BIL 717
Image Processing

Erkut Erdem
Dept. of Computer Engineering
Hacettepe University

LARK based filtering
L0 smoothing

Acknowledgement: The slides are adapted from the ones prepared by P. Milanfar and J. Jia.

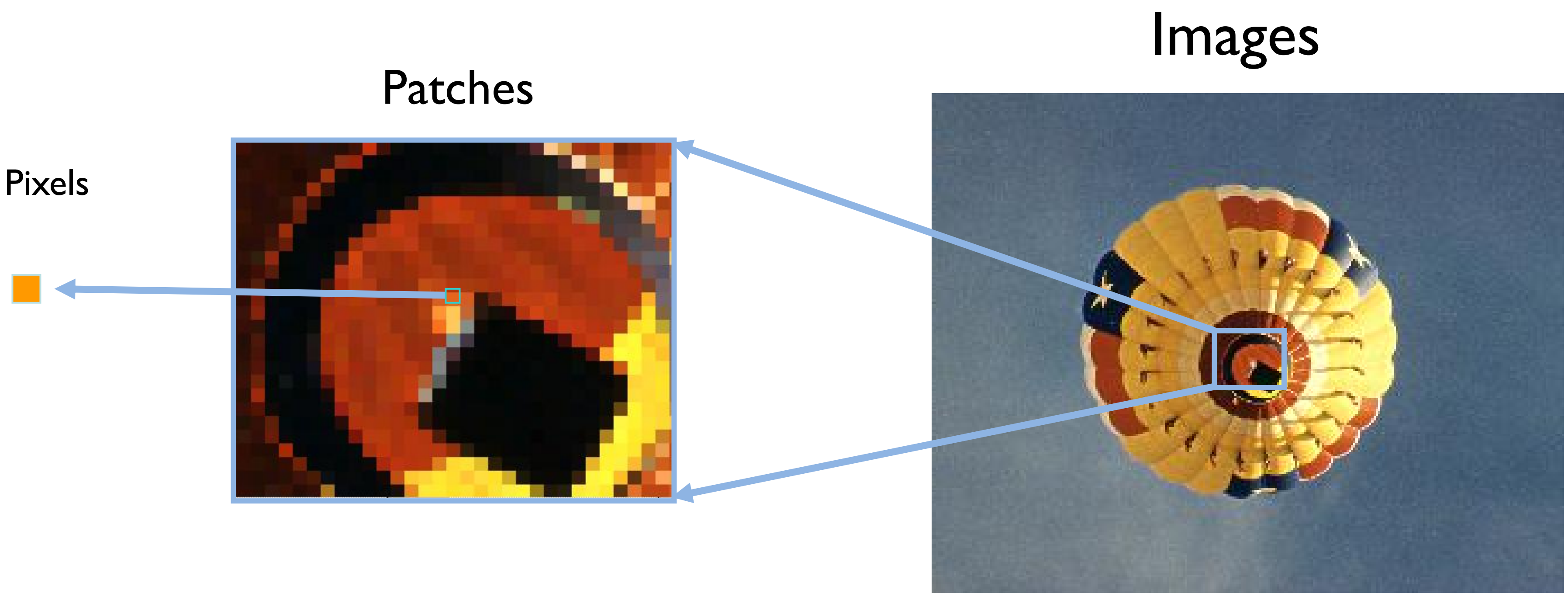
Image Filtering - Review

- We previously discussed:
  - Anisotropic Diffusion PDEs
    [Perona, Malik, ’90, Rudin et al. ‘92, Weickert, ’94]
  - Bilateral filter [Tomasi, Manduchi, ‘98]
  - NL-means filter [Buades, et al. ‘05]

Image Filtering - Today

- Today, we will discuss more recent works:
  - Locally adaptive regression kernels (LARK) based filtering [Takeda et al. ‘07]
  - L0 smoothing [Xu et al. ’11]
  - Method of Karacan, Erdem, Erdem *(work in progress)*

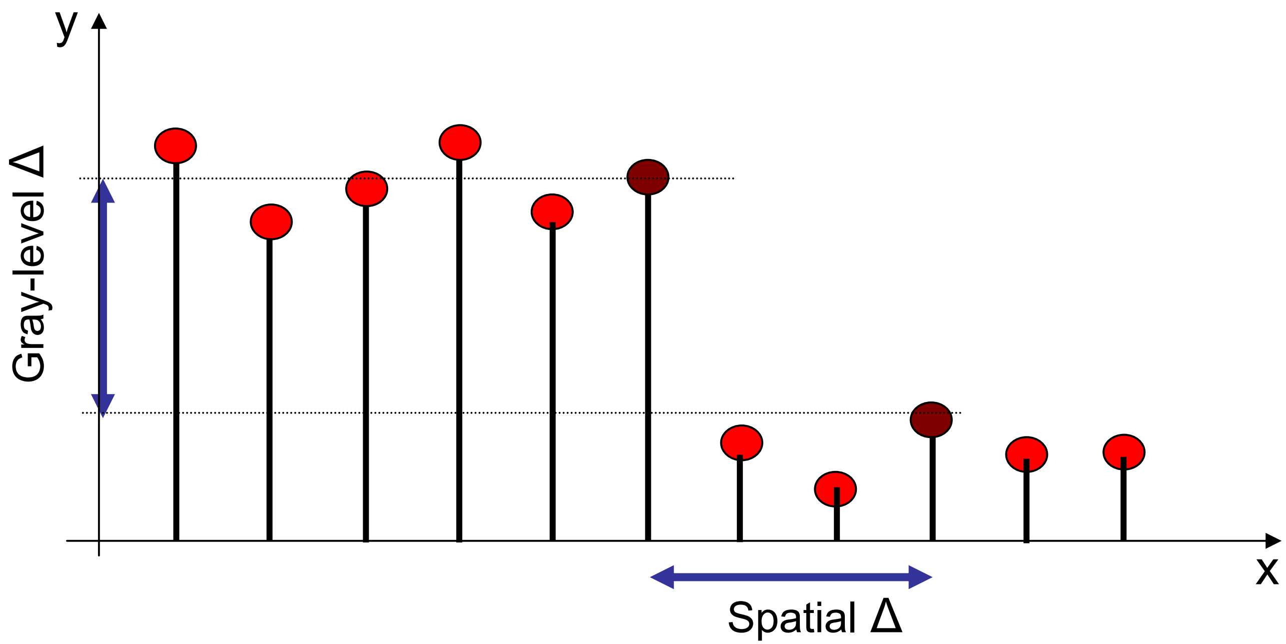
From pixels to patches and to images



Similarities can be defined at different scales..

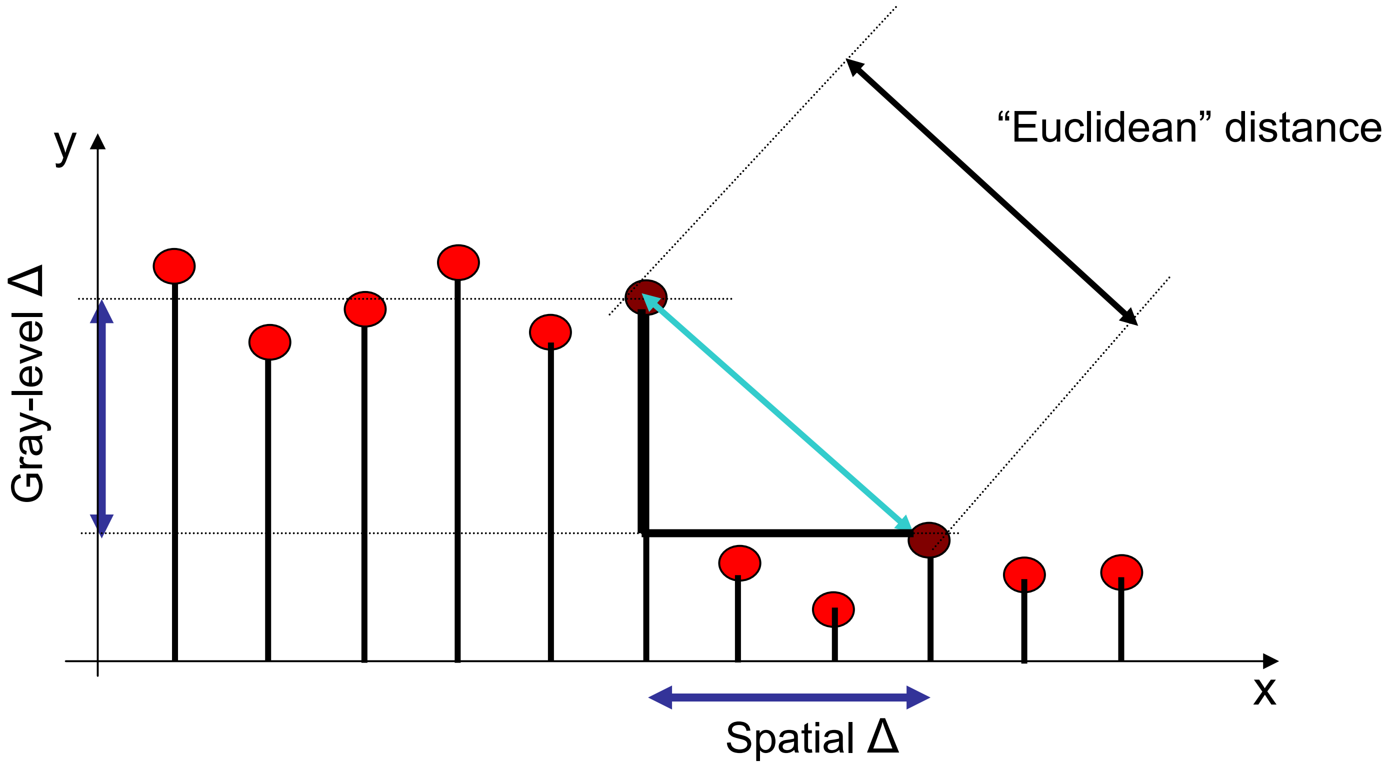
Pixelwise similarity metrics

- To measure the similarity of two pixels, we can consider
  - Spatial distance
  - Gray-level distance



Euclidean metrics

- Natural ways to incorporate the two Δs:
  - Bilateral Kernel [Tomasi, Manduchi, '98] (pixelwise)
  - Non-Local Means Kernel [Buades, et al. '05] (patchwise)

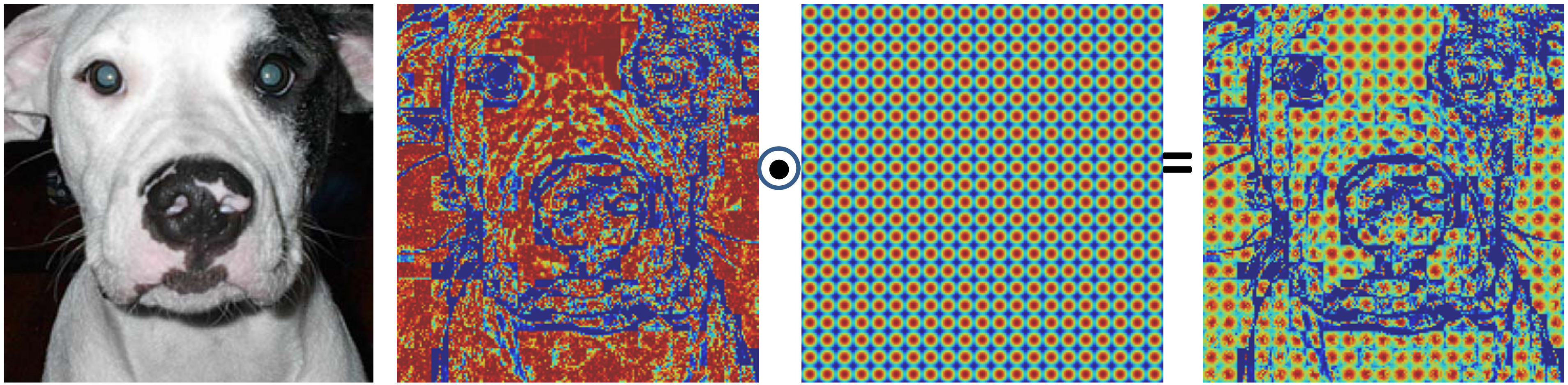


Bilateral Kernel (BL) [Tomasi et al. '98]

Pixels

$$K(\mathbf{x}_l, \mathbf{x}, y_l, y) = \exp\left\{-\frac{\|y_l - y\|^2}{h_r^2} - \frac{\|\mathbf{x}_l - \mathbf{x}\|^2}{h_d^2}\right\}$$

Pixel similarity Spatial similarity

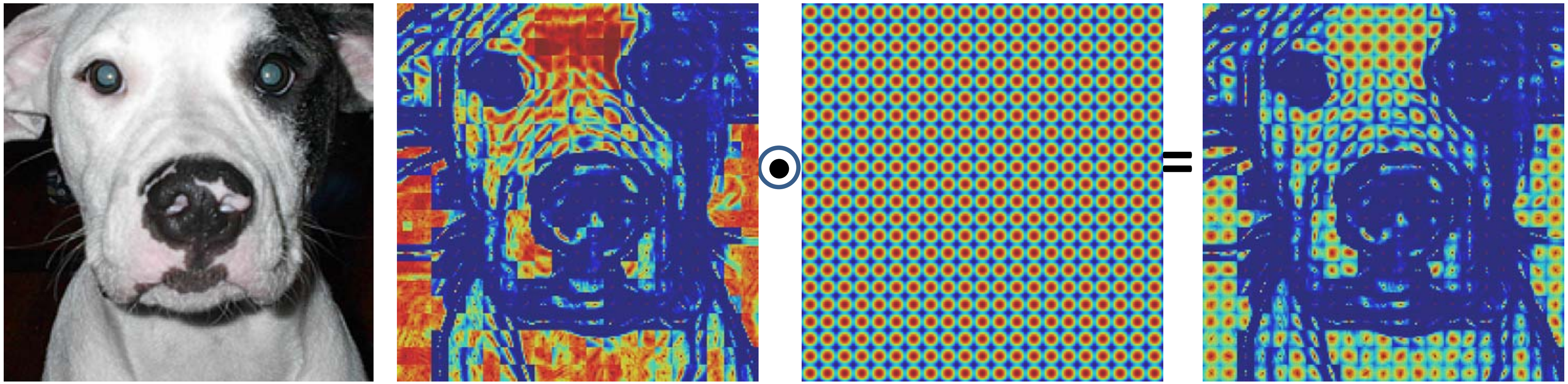


Non-local Means (NLM) [Buades et al. '05]

Patches

$$K(\mathbf{x}_l, \mathbf{x}, \mathbf{y}_l, \mathbf{y}) = \exp\left\{-\frac{\|\mathbf{y}_l - \mathbf{y}\|^2}{h_r^2} - \frac{\|\mathbf{x}_l - \mathbf{x}\|^2}{h_d^2}\right\}$$

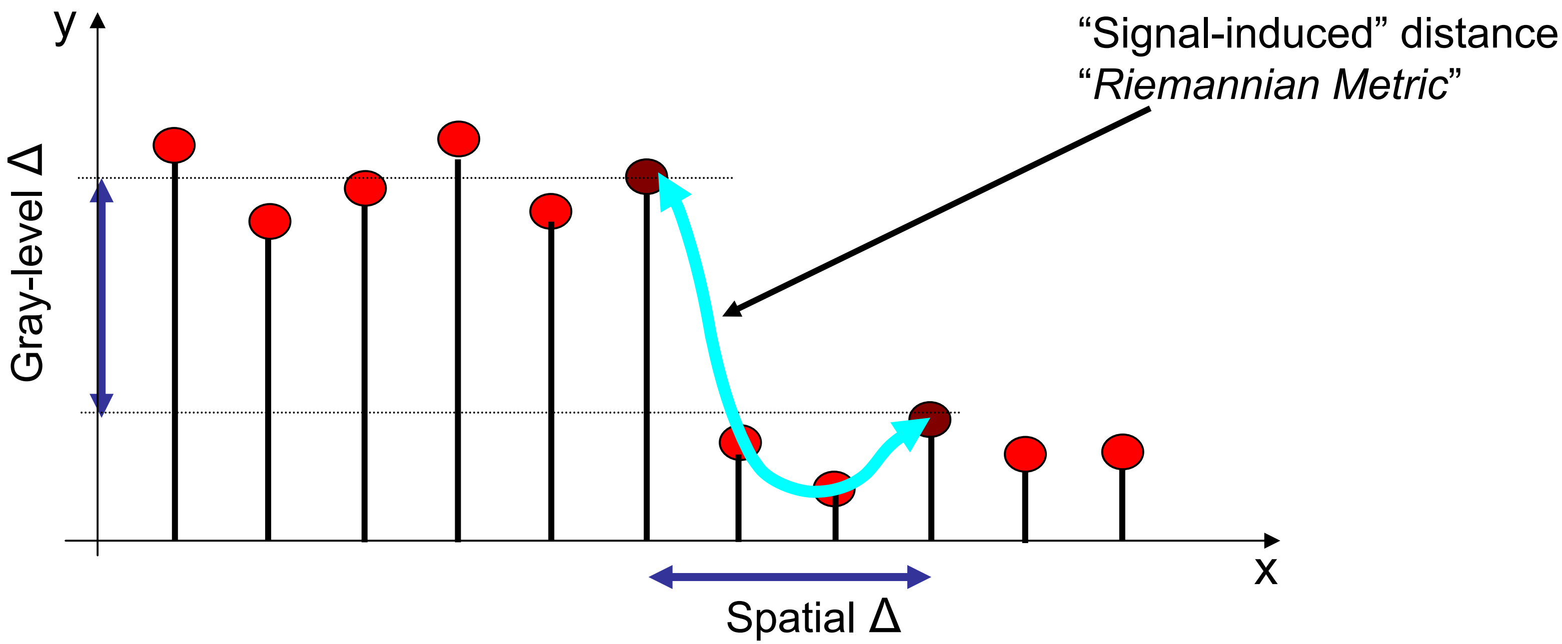
Patch similarity Spatial similarity



Smoothing effect

Beyond Euclidean metrics

- Better similarity measures
- More effective ways to combine the two Δs:
  - LARK Kernel [Takeda, et al. '07]
  - Beltrami Kernel [Sochen, et al.'98]



Non-parametric Kernel Regression

- The data fitting problem

Zero-mean, i.i.d noise (No other assump.)

