Efficient Edge-Aware Filtering and Their Applications

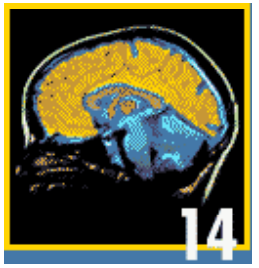
Jiangbo Lu, Dongbo Min,
Minh N. Do

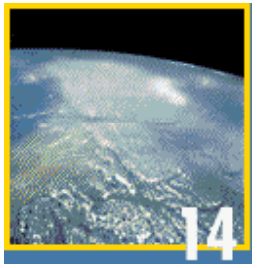
Advanced Digital Sciences Center
Computational Imaging Group
Department of Electrical and Computer Engineering

Imaging

- In 1900…
  - Only simple microscopes, telescopes, and black and white photographs.

- In 2000…
  - Color photographs, movies, and television
  - New imaging that greatly expand our “vision”
    - see tiny atomic particles to vast galaxies
    - see inside the human body
    - track weather patterns, …
  - Improvement in image quality due to advances in sensing and computation





Filtering: data aggregation to reduce noise

If $X_1, X_2, \ldots, X_N$ are *independent and identically distributed.* Then

$$Y = \frac{1}{N}\left(X_1 + X_2 + \ldots + X_N\right)$$

has

$$E(Y) = E(X_i); \quad \mathrm{Var}(Y) = \frac{1}{N}\mathrm{Var}(X_i)$$

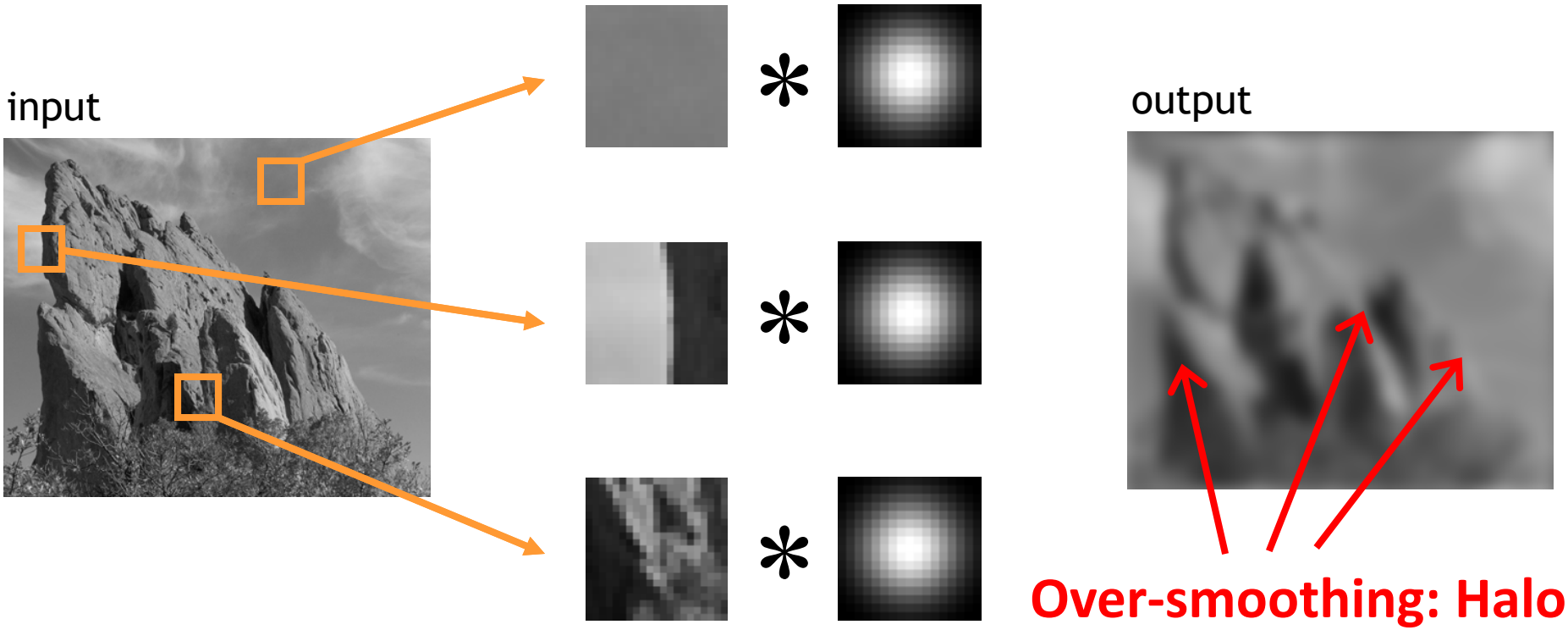
This leads to the moving average filter:

