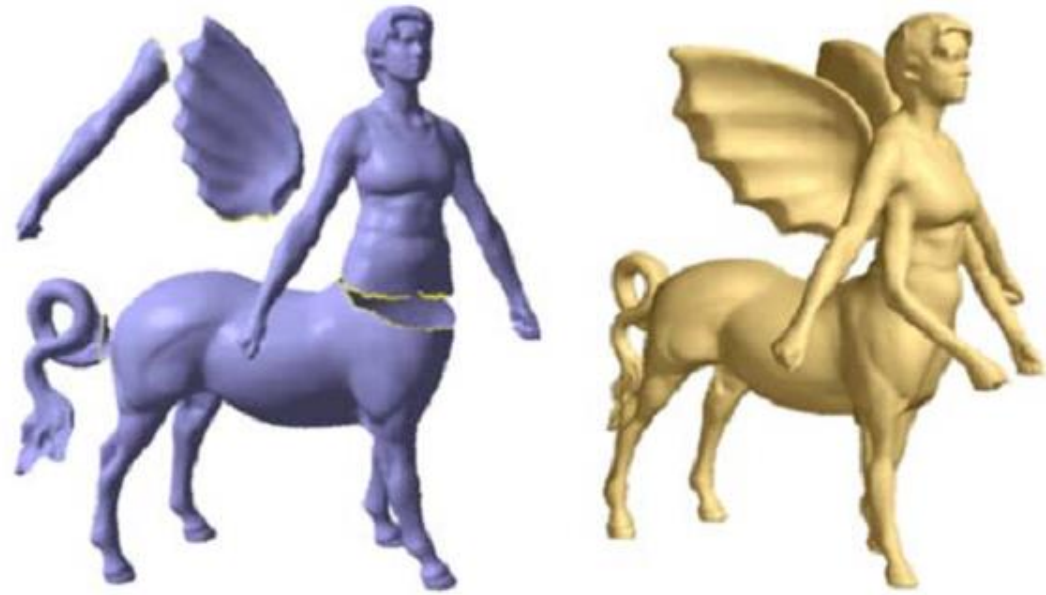
元形状の辺ベクトル
➔ 目標勾配



Poisson 方程式

いくつかの頂点の位置制約
➔ 境界条件

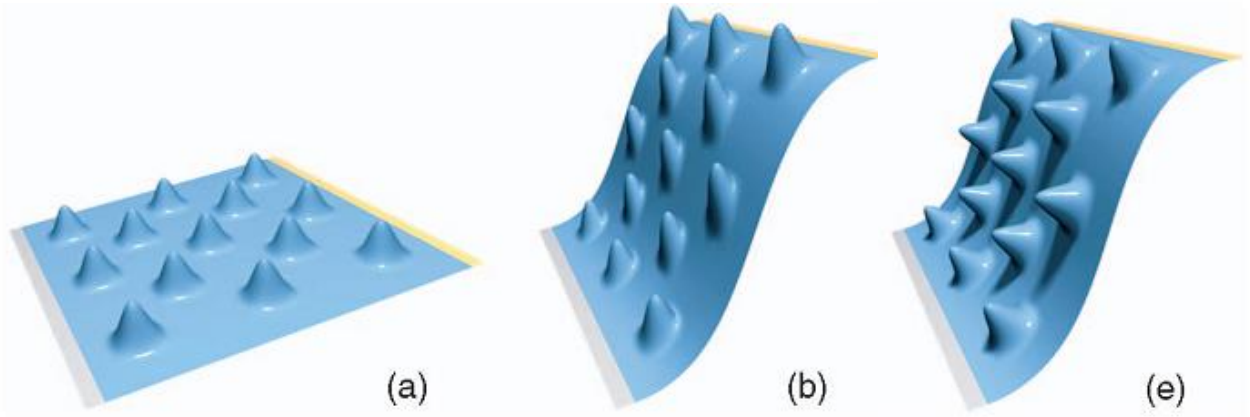




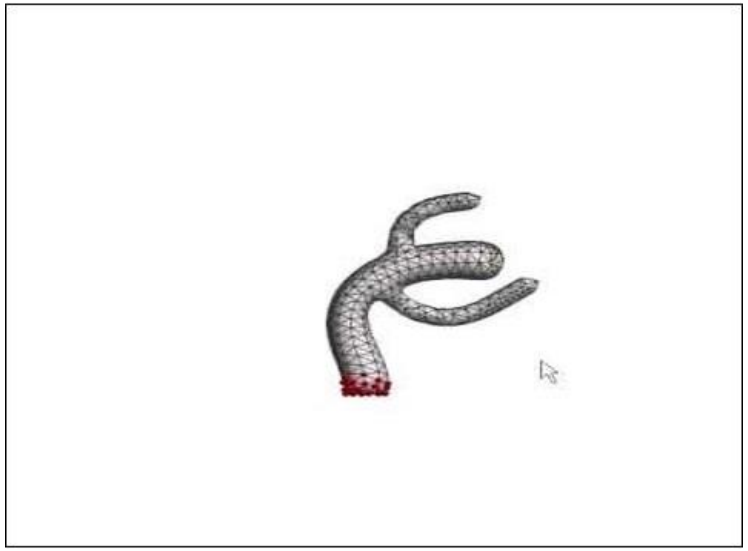
Mesh editing with poisson-based gradient field manipulation [Yu SIGGRAPH04]
Laplacian surface editing [Sorkine SGP04]
Interfaces and algorithms for the creation, modification, and optimization of surface meshes [Nealen PhD07]

変形に伴う局所領域の回転

- 目標勾配も合わせて
  回転させないといけない
  - 非線形で難しい！



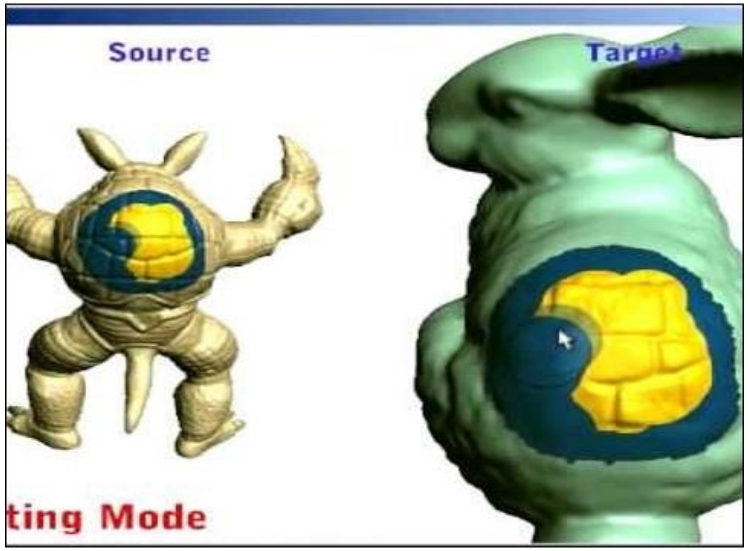