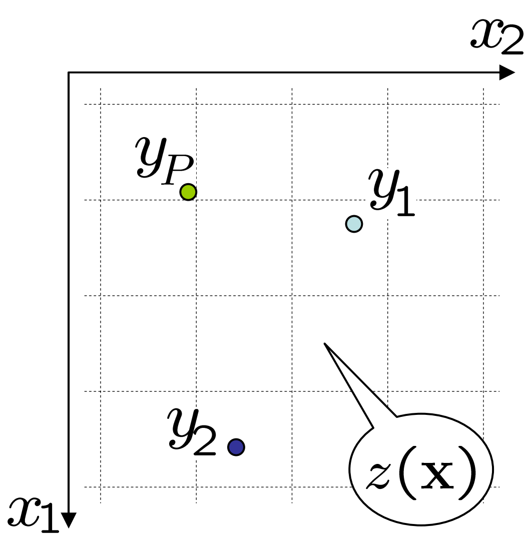
$$y_i = z(\mathbf{x}_i) + \varepsilon_i, \qquad i = 1, 2, \cdots, P$$

Given samples

The sampling position

The number of samples

The regression function



- The particular form of z(x) may remain unspecified for now.

Locality in Kernel Regression

- The data model

$$y_i = z(\mathbf{x}_i) + \varepsilon_i, \quad i = 1, 2, \cdots, P$$

- Local representation (N-term Taylor series expansion)

$$\begin{aligned} z(\mathbf{x}_i) &= \boxed{z(\mathbf{x})} + \boxed{\{\nabla z(\mathbf{x})\}^T} (\mathbf{x}_i - \mathbf{x}) + \boxed{\frac{1}{2!}} (\mathbf{x}_i - \mathbf{x})^T \boxed{\{\mathcal{H} z(\mathbf{x})\}} (\mathbf{x}_i - \mathbf{x}) + \cdots \\ &= \boxed{\beta_0} + \boxed{\boldsymbol{\beta}_1^T} (\mathbf{x}_i - \mathbf{x}) + \boxed{\boldsymbol{\beta}_2^T} \mathrm{vech}\left\{ (\mathbf{x}_i - \mathbf{x})(\mathbf{x}_i - \mathbf{x})^T \right\} + \cdots, \end{aligned}$$

Unknowns

- Note that with a polynomial basis, we only need to estimate the first unknown β_0

Finding the unknowns via optimization

- We have a local representation with respect to each sample:

$$
\begin{aligned}
y_1 &= \beta_0 + \boldsymbol{\beta}_1^T(\mathbf{x}_1 - \mathbf{x}) + \boldsymbol{\beta}_2^T \,\text{vech}\left\{(\mathbf{x}_1 - \mathbf{x})(\mathbf{x}_1 - \mathbf{x})^T\right\} + \cdots + \varepsilon_1,\\
y_2 &= \beta_0 + \boldsymbol{\beta}_1^T(\mathbf{x}_2 - \mathbf{x}) + \boldsymbol{\beta}_2^T \,\text{vech}\left\{(\mathbf{x}_2 - \mathbf{x})(\mathbf{x}_2 - \mathbf{x})^T\right\} + \cdots + \varepsilon_2,\\
&\vdots \qquad\qquad\qquad\qquad \vdots\\
y_P &= \beta_0 + \boldsymbol{\beta}_1^T(\mathbf{x}_P - \mathbf{x}) + \boldsymbol{\beta}_2^T \,\text{vech}\left\{(\mathbf{x}_P - \mathbf{x})(\mathbf{x}_P - \mathbf{x})^T\right\} + \cdots + \varepsilon_P,
\end{aligned}
$$

- Optimation

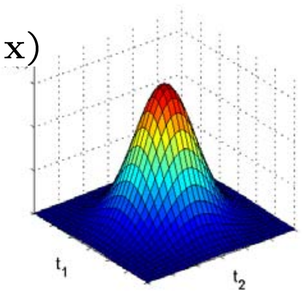
The regression order

$$
\min_{\{\boldsymbol{\beta}_n\}_{n=0}^{N}} \sum_{i=1}^{P} \Big[y_i - \overbrace{\beta_0 - \boldsymbol{\beta}_1^T(\mathbf{x}_i - \mathbf{x}) - \boldsymbol{\beta}_2^T \text{vech}\left\{(\mathbf{x}_i - \mathbf{x})(\mathbf{x}_i - \mathbf{x})^T\right\} - \cdots}^{\text{N+1 terms}} \Big]^2 \mathsf{K}(\mathbf{x}_i - \mathbf{x})
$$

This term give the estimated pixel value z(x).

The choice of the kernel function is open, e.g. Gaussian.

$\mathsf{K}(\mathbf{x}_i - \mathbf{x})$



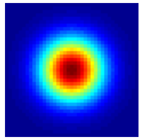
$$
\boxed{\hat{z}(\mathbf{x}) = \sum_{i=1}^{P} W_i(\mathbf{x}, K, h, N)\, y_i}
$$

Defining Data-Adaptive Kernels

- *Classic* Kernel: Locally Linear Filter:

$$\hat{z}(\mathbf{x}) = \hat{\beta}_0 = \sum_i W(\mathbf{x}_i, \mathbf{x}, N)\, y_i$$

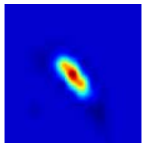
Uses distance $x-x_i$



- *Data-Adaptive* Kernel: Locally Non-Linear Filter:

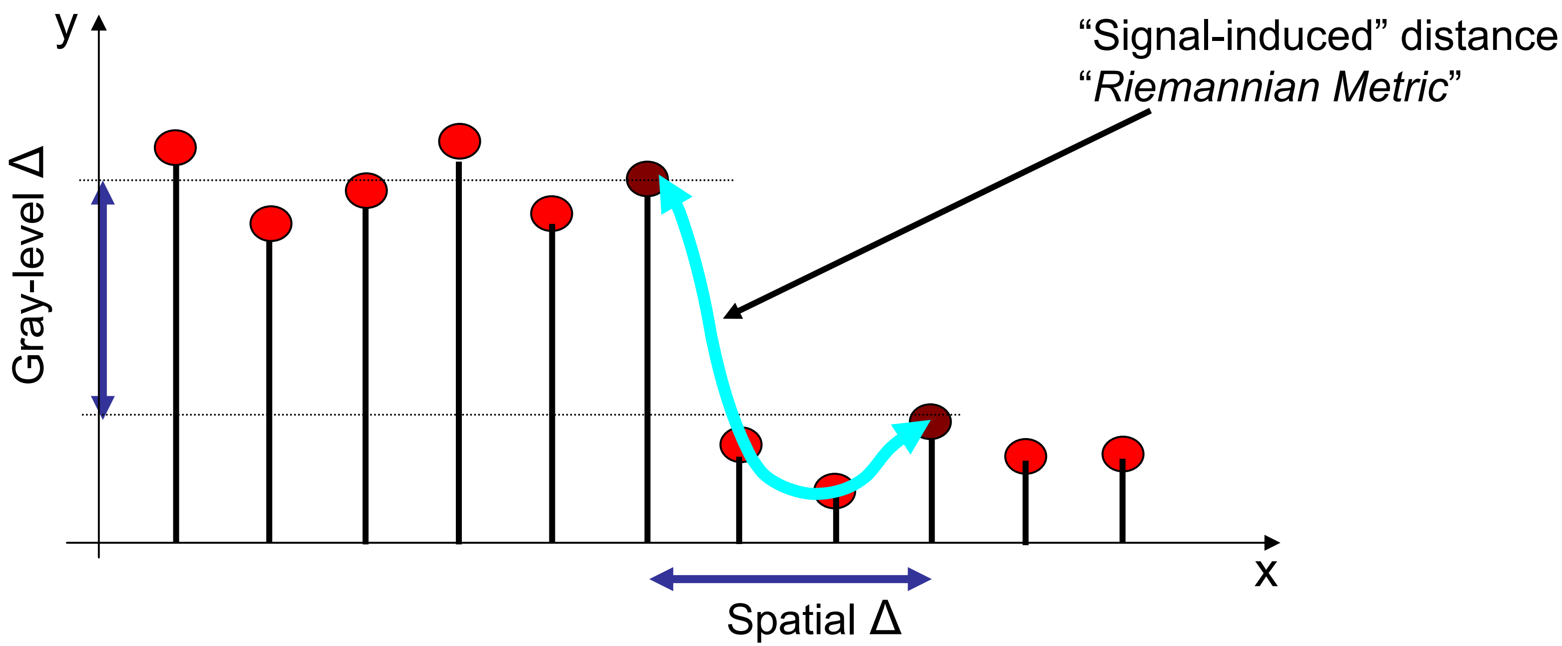
$$\hat{z}(\mathbf{x}) = \hat{\beta}_0 = \sum_i W(\mathbf{x}_i, \mathbf{x}, y_i, y, N)\, y_i$$

Uses $x-x_i$ and $y-y_i$



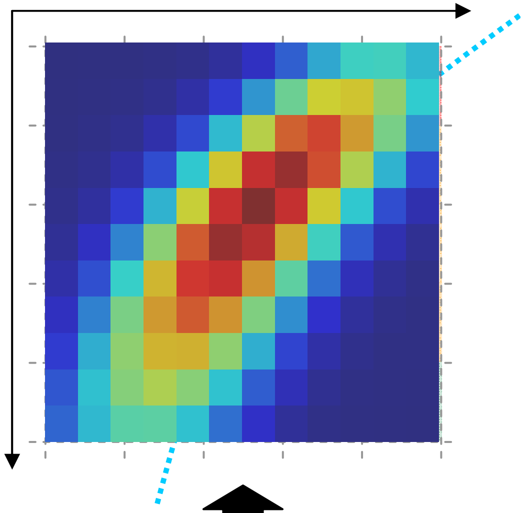
Recall - Beyond Euclidean metrics

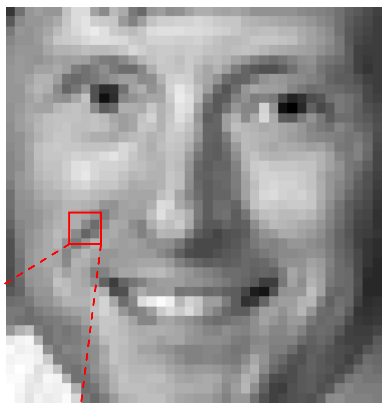
- Better similarity measures
- More effective ways to combine the two Δs:
  - LARK Kernel [Takeda, et al. '07]
  - Beltrami Kernel [Sochen, et al.'98]

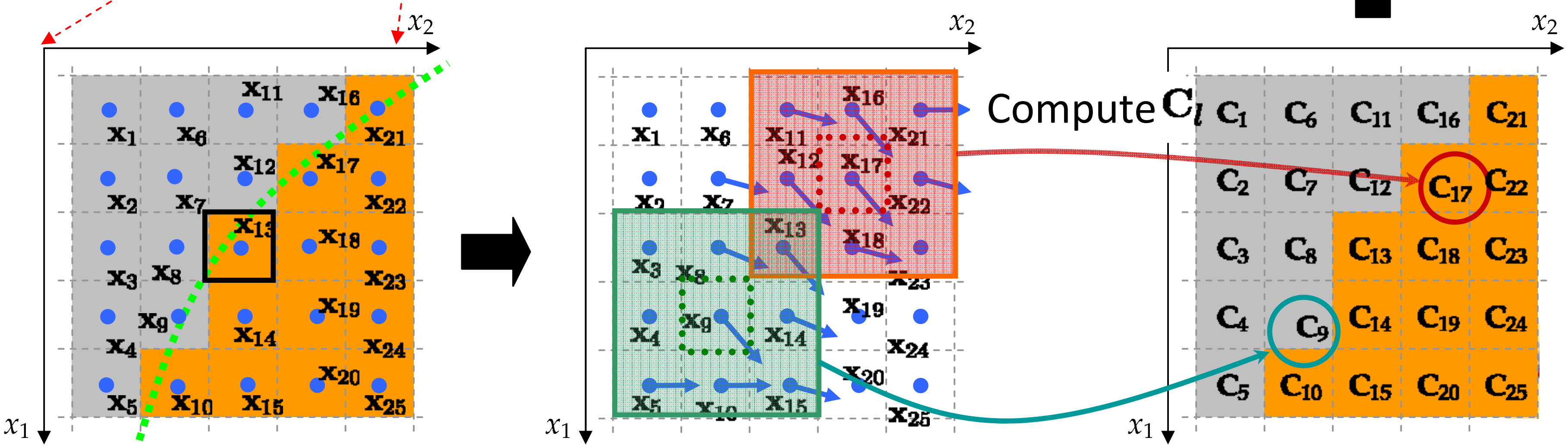


LARK Kernels

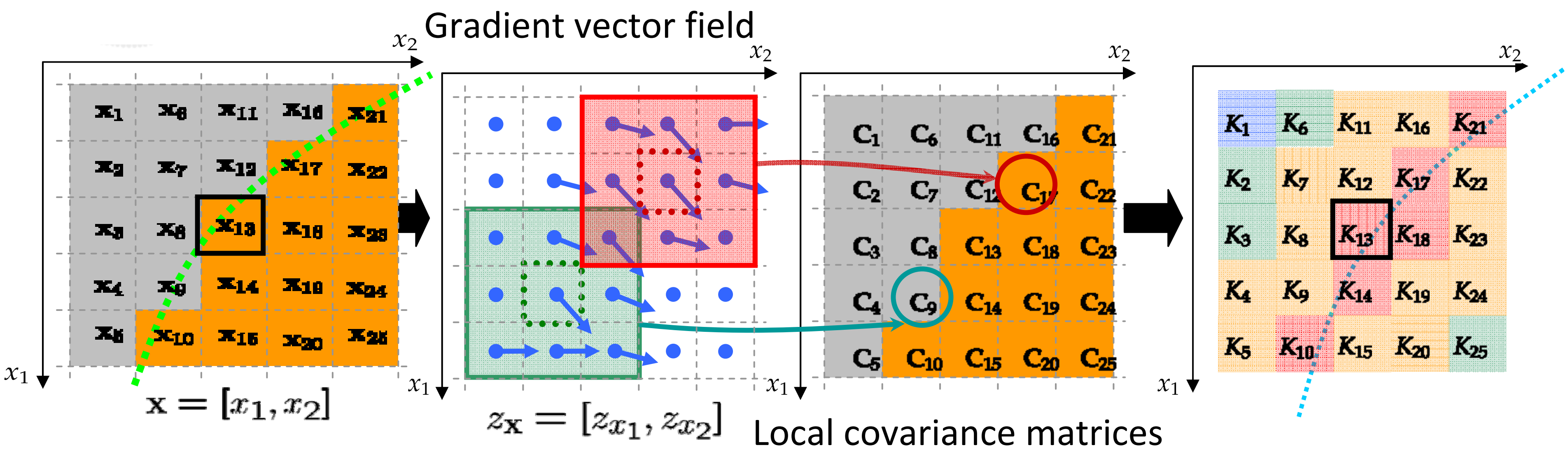
$$K(\mathbf{C}_l, \mathbf{x}_l, \mathbf{x}) = \exp\left\{-(\mathbf{x}_l - \mathbf{x})'\mathbf{C}_l(\mathbf{x}_l - \mathbf{x})\right\}$$







LARK Kernels



Locally Adaptive Regression Kernel: LARK

$$K(\mathbf{C}_l, \mathbf{x}_l, \mathbf{x}) = \exp\left\{-(\mathbf{x}_l - \mathbf{x})'\mathbf{C}_l(\mathbf{x}_l - \mathbf{x})\right\}$$

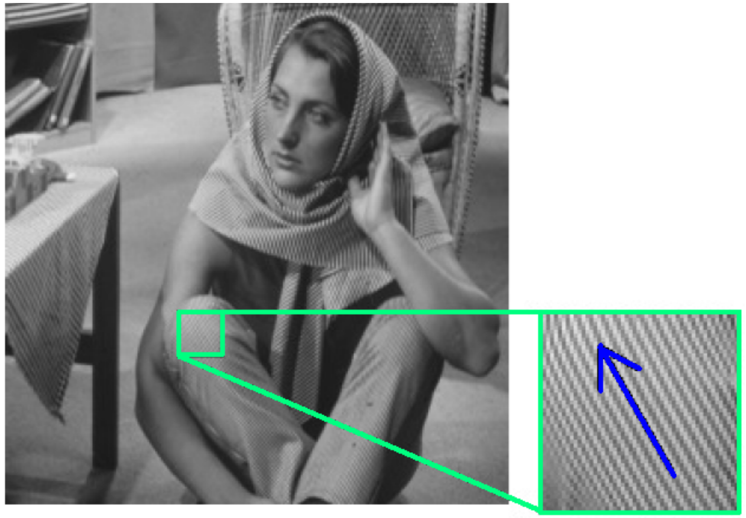
"Structure tensor"

$$\mathbf{C}_l = \sum_{k \in \Omega_l} \begin{bmatrix} z_{x_1}^2(\mathbf{x}_k) & z_{x_1}(\mathbf{x}_k) z_{x_2}(\mathbf{x}_k) \\ z_{x_1}(\mathbf{x}_k) z_{x_2}(\mathbf{x}_k) & z_{x_2}^2(\mathbf{x}_k) \end{bmatrix}$$