$$y[m] = \frac{1}{N}\sum_{n=0}^{N-1} x[m-n] = (x * h)[m]$$

where

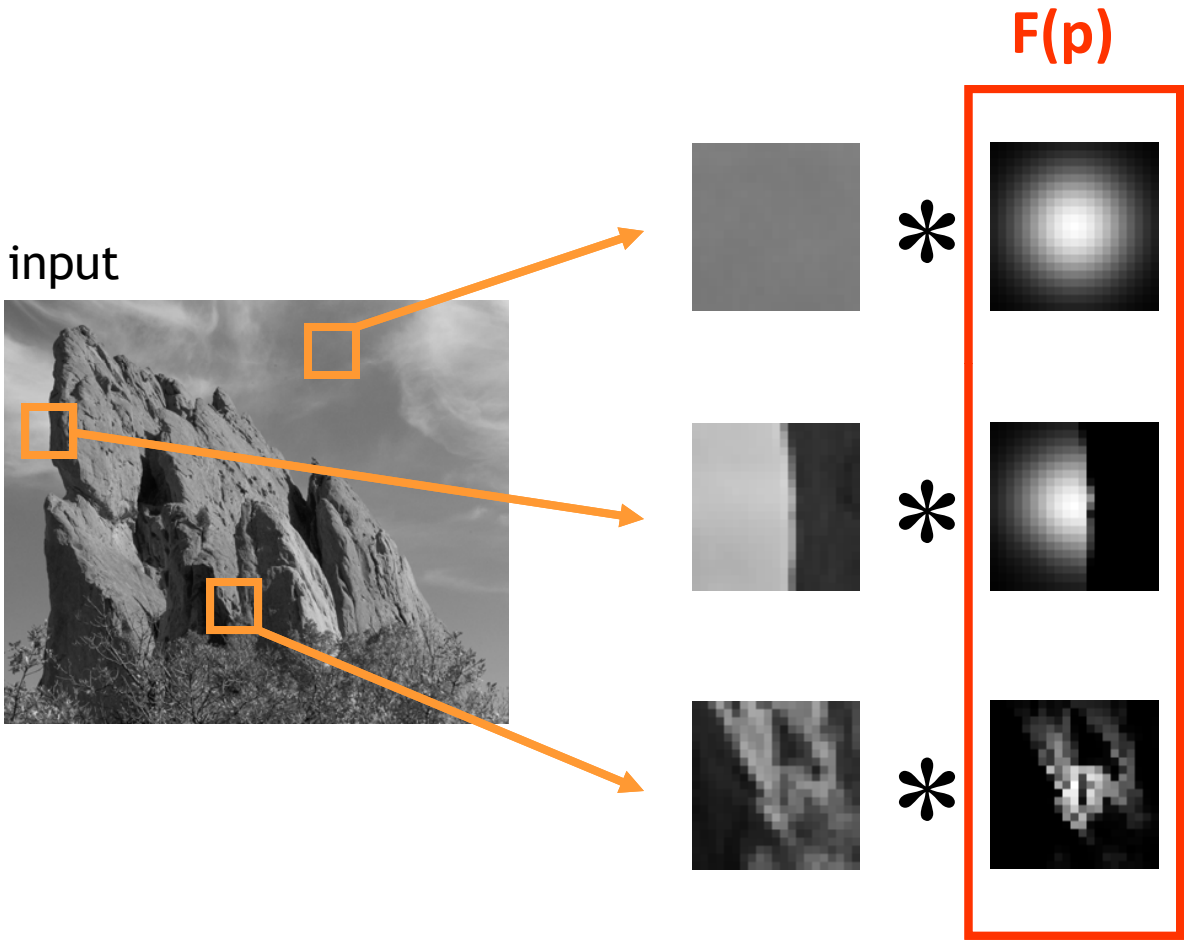
$$h[n] = \begin{cases} \frac{1}{N} & \text{if } n \in \{0, 1, \ldots, N-1\} \\ 0 & \text{else.} \end{cases}$$