- Local-global 最適化アルゴリズム [Sorkine07]
  - Local step: 頂点座標を固定し、
    SVD で局所領域の回転を計算
  - Global step: 局所領域の回転を固定し、
    Poisson 方程式を解いて頂点座標を更新



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: サーフェスメッシュのためのクローンブラシ

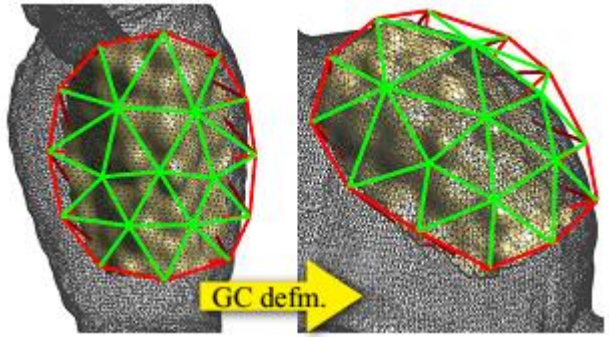


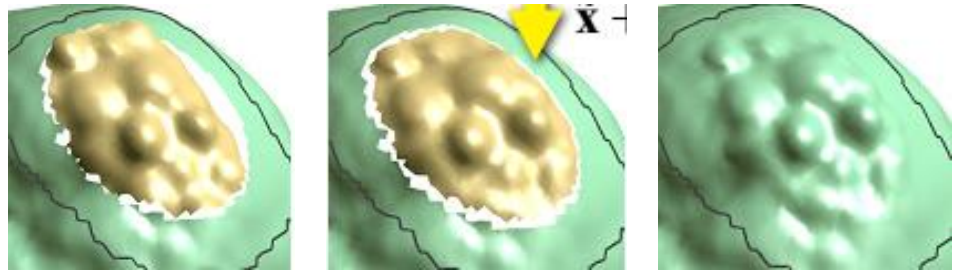
https://www.youtube.com/watch?v=FPsccn_gG8E

- 変形計算を 2 ステップに分解：

1. 局所領域の回転
   ➔ cage-based な方法で高速に計算

2. 正確なオフセット
   ➔ 画像合成用の GPU Poisson ソルバ を流用





GeoBrush: Interactive Mesh Geometry Cloning [Takayama EG11]