Gradient Covariance Matrix and Local Geometry

Gradient matrix over a local patch:

$$\mathbf{C}_l = \sum_{k \in \Omega_l} \begin{bmatrix} z_{x_1}^2(\mathbf{x}_k) & z_{x_1}(\mathbf{x}_k) z_{x_2}(\mathbf{x}_k) \\ z_{x_1}(\mathbf{x}_k) z_{x_2}(\mathbf{x}_k) & z_{x_2}^2(\mathbf{x}_k) \end{bmatrix}$$



$$\mathbf{C}_l = \mathbf{G}^{\mathrm{T}}\mathbf{G}$$

$$\mathbf{G} = \mathbf{U}\mathbf{S}\mathbf{V}^T = \mathbf{U} \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix}^T$$

Capturing locally dominant orientations

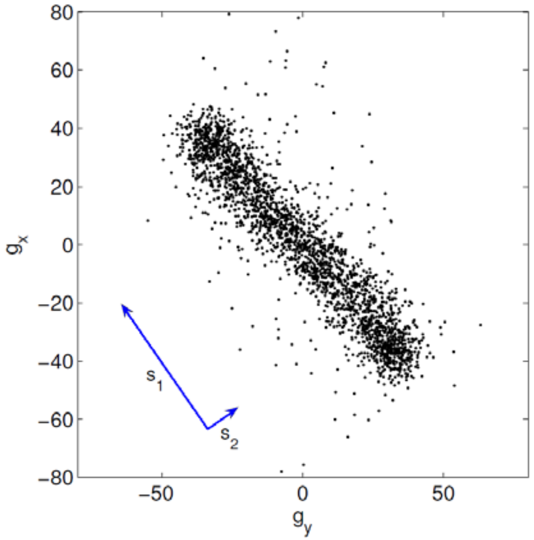
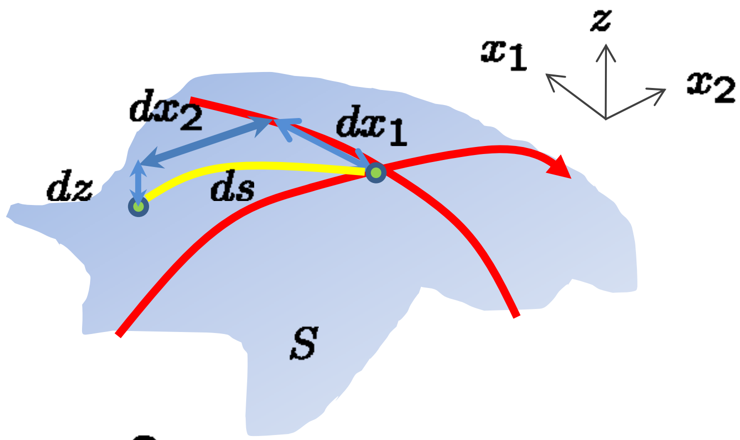


Image as a Surface Embedded in the Euclidean 3-space



$$S(x_1, x_2) = \{x_1, x_2, z(x_1, x_2)\} \in \mathbb{R}^3$$

Arclength on the surface

$$\begin{aligned} ds^2 &= dx_1^2 + dx_2^2 + dz^2 \\ &= dx_1^2 + dx_2^2 + (z_{x_1} dx_1 + z_{x_2} dx_2)^2 \\ &= (1 + z_{x_1}^2) dx_1^2 + 2 z_{x_1} z_{x_2} dx_1 dx_2 + (1 + z_{x_2}^2) dx_2^2 \end{aligned}$$

Chain rule

$$= (dx_1 \quad dx_2) \begin{pmatrix} 1 + z_{x_1}^2 & z_{x_1} z_{x_2} \\ z_{x_1} z_{x_2} & 1 + z_{x_2}^2 \end{pmatrix} \begin{pmatrix} dx_1 \\ dx_2 \end{pmatrix}$$

$$\Rightarrow (\mathbf{x}_l - \mathbf{x})^T (\mathbf{C}_l + \mathbf{I})(\mathbf{x}_l - \mathbf{x})$$

Riemannian metric

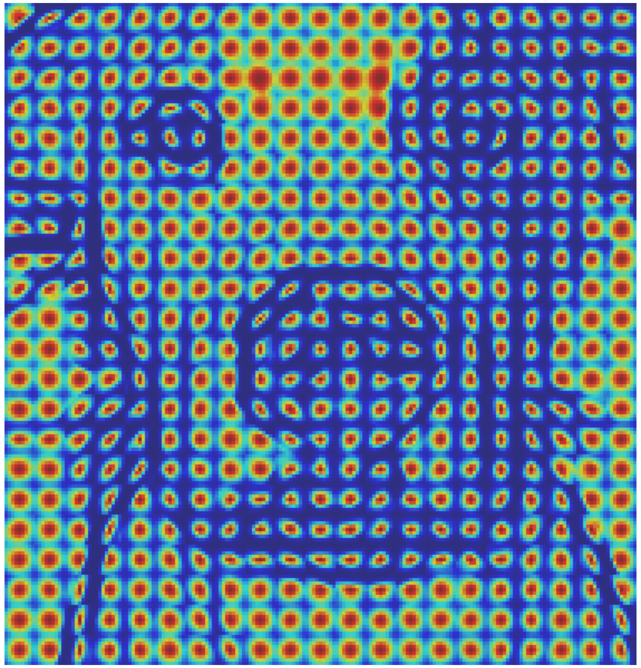
Regularization term

$$K(\mathbf{C}_l, \mathbf{x}_l, \mathbf{x}) = \exp\left\{-(\mathbf{x}_l - \mathbf{x})' \mathbf{C}_l (\mathbf{x}_l - \mathbf{x})\right\}$$

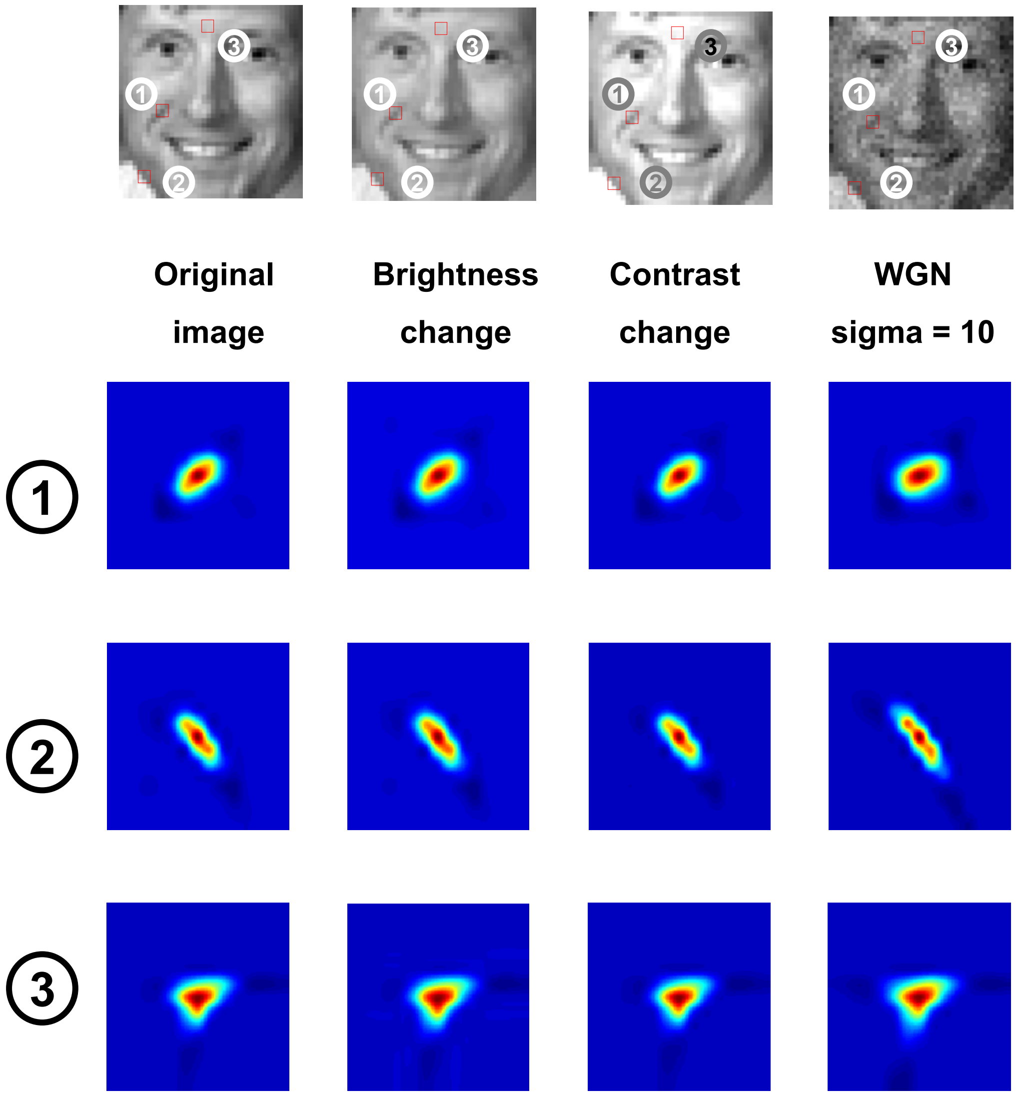
(Dense) LARK Kernels as Visual Descriptors [Seo and Milanfar '10]

$$K(\mathbf{C}_l, \mathbf{x}_l, \mathbf{x}) = \exp\left\{-\underbrace{(\mathbf{x}_l - \mathbf{x})'\mathbf{C}_l(\mathbf{x}_l - \mathbf{x})}\right\}$$

Measure the similarity of pixels using the metric implied by the local structure of the image

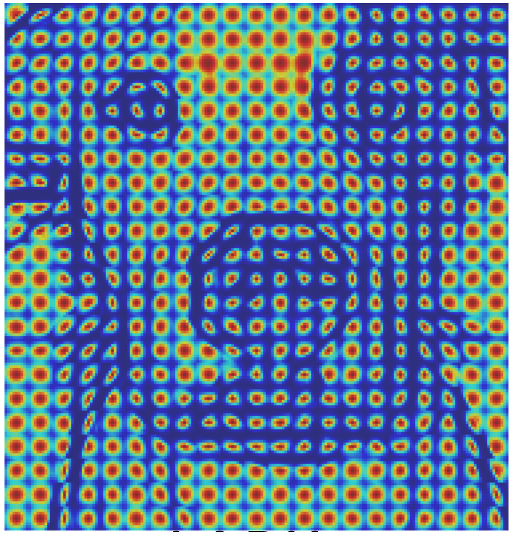


Robustness of LARK Descriptors



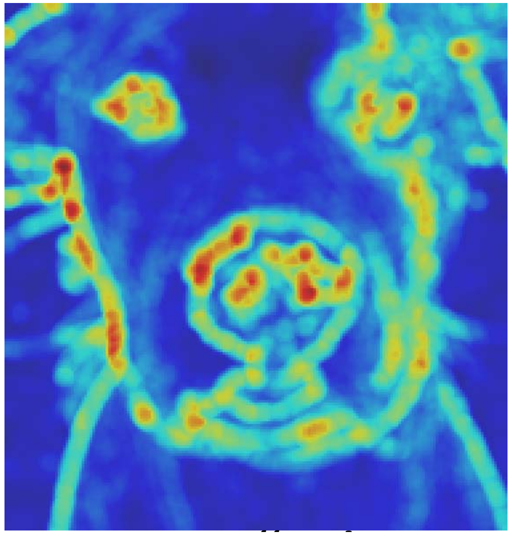
A Variant Better-suited for Restoration [Takeda et al. '07]

$$K(\mathbf{C}_l, \mathbf{x}_l, \mathbf{x}) = \boxed{\sqrt{\det \mathbf{C}_l}} \exp\left\{-(\mathbf{x}_l - \mathbf{x})'\mathbf{C}_l(\mathbf{x}_l - \mathbf{x})\right\}$$



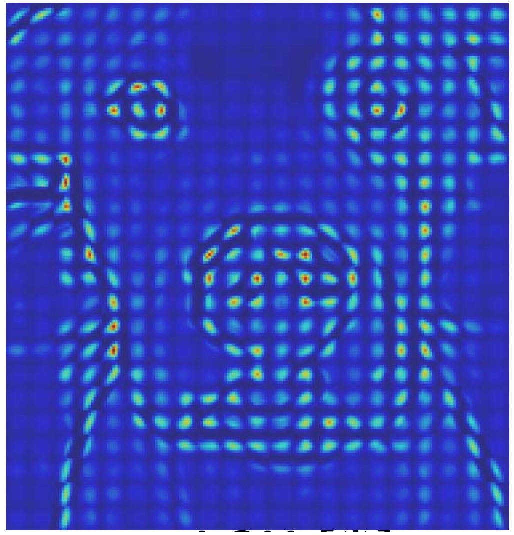
LARK

$\odot$



Edge strength

$=$



LSK

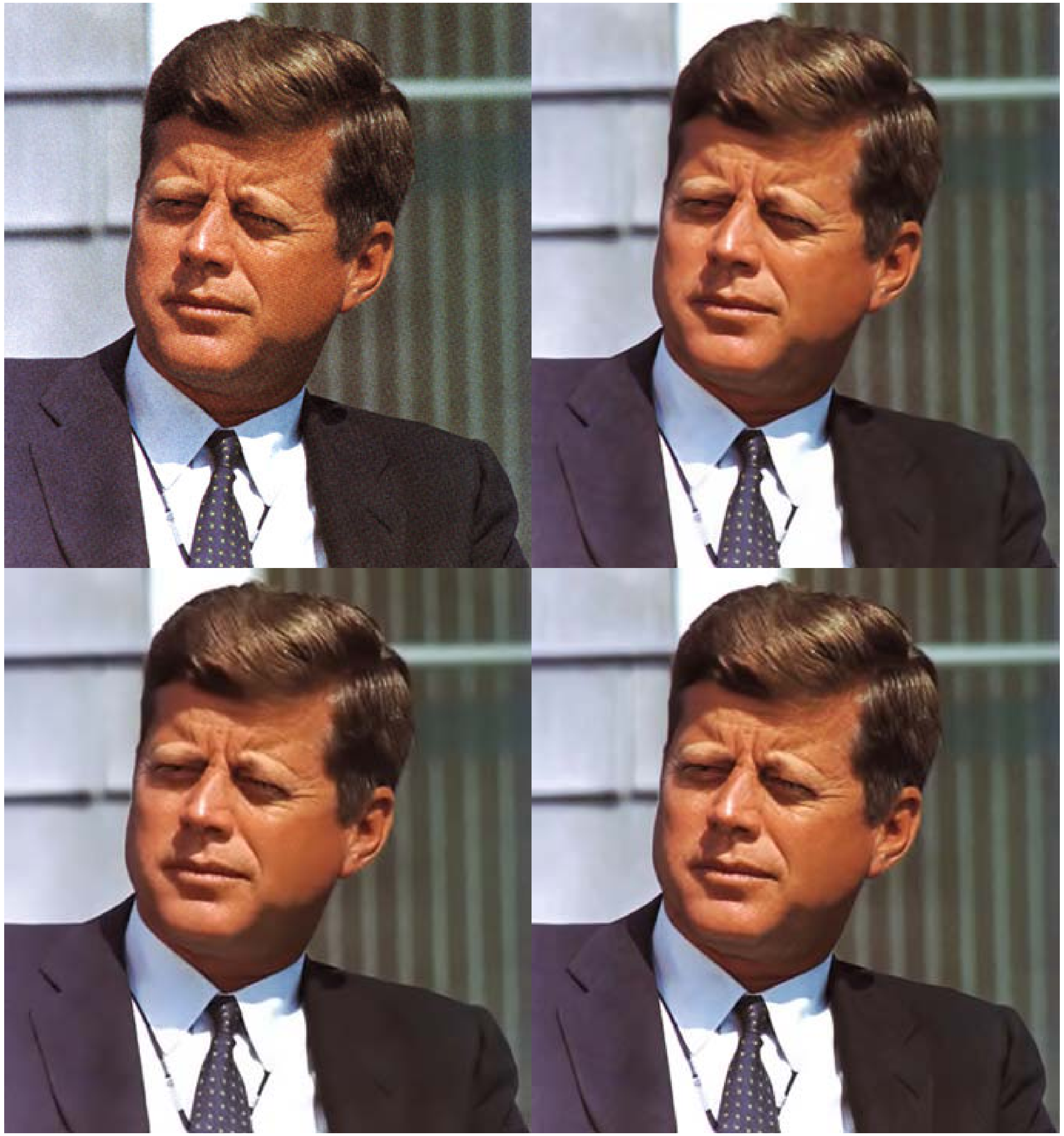
Film Grain Reduction (Real Noise)

Noisy image

Film Grain Reduction (Real Noise)

LARK

Film Grain Reduction (Real Noise)