Gaussian filter



Same Gaussian kernel everywhere

Bilateral filter: an example of edge-aware



Edge-preserving capability is important

The kernel shape depends on the image content.

Basic notation

- For each pixel **p** (corrupted or not), a pointwise-adaptive function **F(p)** is determined
  - to characterize or approximate the underlying image structure

$$f(I)_{\mathbf{p}} = \sum_{\mathbf{q}} F(\mathbf{p})_{\mathbf{q}}\, I_{\mathbf{q}}$$

- For a given local window, **F(p)**
  - Data-adaptive
  - Shift-variant, nonlinear
  - Spatially varying
- **F(p)** – a key factor

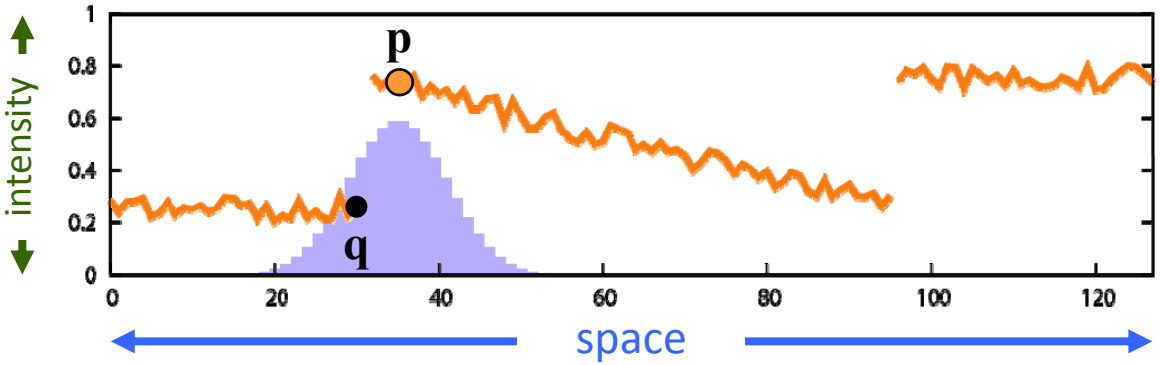
for different performance

Illustration in 1D example: Gaussian blur

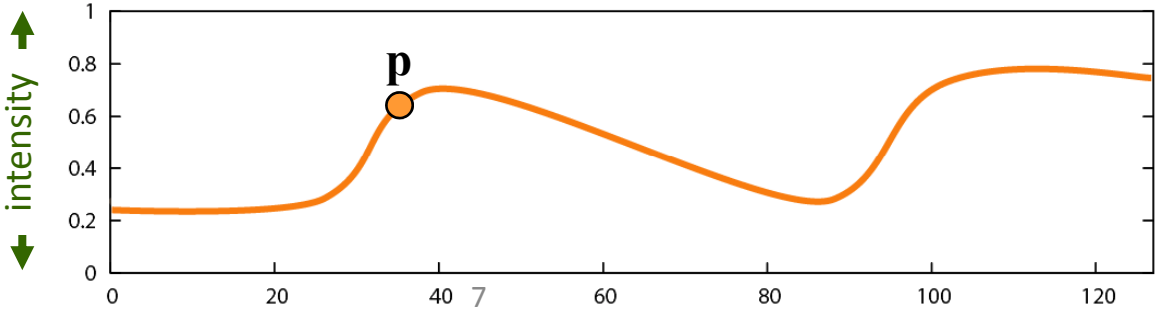
$$gb(I)_{\mathbf{p}} = \sum_{\mathbf{q}} G_{\sigma}(\|\mathbf{p}-\mathbf{q}\|) I_{\mathbf{q}}$$

space

- weighted average of neighbors
- depends only on spatial distance
- no **edge term**



Input



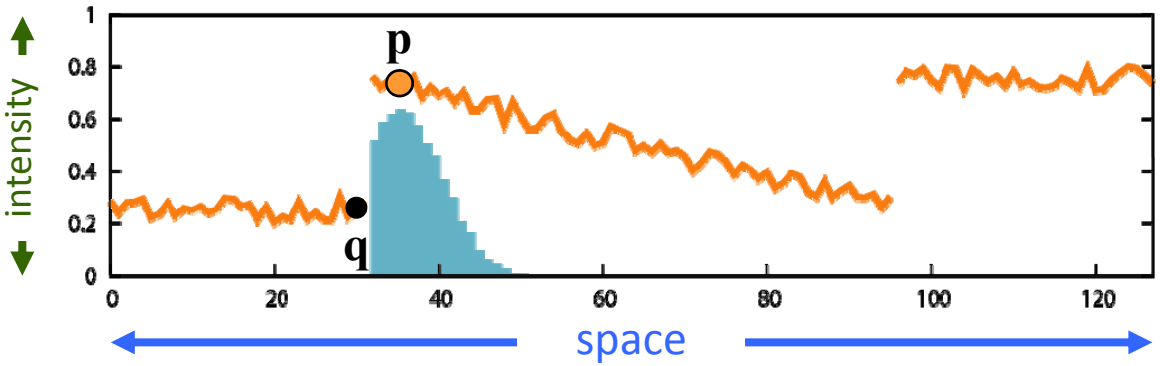
Output

Illustration in 1D example: Bilateral filter

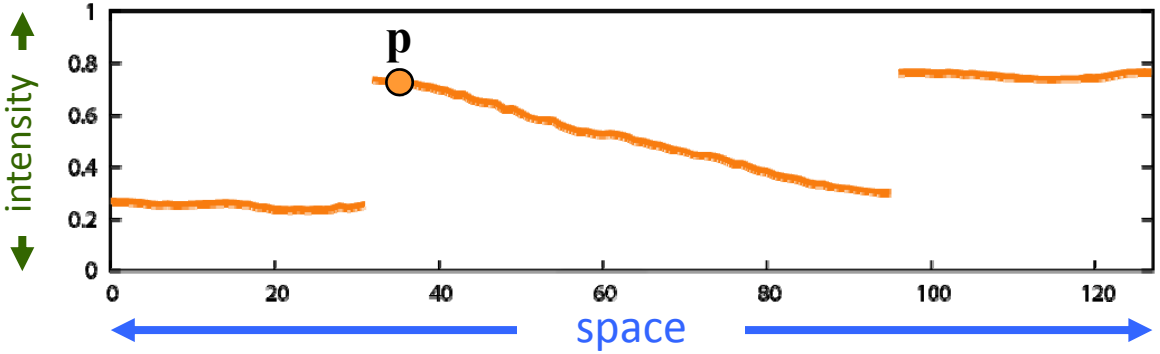
$$bf(I)_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) I_{\mathbf{q}}$$

space range

- weighted average of neighbors
- depends on **spatial** and **range** difference