LARK

KSVD

BM3D

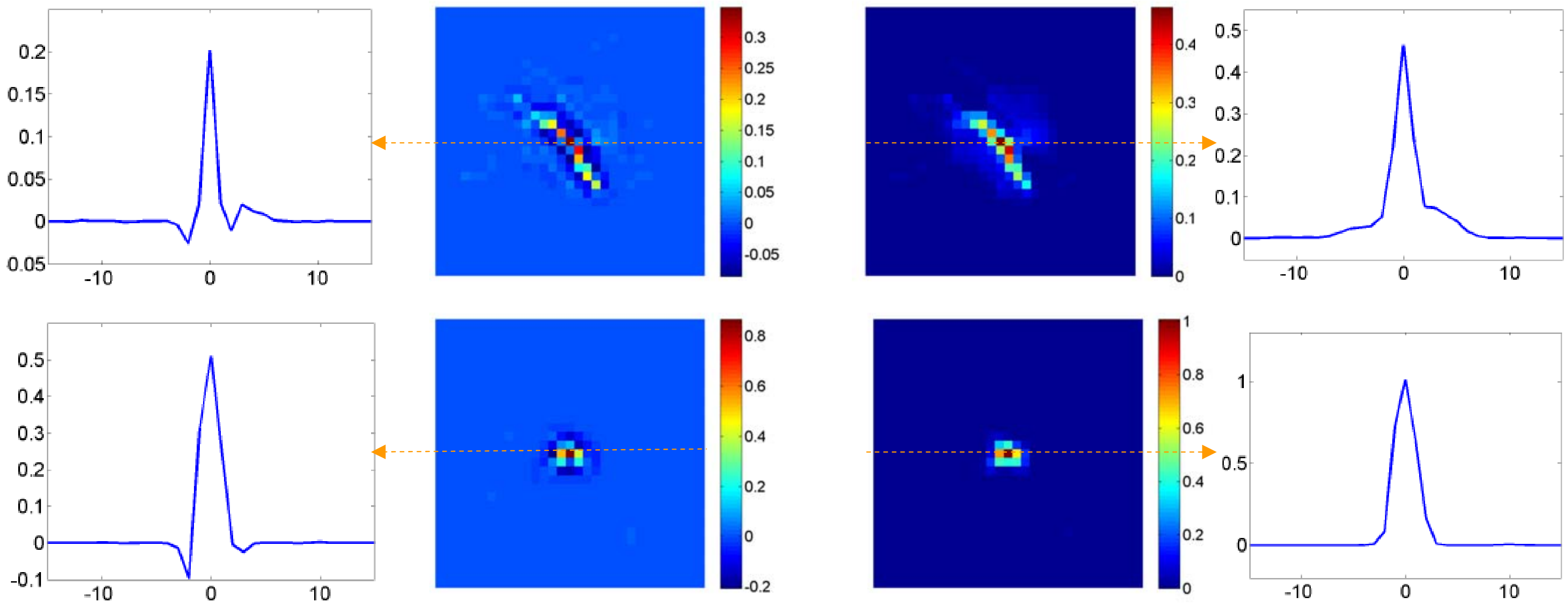
Adaptive Sharpening/Denoising

- Sharpening the LARK Kernel

"Sharpness" parameter

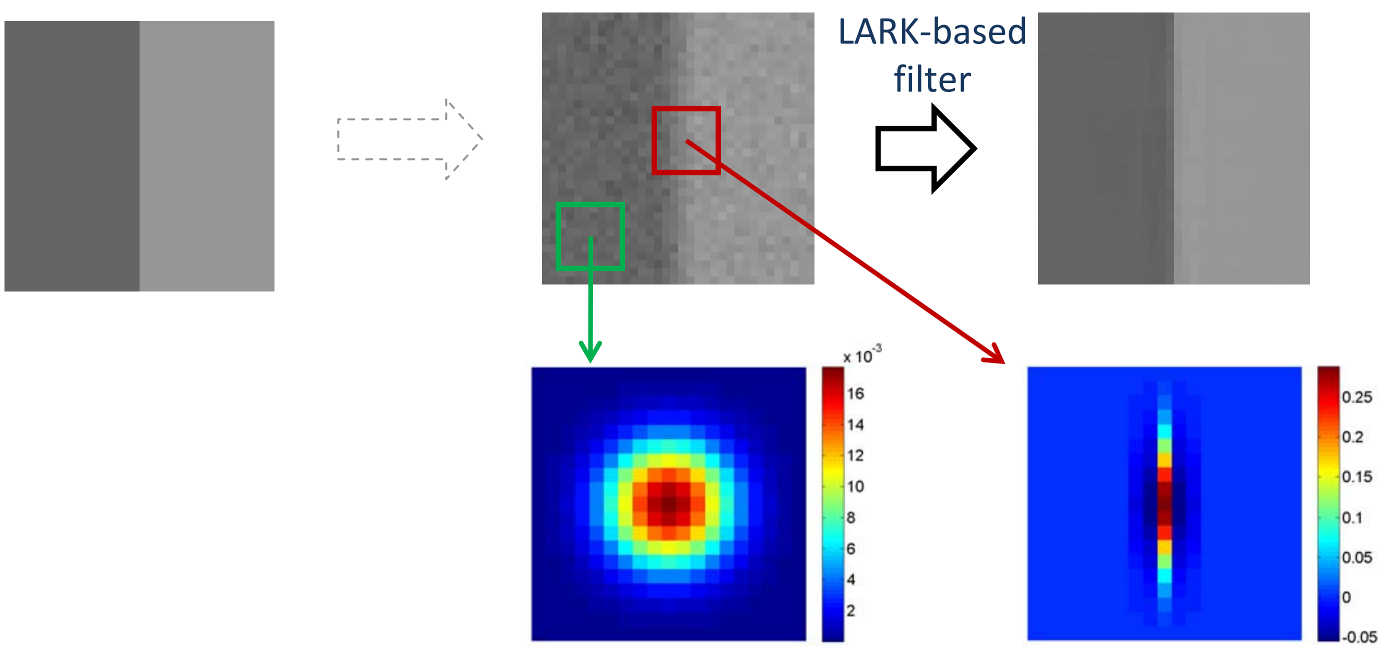
Laplacian operator

$$S = K - \kappa L \otimes K$$



LARK-based Simultaneous Sharpening/Deblurring/Denoising

- Net effect:
  - aggressive denoising in “flat” areas
  - Selective denoising and sharpening in “edgy” areas



Locally adaptive denoise/deblur filters

Examples

original image

original image

LARK

state-of-the-are methods

Examples

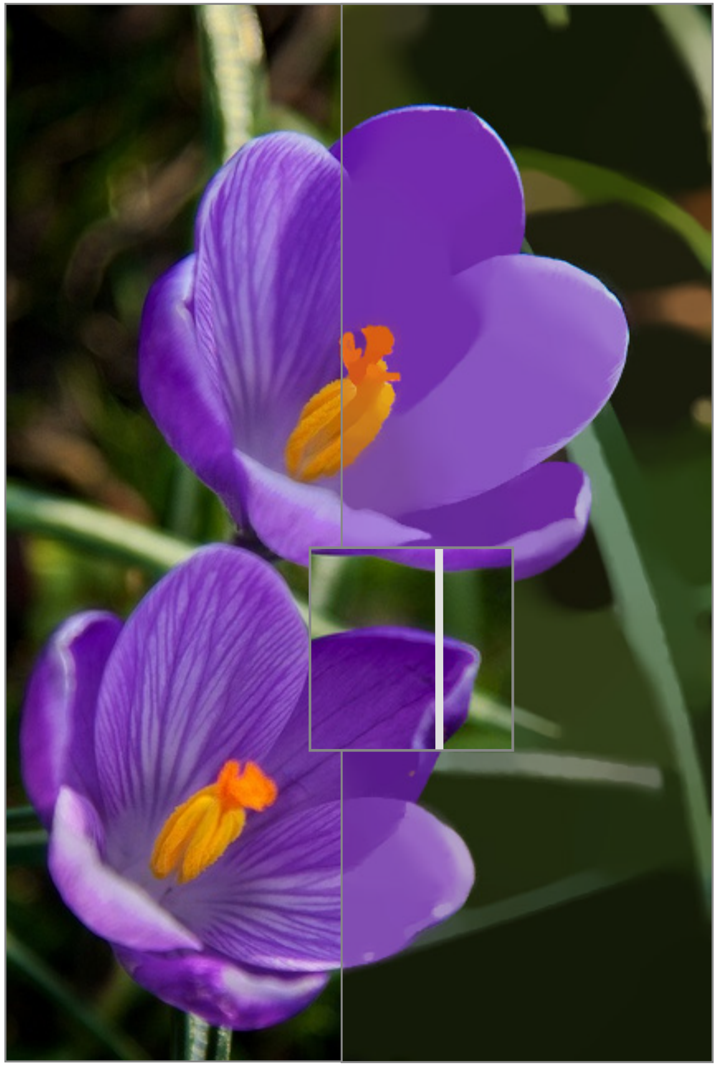
Examples

LARK

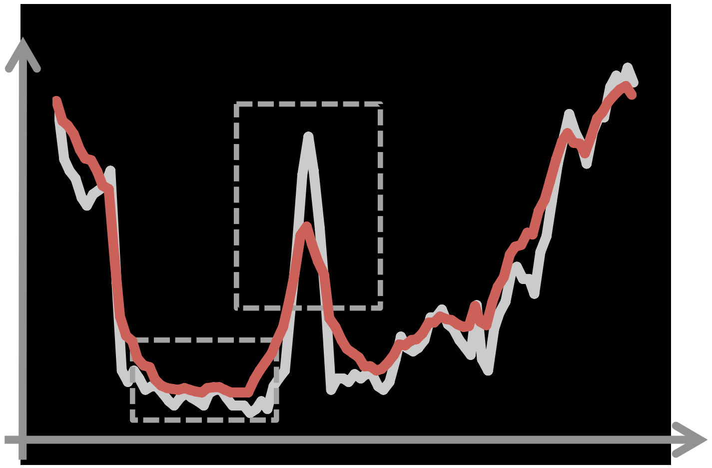
LARK-based Image filtering – Summary

- another patchwise formulation
- considers a Riemannian metric
- encodes local geometry
- better similarity measurements

Image Smoothing



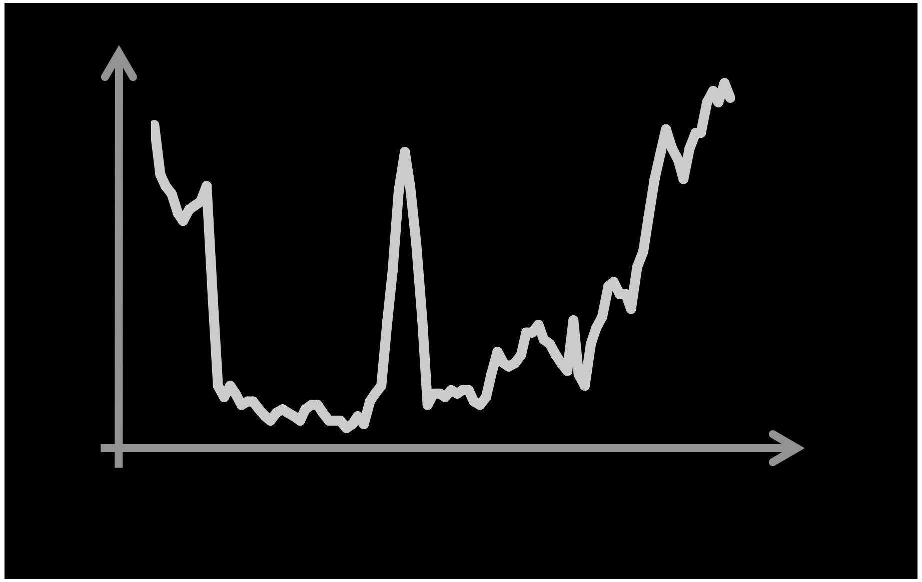
Recall the general goals:
- Suppress insignificant details
- Maintain major edges



L0 Smoothing Method

A general and effective global smoothing strategy based on <u>a sparsity measure</u>

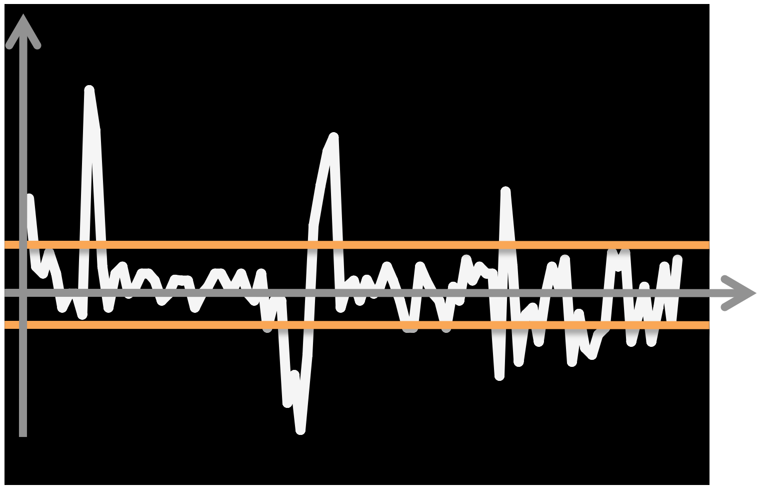
$$c(f) := \#\{p \mid |\nabla f_p| \neq 0\}$$



L0 Smoothing Method

A general and effective global smoothing strategy based on <u>a sparsity measure</u>

$$c(f) := \#\{p \mid |\nabla f_p| \neq 0\}$$

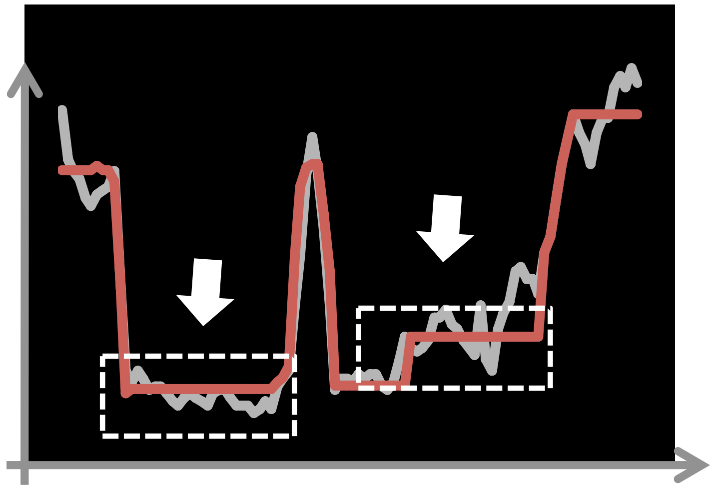


which corresponds to the L0-norm of gradient

Two Features

1. <u>Flattening</u> insignificant details

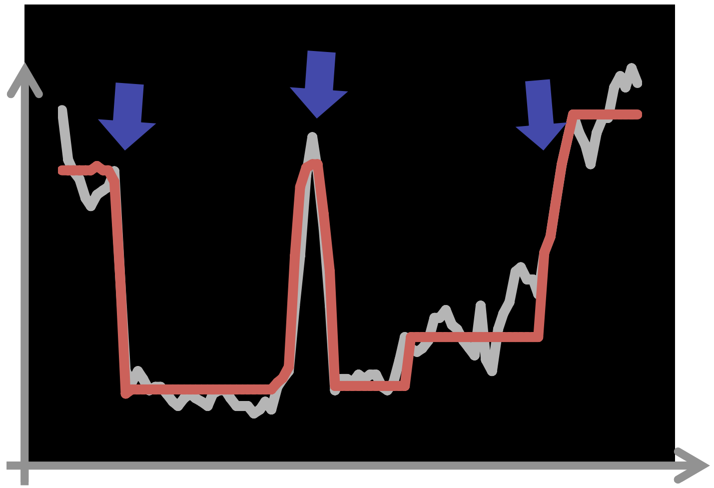
By removing small non-zero gradients



Two Features

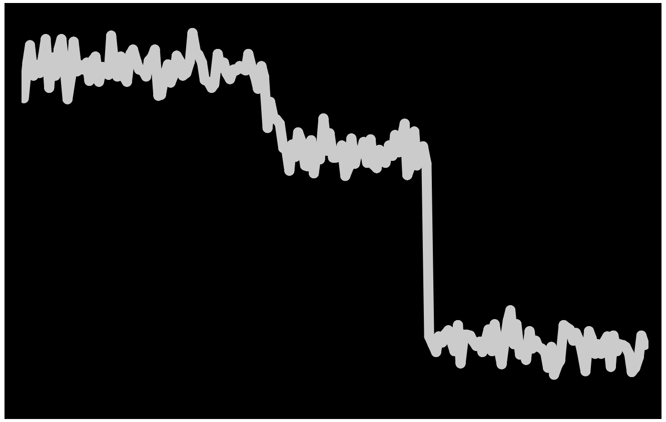
2. Enhancing prominent edges

Because large gradients receive the same penalty as small ones



$$\#\{p \mid |\nabla f_p| \neq 0\} = \#\{p \mid |\alpha \nabla f_p| \neq 0\}$$

Framework in 1D



- Constrain # of non-zero gradients

$$c(f) = \#\{p \,|\, \left| f_p - f_{p+1} \right| \neq 0\} = k$$

- Make the result similar to the input

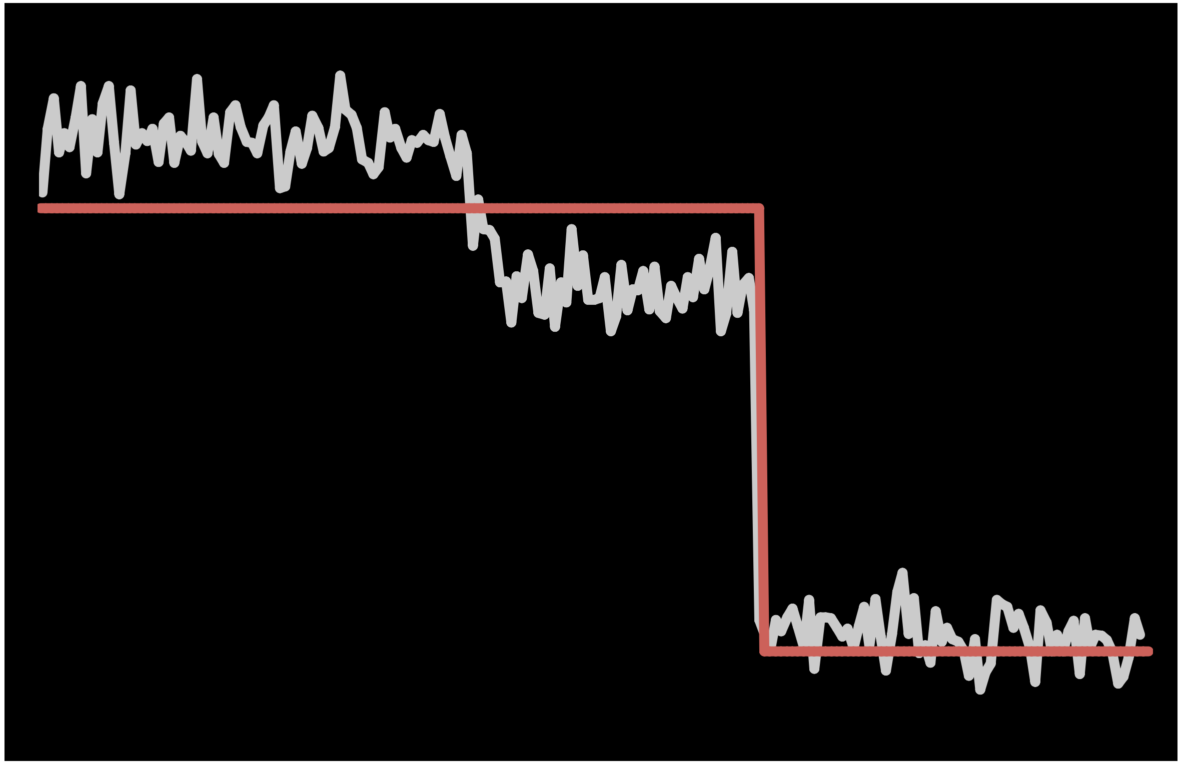
$$\min_f \sum_p (f_p - g_p)^2$$

- Objective function

$$\min_f \sum_p (f_p - g_p)^2 \qquad \text{s.t.} \qquad c(f) = k$$

Framework in 1D

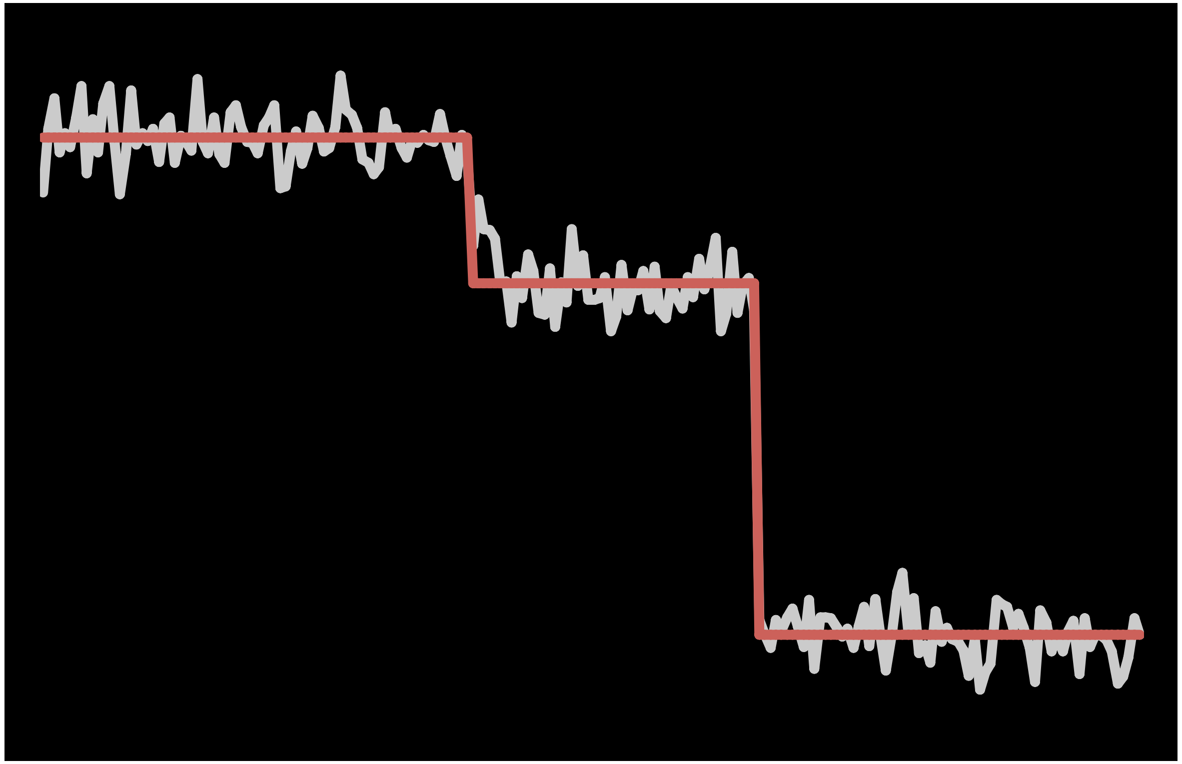
- Input 1D signal g



$$\min_f \sum_p (f_p - g_p)^2 \qquad \text{s.t.} \qquad c(f) = 1$$

Framework in 1D

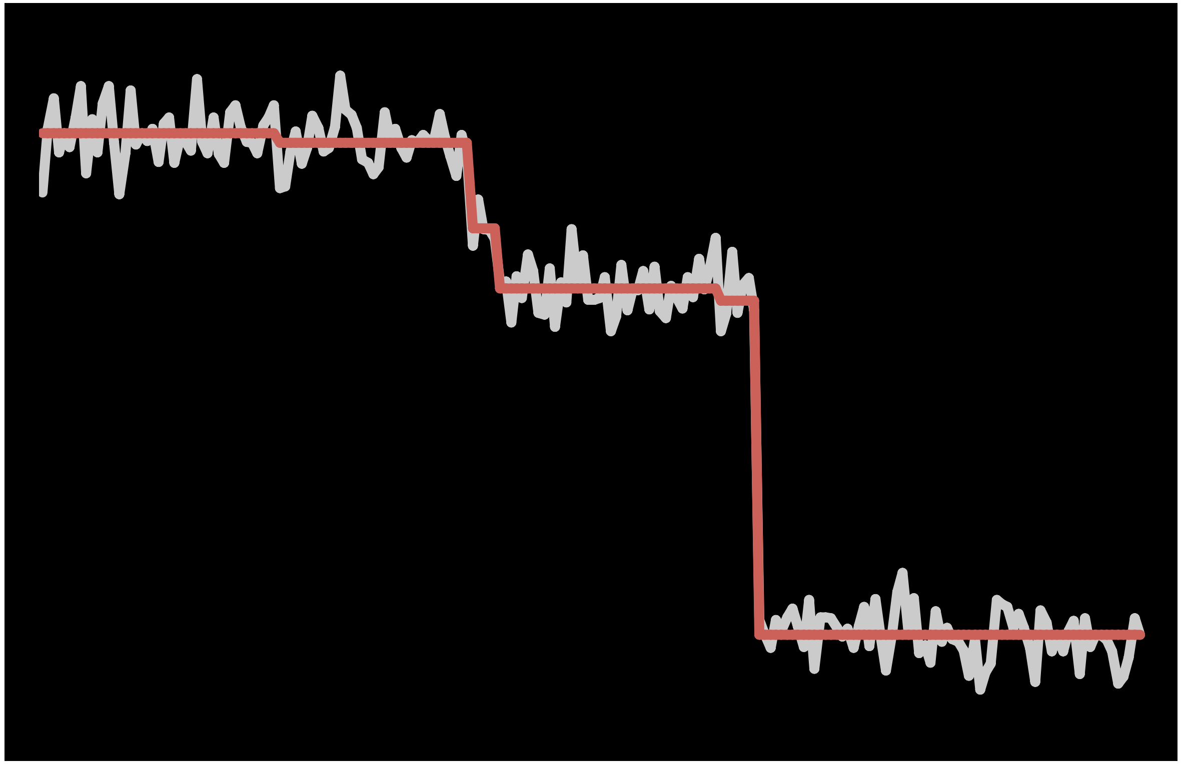
- Input 1D signal g



$$\min_f \sum_p (f_p - g_p)^2 \qquad s.t. \qquad c(f) = 2$$

Framework in 1D

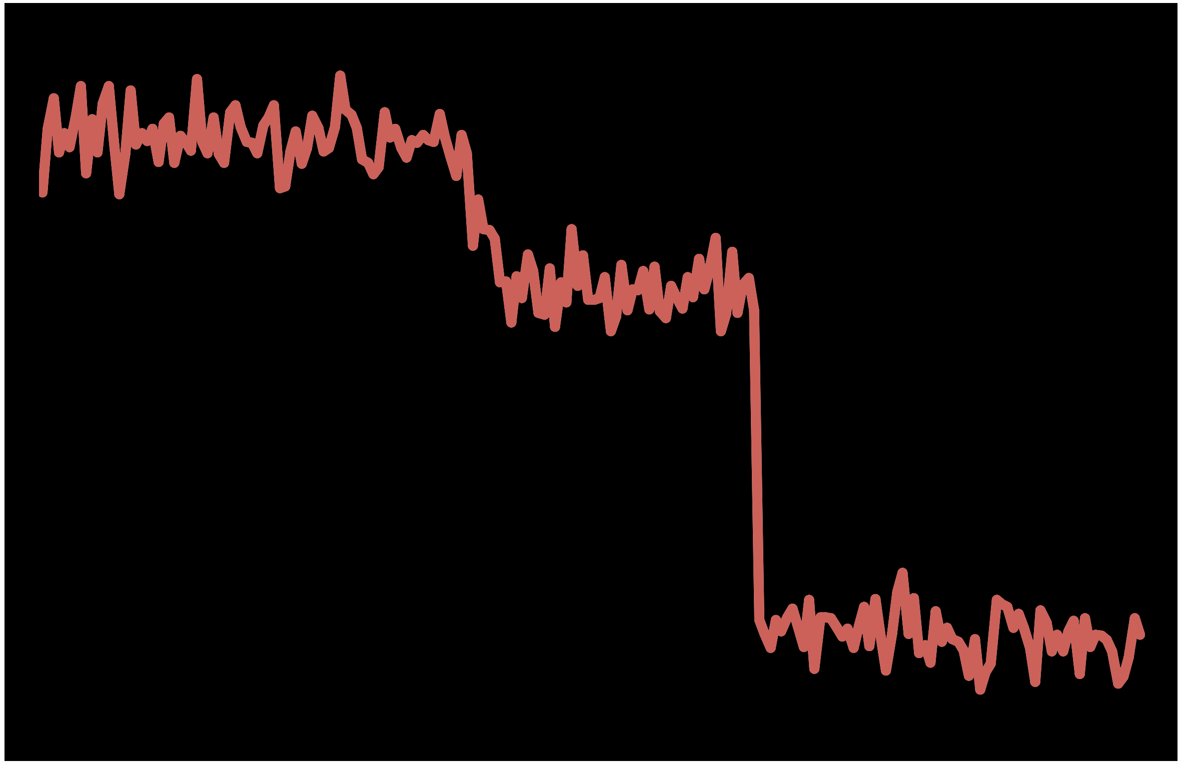
- Input 1D signal g



$$\min_f \sum_p (f_p - g_p)^2 \qquad s.t. \qquad c(f) = 5$$

Framework in 1D

- Input 1D signal g



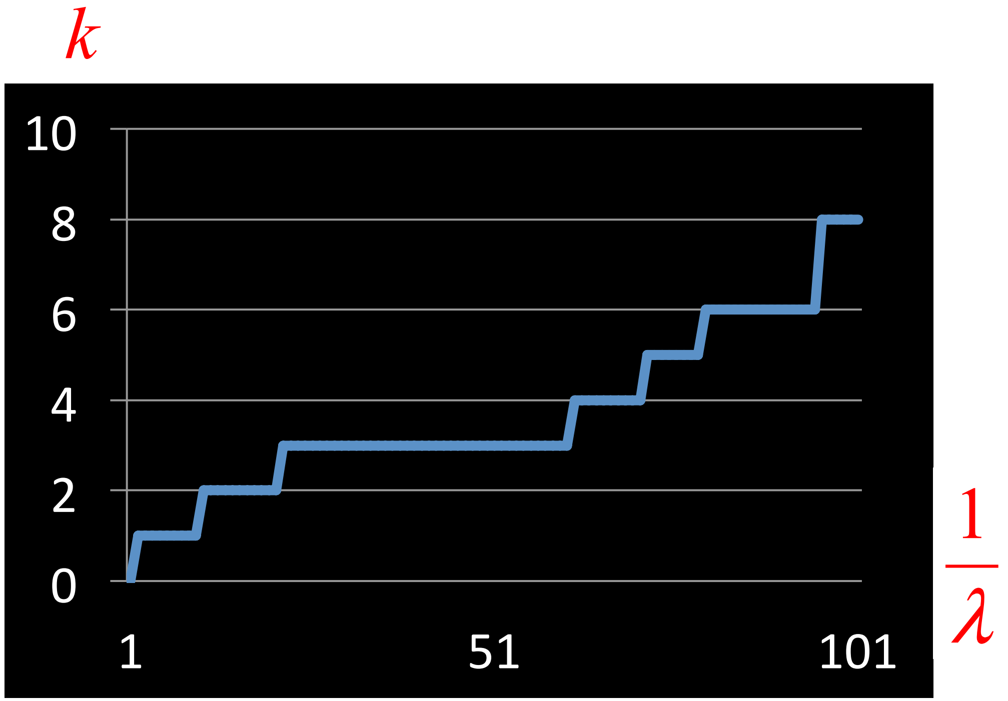
$$\min_f \sum_p (f_p - g_p)^2 \qquad s.t. \qquad c(f) = 200$$

Transformation

$$\min_f \sum_p (f_p - g_p)^2 \quad \text{s.t.} \quad c(f) = k$$

⬇

$$\min_f \sum_p (f_p - g_p)^2 + \lambda \cdot c(f)$$



2D Image

$$\min_f \sum_p (f_p - g_p)^2 + \lambda \cdot c(\partial_x f, \partial_y f)$$

$$c(\partial_x f, \partial_y f) = \#\{p \mid |\partial_x f_p| + |\partial_y f_p| \neq 0\}$$

Finding the global optimum is NP hard

Approximation

$$\min_f \sum_p (f_p - g_p)^2 + \lambda \cdot c(\quad h, v \quad)$$

$$+\beta \cdot \sum_p \left((\partial_x f_p - h_p)^2 + (\partial_y f_p - v_p)^2 \right)$$

Separately estimate f and (h, v)

Iterative Optimization

- Compute f given h, v

$$E(f) = \sum_p (f_p - g_p)^2 + \beta \cdot \left((\partial_x f_p - h_p)^2 + (\partial_y f_p - v_p)^2 \right)$$

- Compute h, v given f

$$E(h,v) = \sum_p \left((\partial_x f_p - h_p)^2 + (\partial_y f_p - v_p)^2 \right) + \frac{\lambda}{\beta} c(h,v)$$

- Gradually approximate the original problem

$$\beta \leftarrow 2\beta$$

Iterative Optimization

- Compute f given h, v

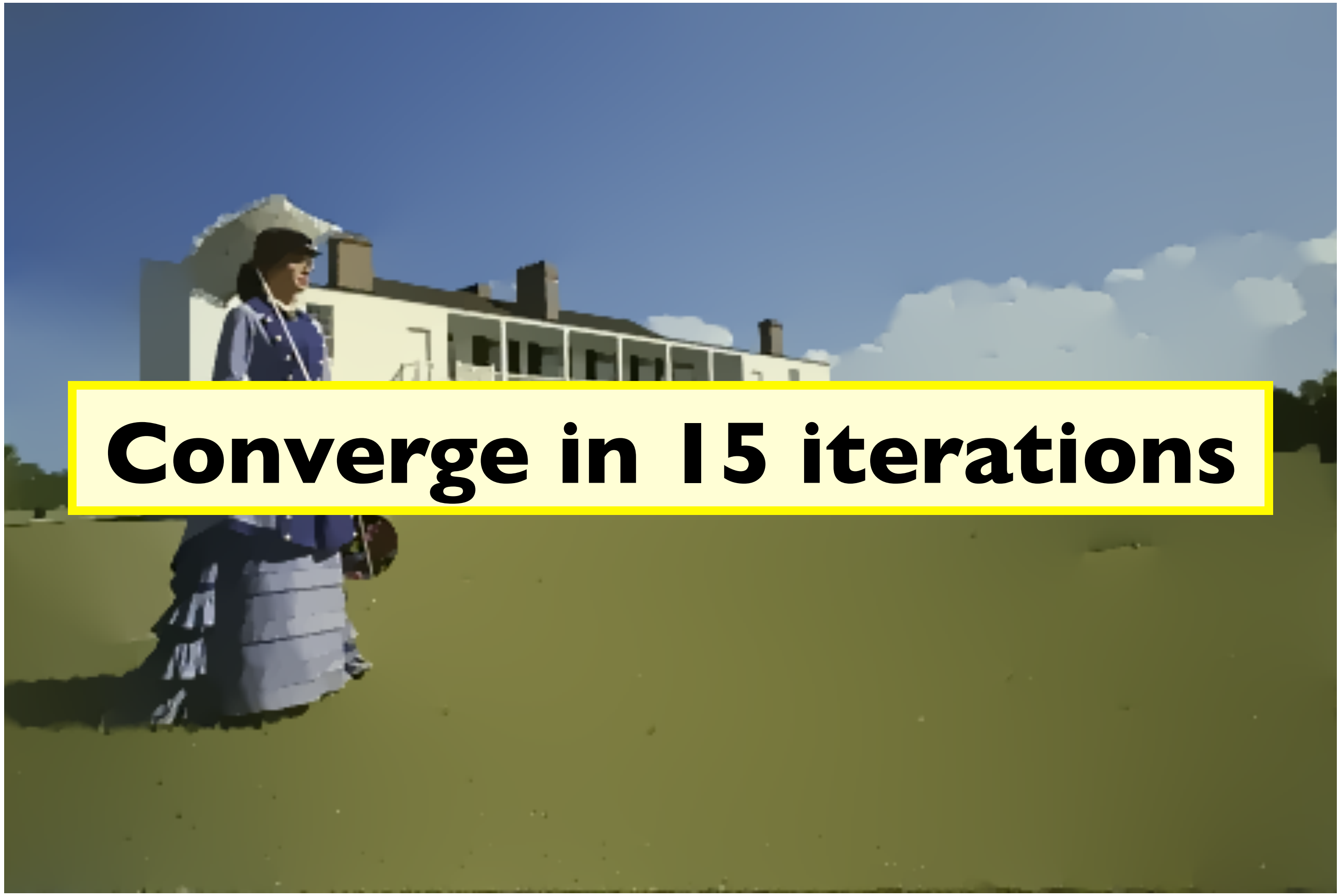
$$E(f) = \sum_p (f_p - g_p)^2 + \beta \cdot \left((\partial_x f_p - h_p)^2 + (\partial_y f_p - v_p)^2 \right)$$