Input



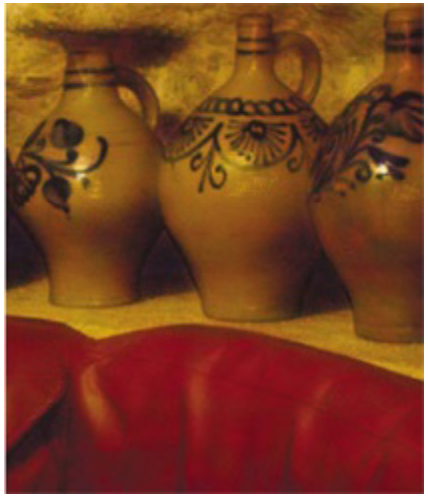
Output

Variants of Bilateral Filter: Joint/Cross Filter

- Idea: Range kernel is computed using a guide (**CLEAN**) image

$$BF[A]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|) G_{\sigma_r}(| B_{\mathbf{p}} - B_{\mathbf{q}} |) A_{\mathbf{q}}$$

BF[A]: Output image



B: Guide image

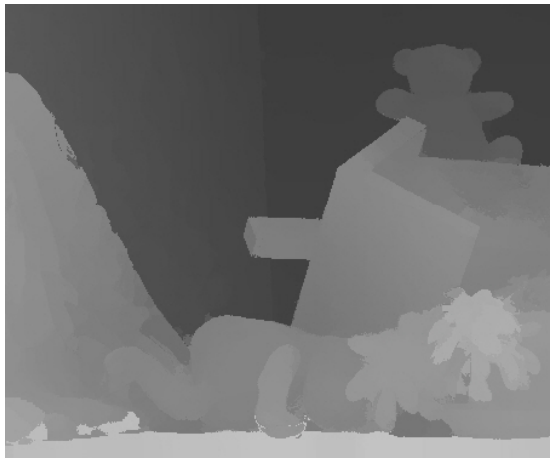
A: Input noisy image

Variants of Bilateral Filter: Joint/Cross Filter

- Idea: Range kernel is computed using a guide (**CLEAN**) image

$$BF[A]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|) G_{\sigma_r}(| B_{\mathbf{p}} - B_{\mathbf{q}} |) A_{\mathbf{q}}$$

BF[A]: Output depth



B: Guide image

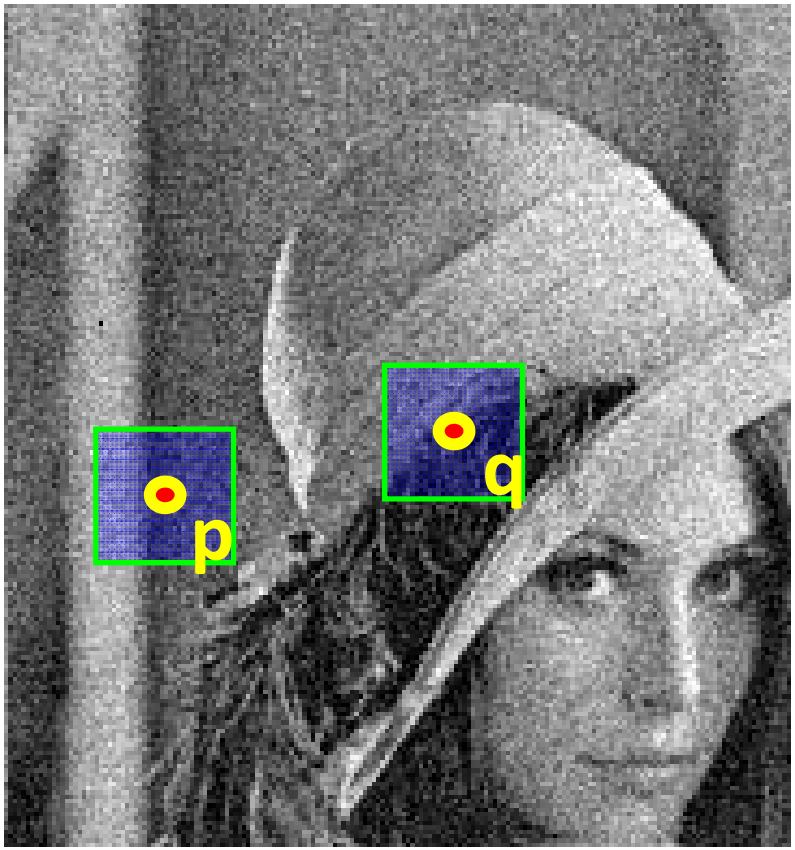
A: Input depth

Variants of Bilateral Filter: Non-local Mean Filter

- **Key idea**: Compute 'Similarity' with patches
  - NL-Means: Averages neighbors with similar neighborhoods
  - Bilateral: Averages neighbors with similar intensities;

$$NLM\ [A]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_r}\left(|\vec{V}_{\mathbf{p}} - \vec{V}_{\mathbf{q}}|^2\right) A_{\mathbf{q}}$$

$|\vec{V}_{\mathbf{p}} - \vec{V}_{\mathbf{q}}|^2$ Vector distance of two patches centered p and q



- Noisy source image:

- Bilateral Filter

(better, but similar ‘stairsteps’:

- NL-Means:

Sharp,

Low noise,

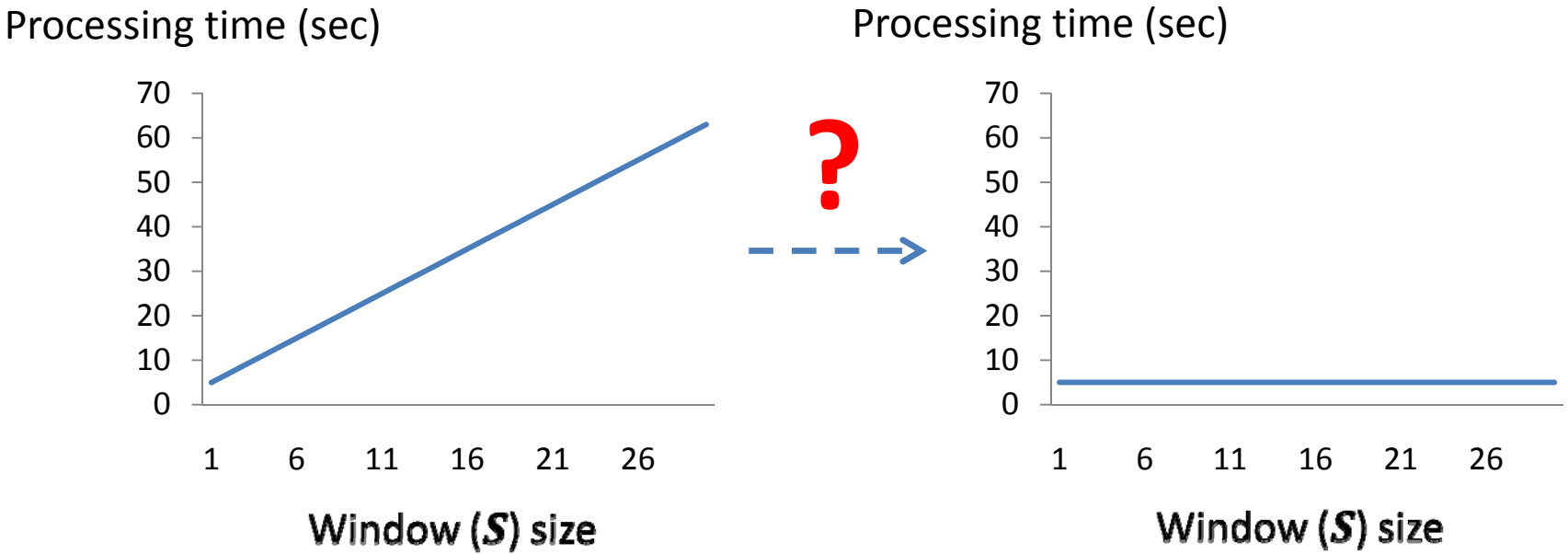
Few artifacts.