Both the sub-problems are with closed-form solutions

- Gradually approximate the original problem

$$\beta \leftarrow 2\beta$$

One Example



Iteration #

Smoothing Strength

Input

Smoothing Strength

Smoothing Strength

λ=0.02

Smoothing Strength

λ=0.03

Comparison

L0 Smoothing Result
Total Variation
BWLS

Comparison

L0 smoothing Result

Comparison

BLF

Comparison

Total Variation

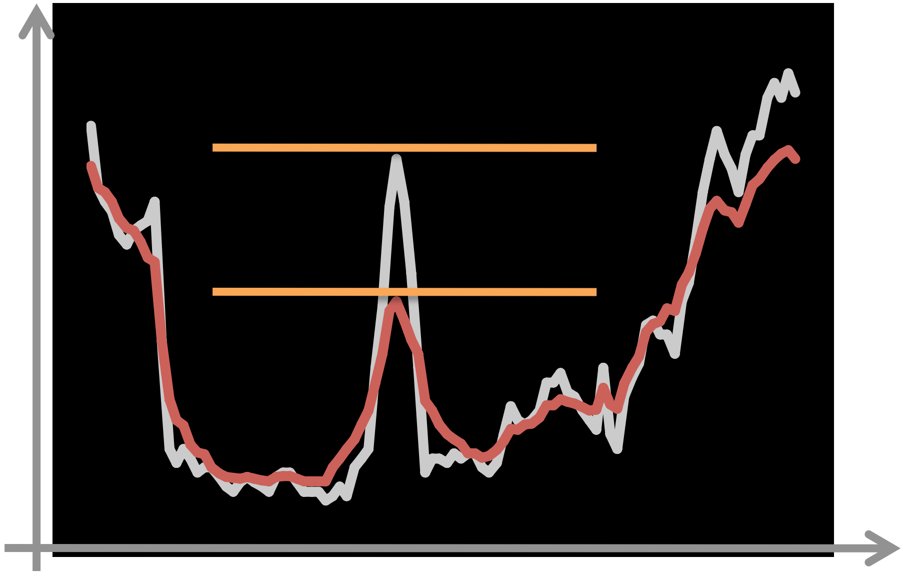
Comparison

WLS

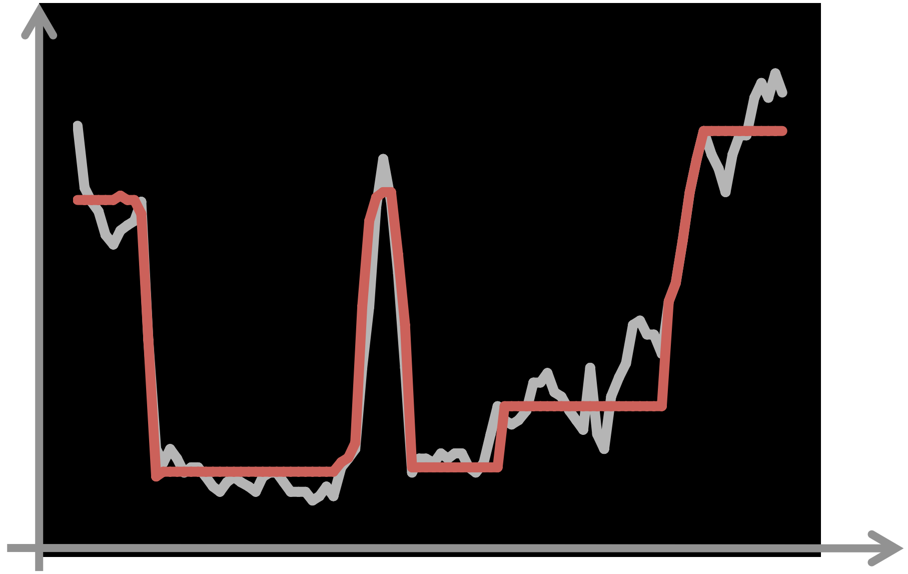
Comparison

L0 smoothing Result

Comparison

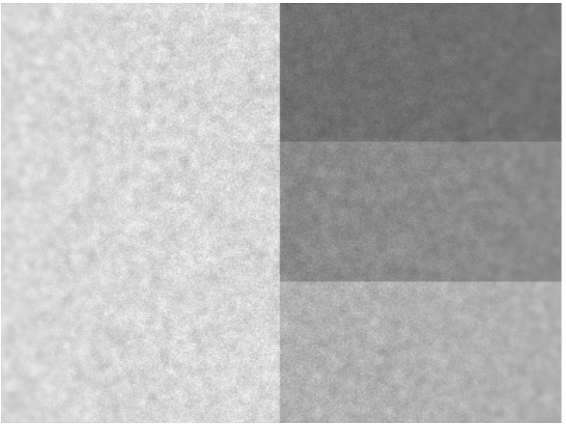


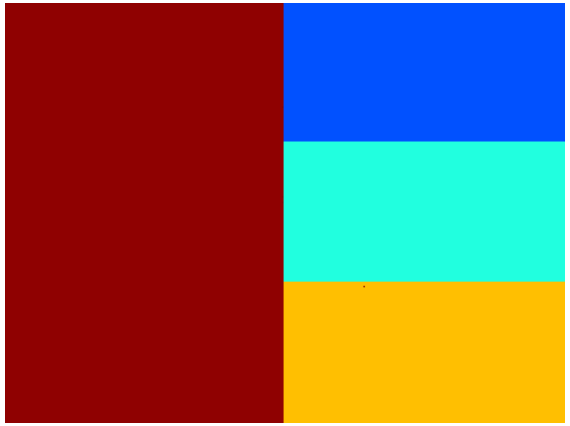
Total Variation



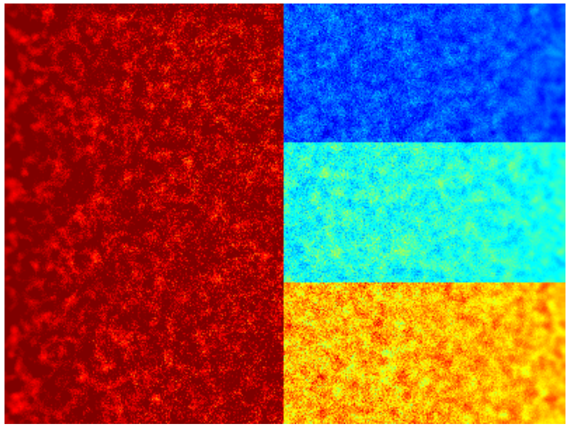
L0 Smoothing Result

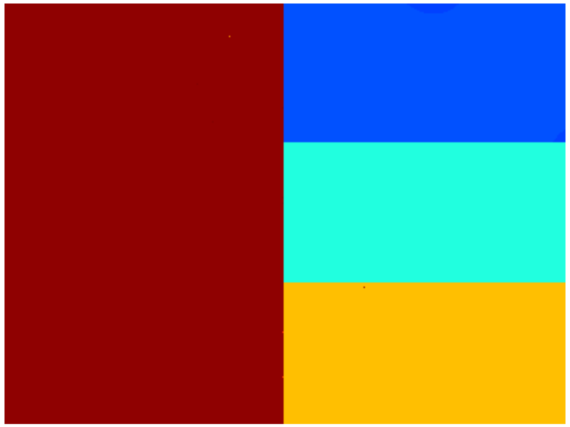
Another Example

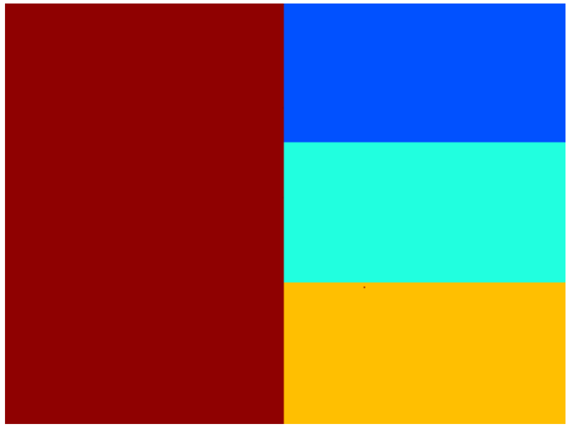




5 times







Input

1 times

20 times

Edge Enhancement and Extraction

Edge Enhancement and Extraction

Gradient Map

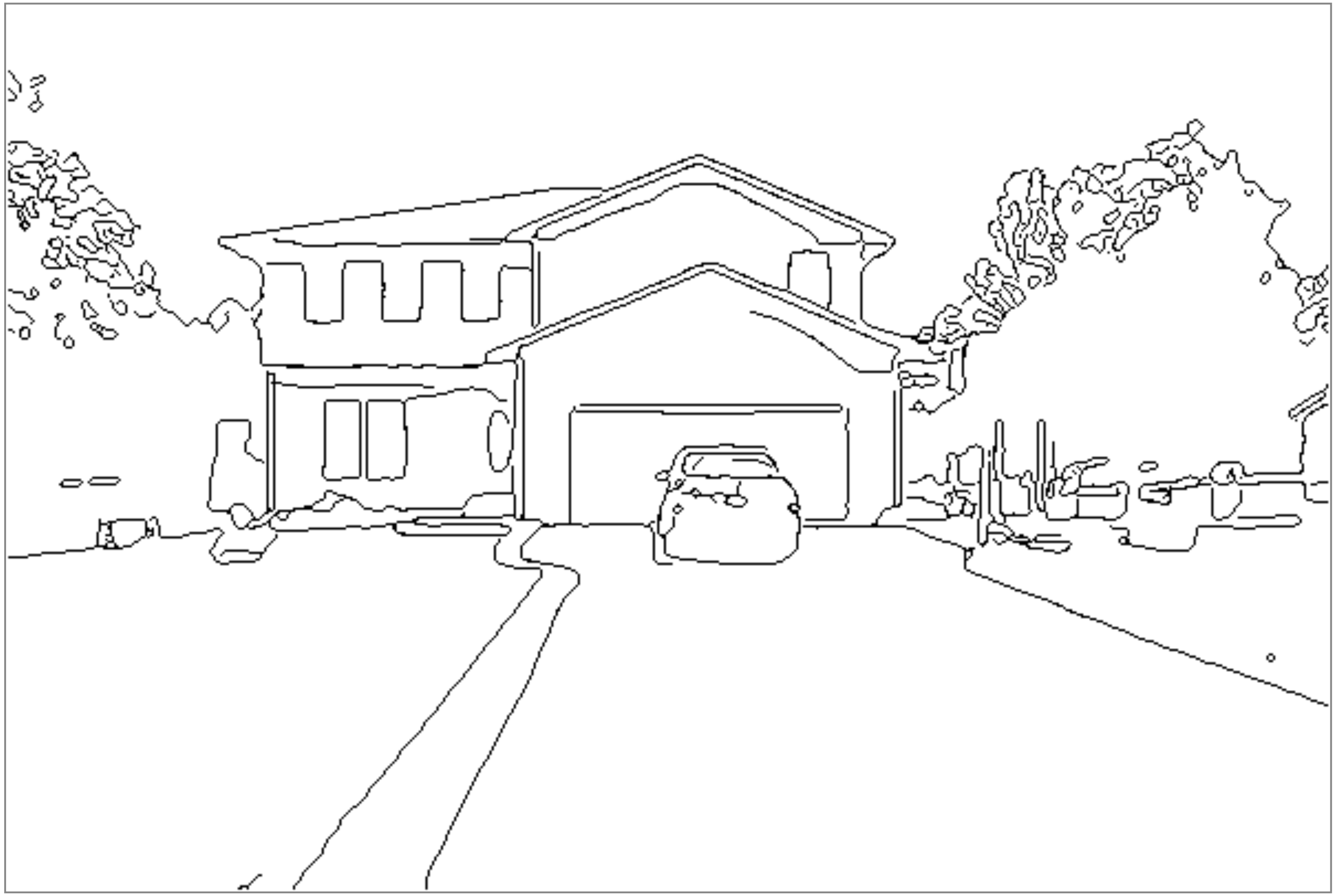
Edge Enhancement and Extraction

Extracted Edge

Edge Enhancement and Extraction

L0 Smoothing result

Edge Enhancement and Extraction



Extracted Edge

Edge Enhancement and Extraction

Edge Enhancement and Extraction

Edge Enhancement and Extraction

Edge Enhancement and Extraction

Without smoothing

With smoothing

Edge Enhancement and Extraction

Without smoothing

With smoothing

Image Abstraction

Image Abstraction

Pencil Sketch

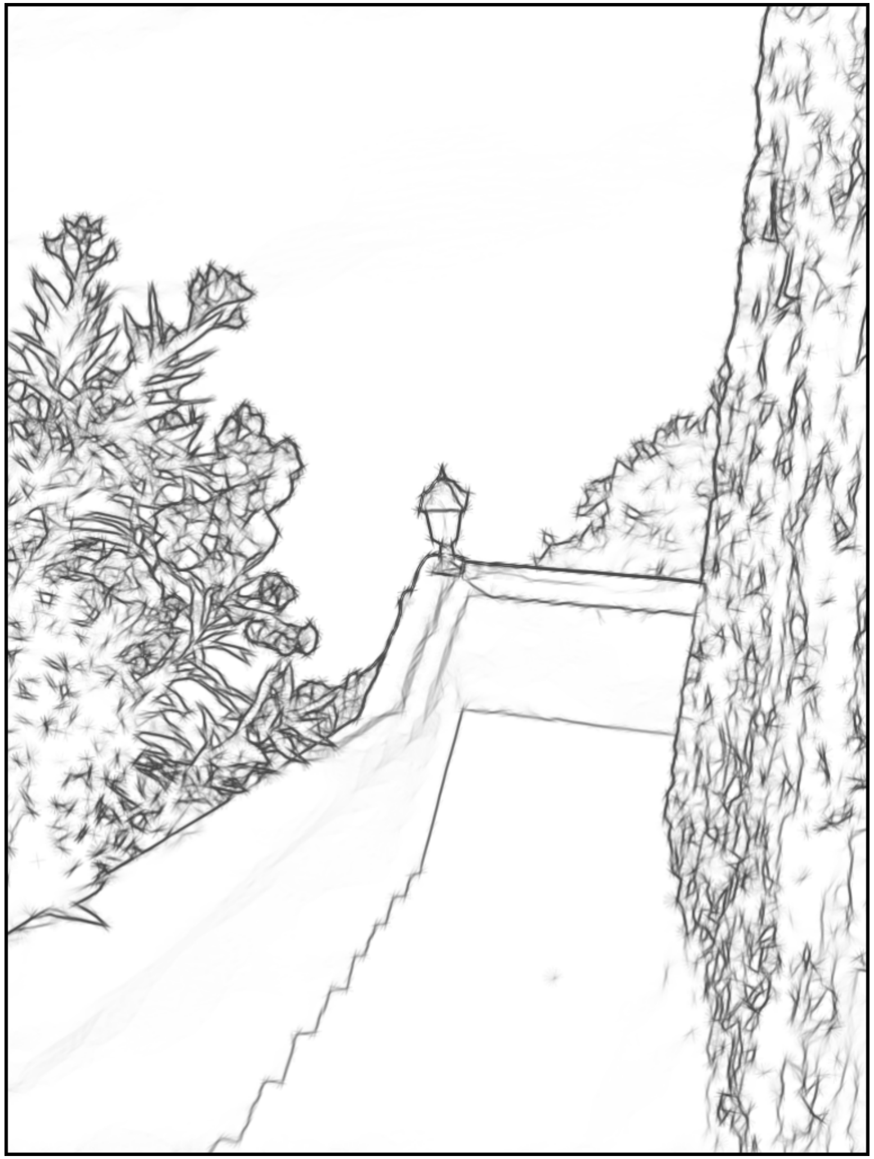
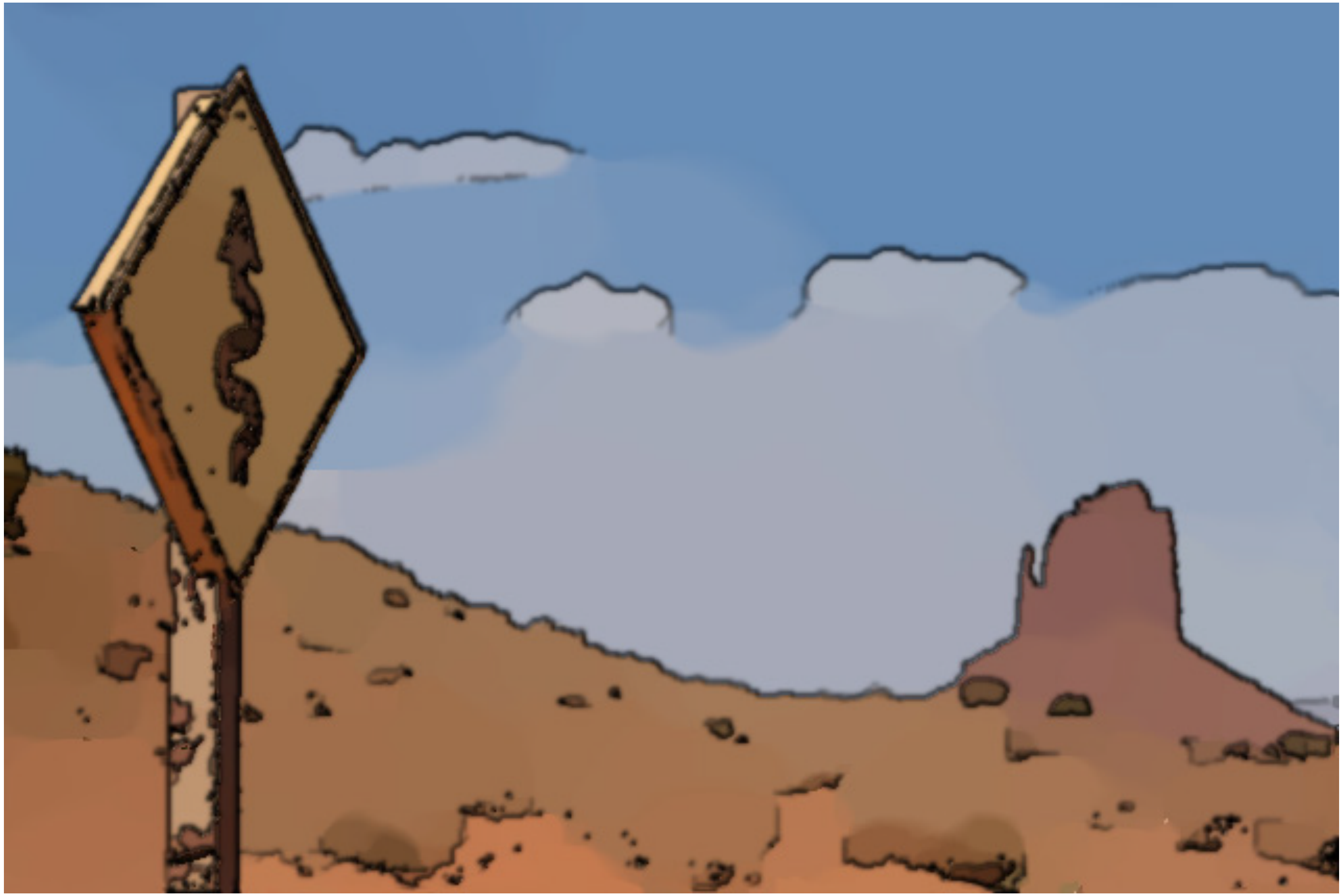
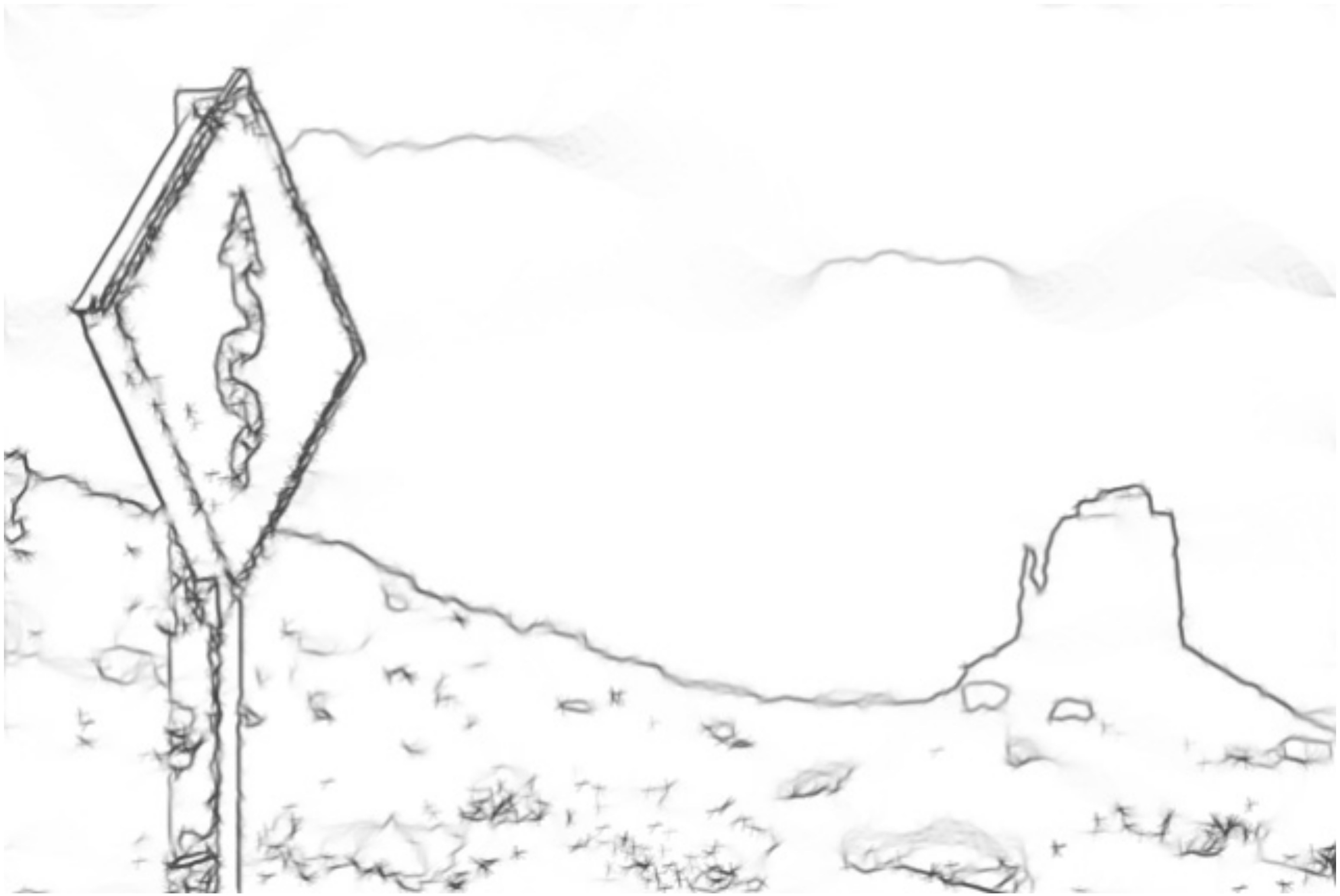


Image Abstraction

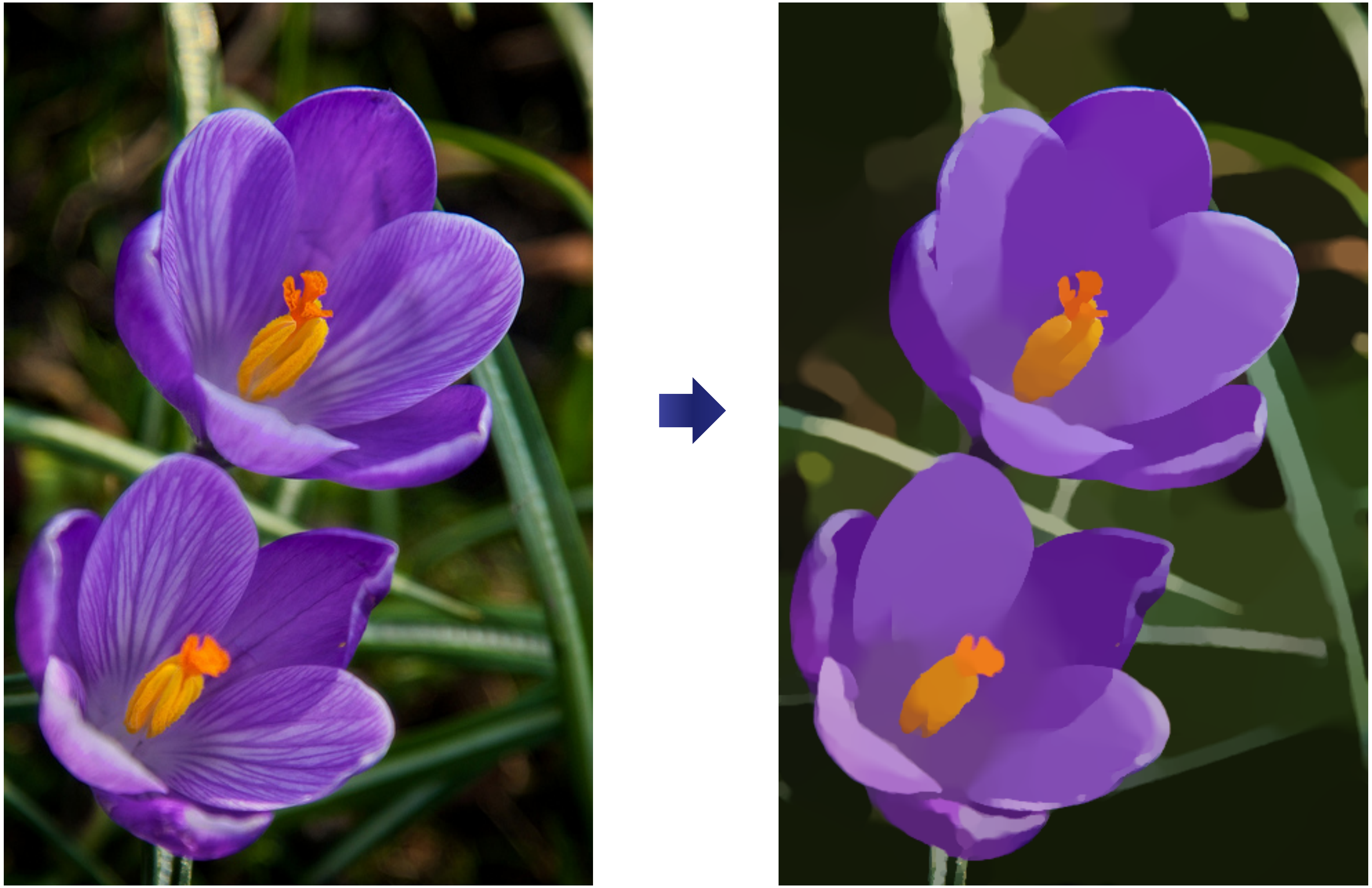
Image Abstraction



Pencil Sketch



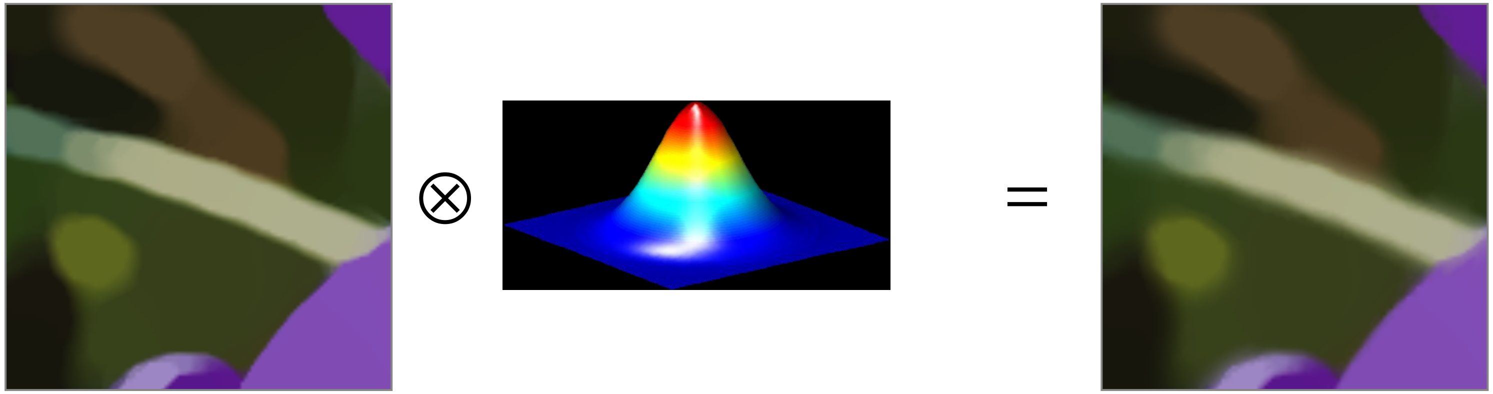
Detail Manipulation



Detail Manipulation

Base layer

Edge Adjustment



Spatially varying Gaussian blur in an optimization procedure

Edge Adjustment

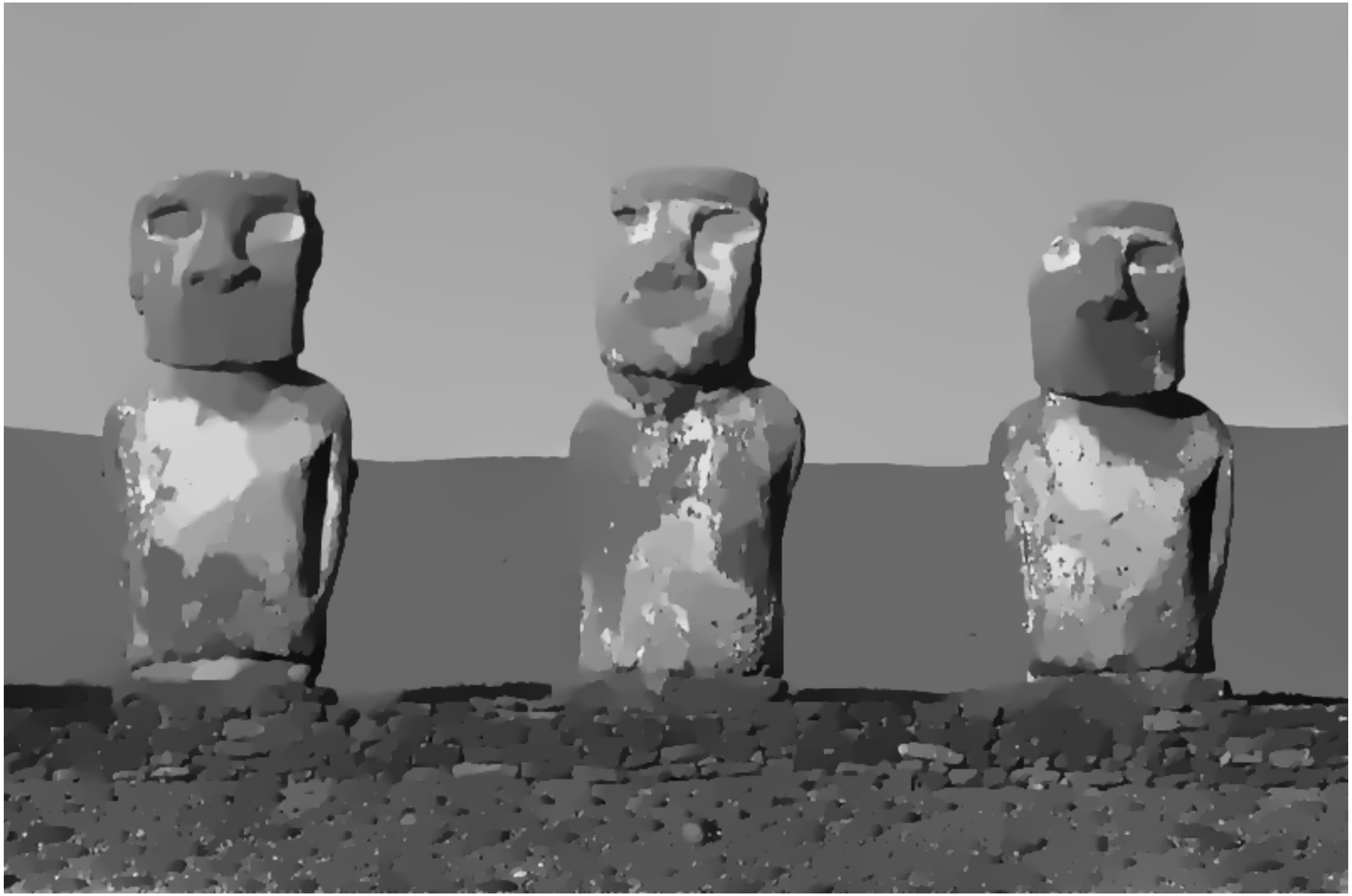
Input

Detail Boosting

Detail Manipulation

Image of [Farbman et al. 08]

Detail Manipulation



Detail Manipulation

Combined with other smoothing

Strong texture will be preserved

Combined with other smoothing

L0 smoothing result

Combined with other smoothing

Bilateral Filter

Combined with other smoothing

Bilateral Filter + L0 smoothing

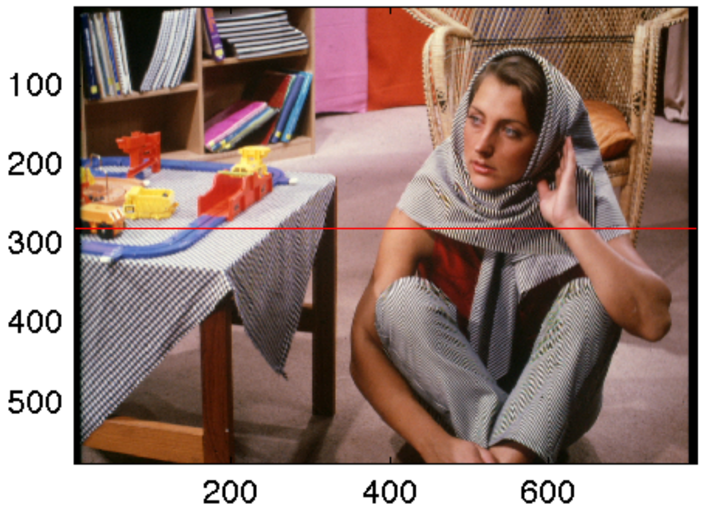
L0 Smoothing - Summary

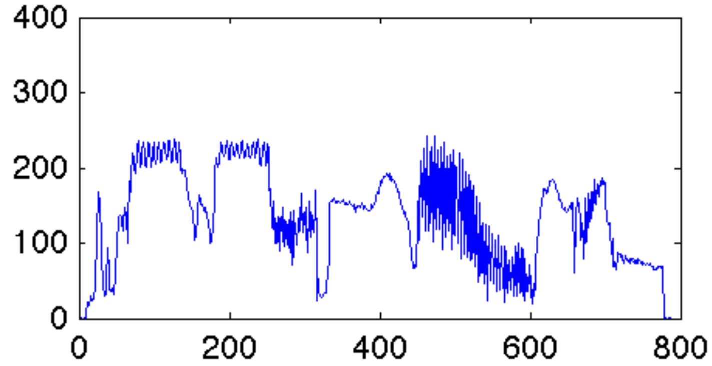
- A simple and general smoothing framework
- Approximate L0-norm gradient measure
- Flatten low-amplitude details
- Enhance prominent structures