Acceleration of Bilateral Filter: O(1) time

- **O(1) time algorithm?**

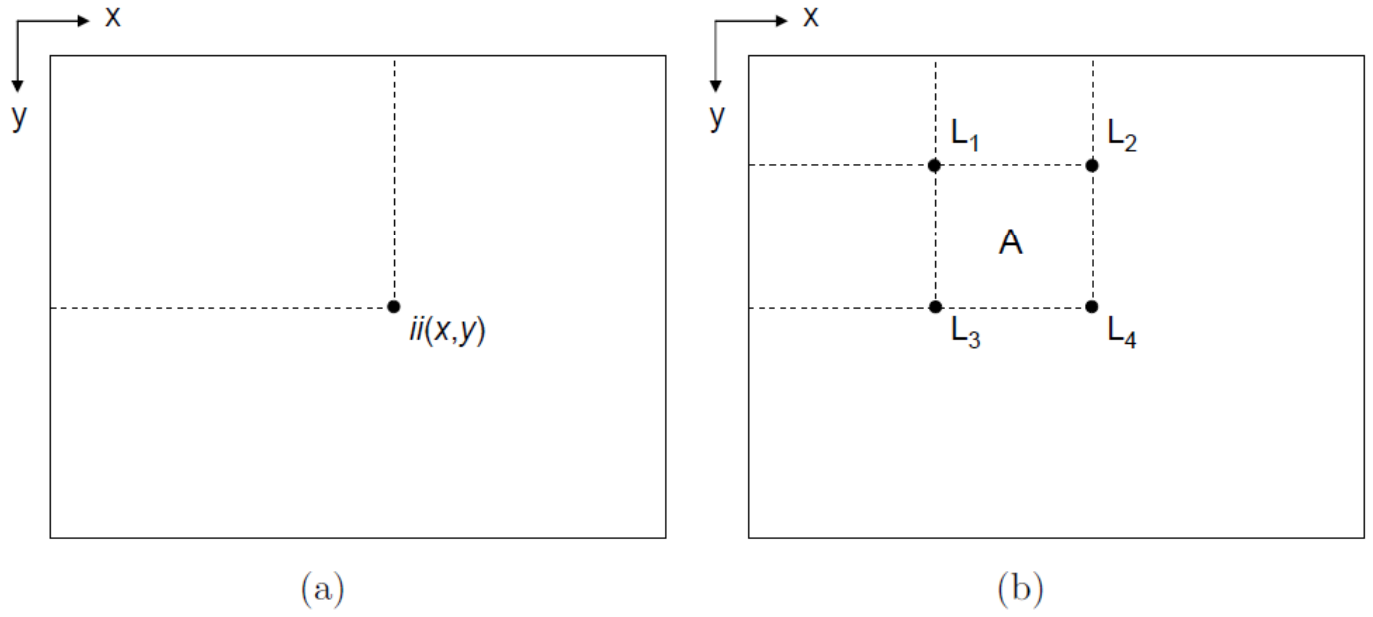
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)\, I_{\mathbf{q}}$$

Non-linear weight!