Method of Karacan, Erdem and Erdem *(work in progress)*

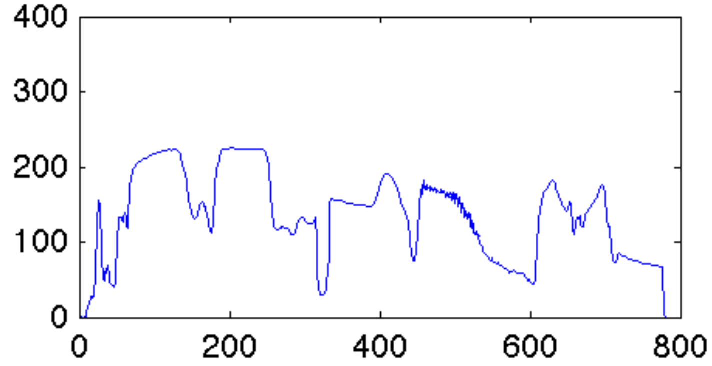
- A patch-based approach for smoothing
- Uses a descriptor which encodes local geometry
- Decomposes an image into structure and noise+texture components

Method of Karacan, Erdem and Erdem *(work in progress)*

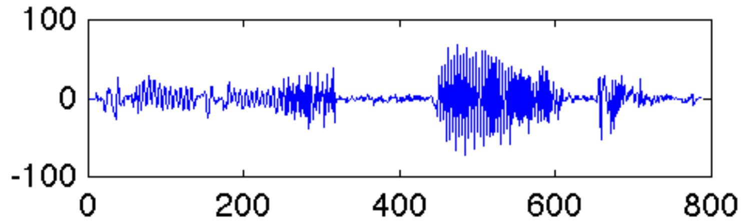




input image



structure



oscillatory

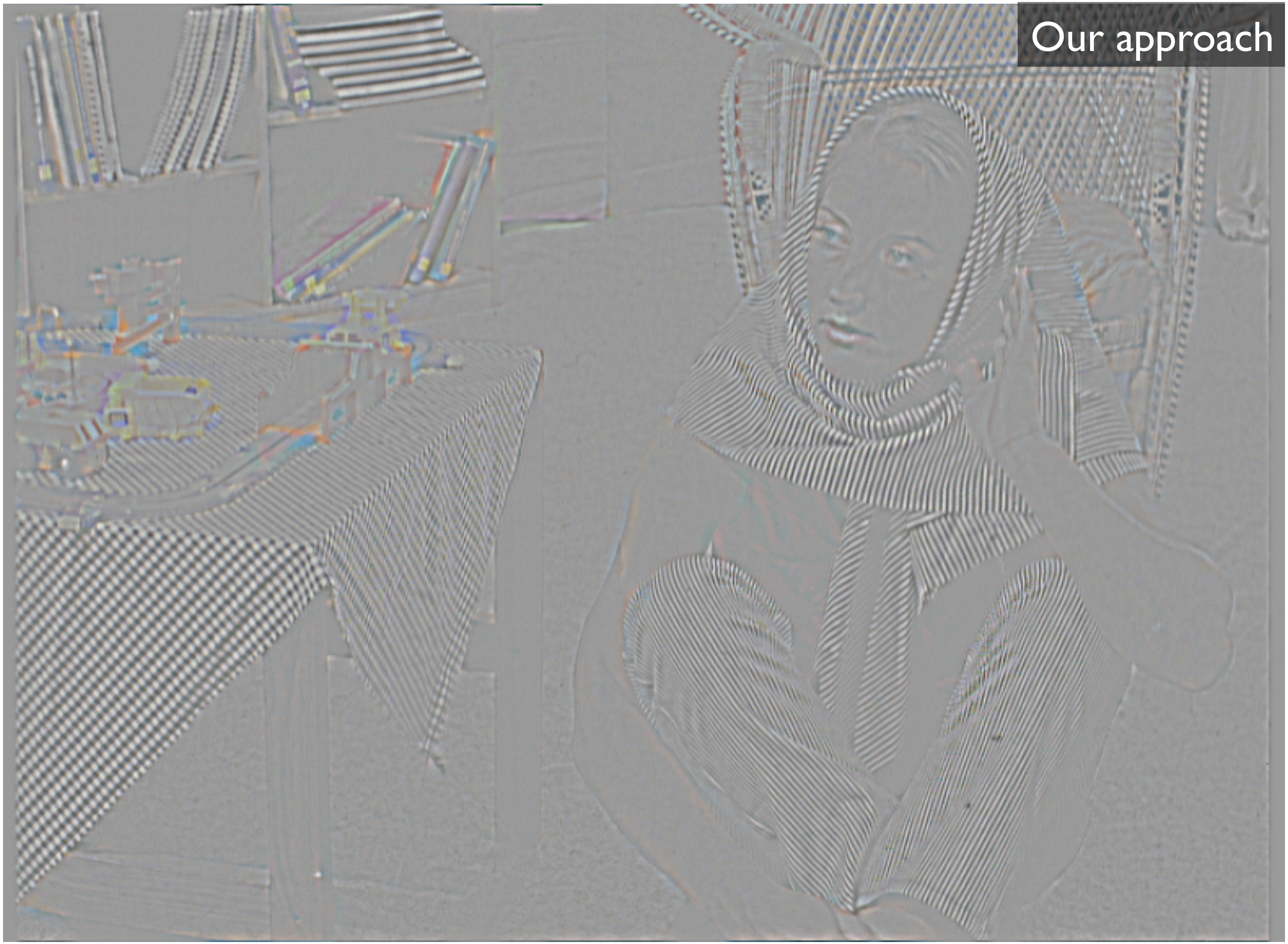

input image


Bilateral Filter


L0 smoothing


L0 smoothing


Our approach

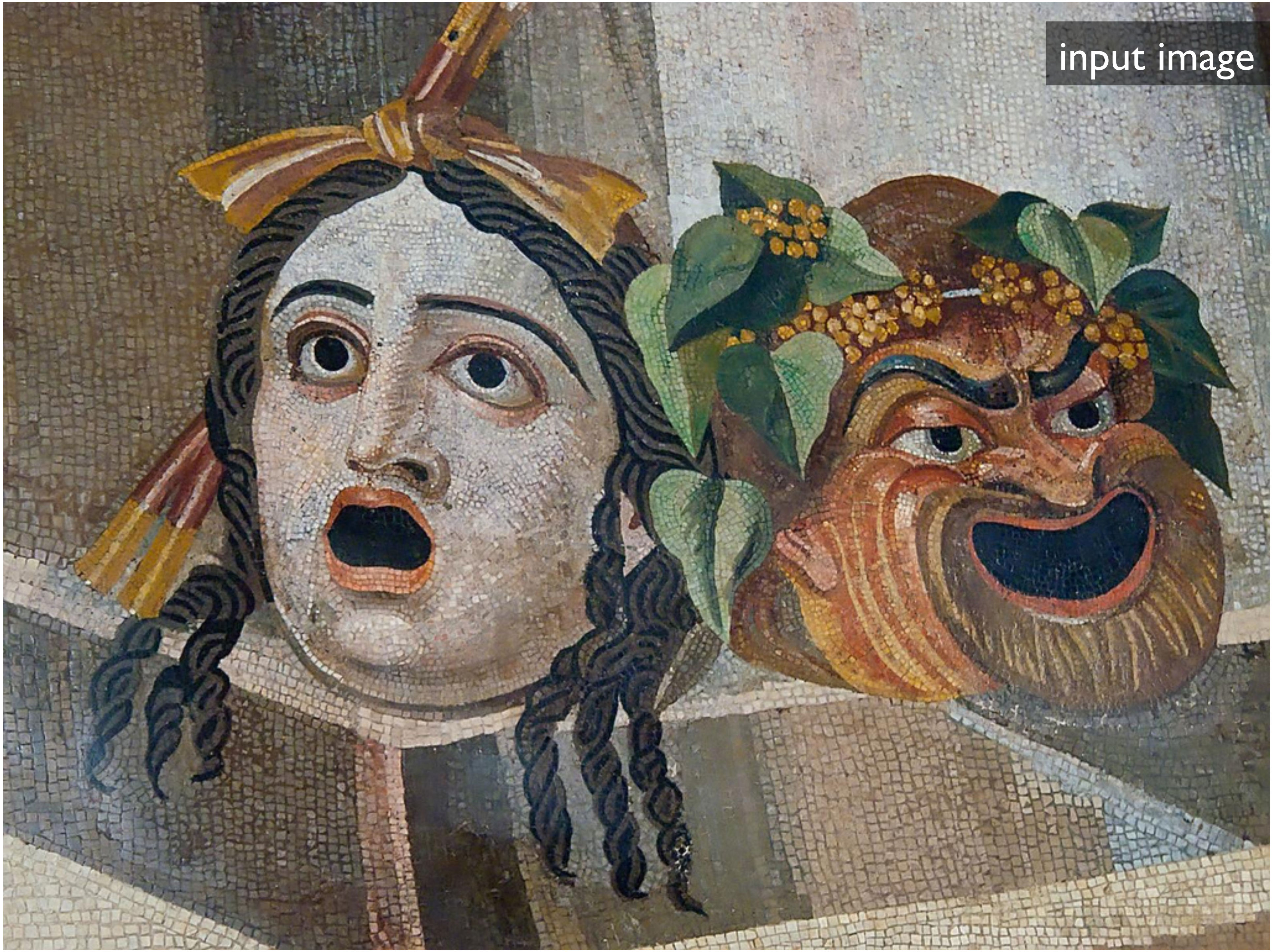

Our approach


input image

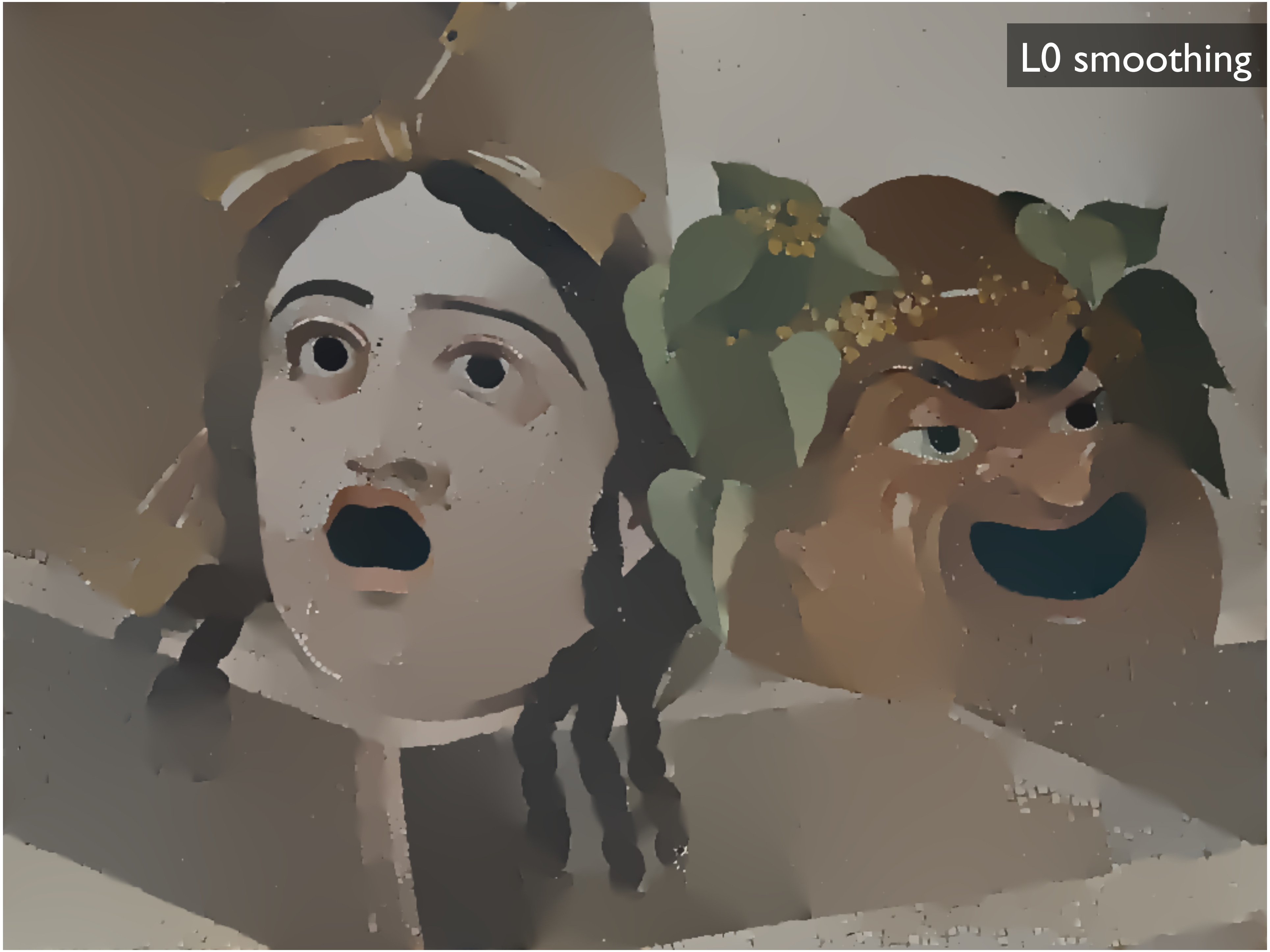

input image

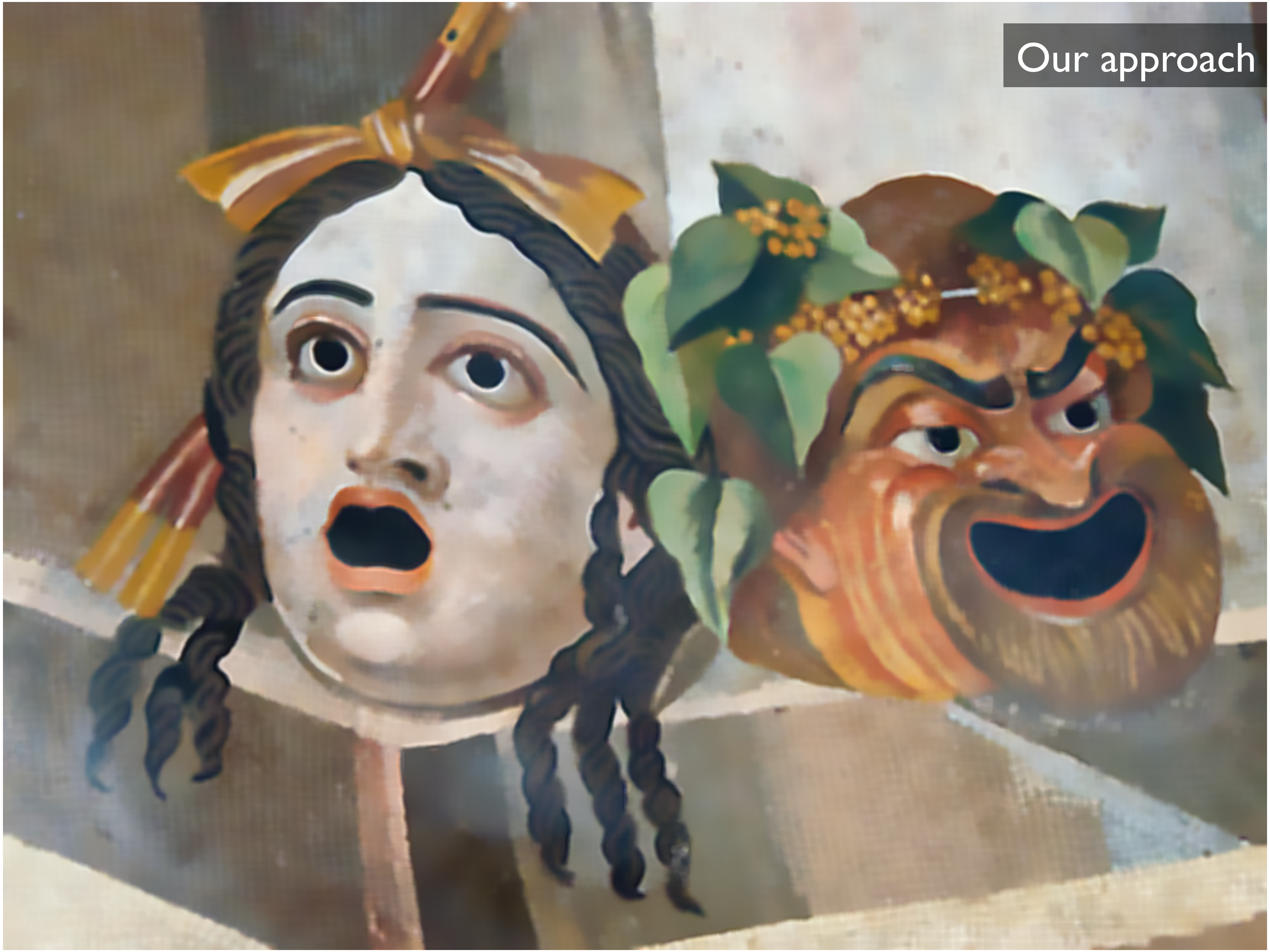

Our approach


Our approach


input image


Bilateral filter


L0 smoothing


Our approach