Brute force algorithm **O(1) time algorithm**

Save computation by exploiting redundancy and sharing



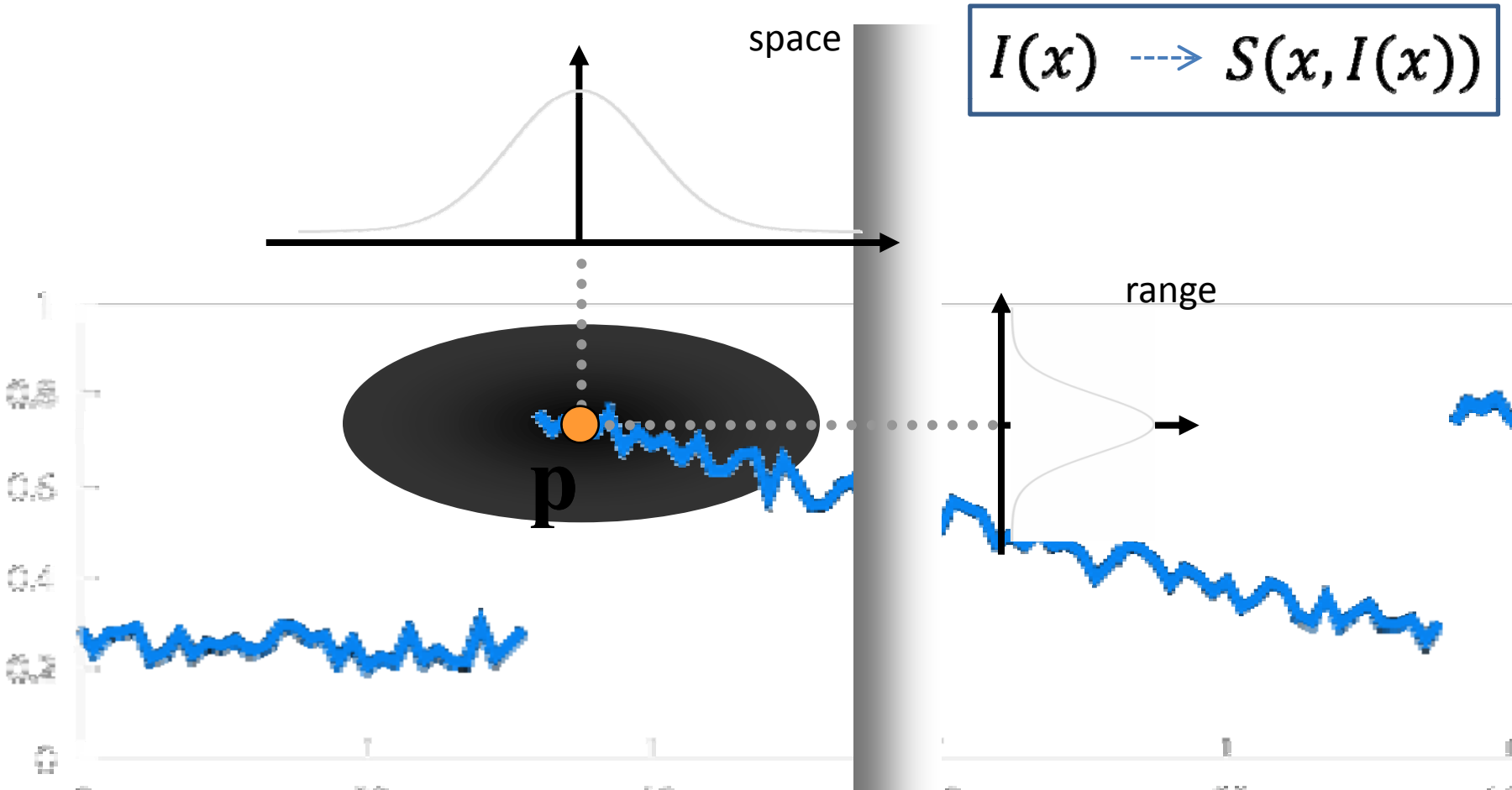
(a) (b)

Figure 1: Integral image representation. (a) integral image and (b) region A can be computed using the following four array references: $L_4 + L_1 - (L_2 + L_3)$.

Acceleration of Bilateral Filter: O(1) time

- Bilateral Grid [ECCV 2006, IJCV 2008]

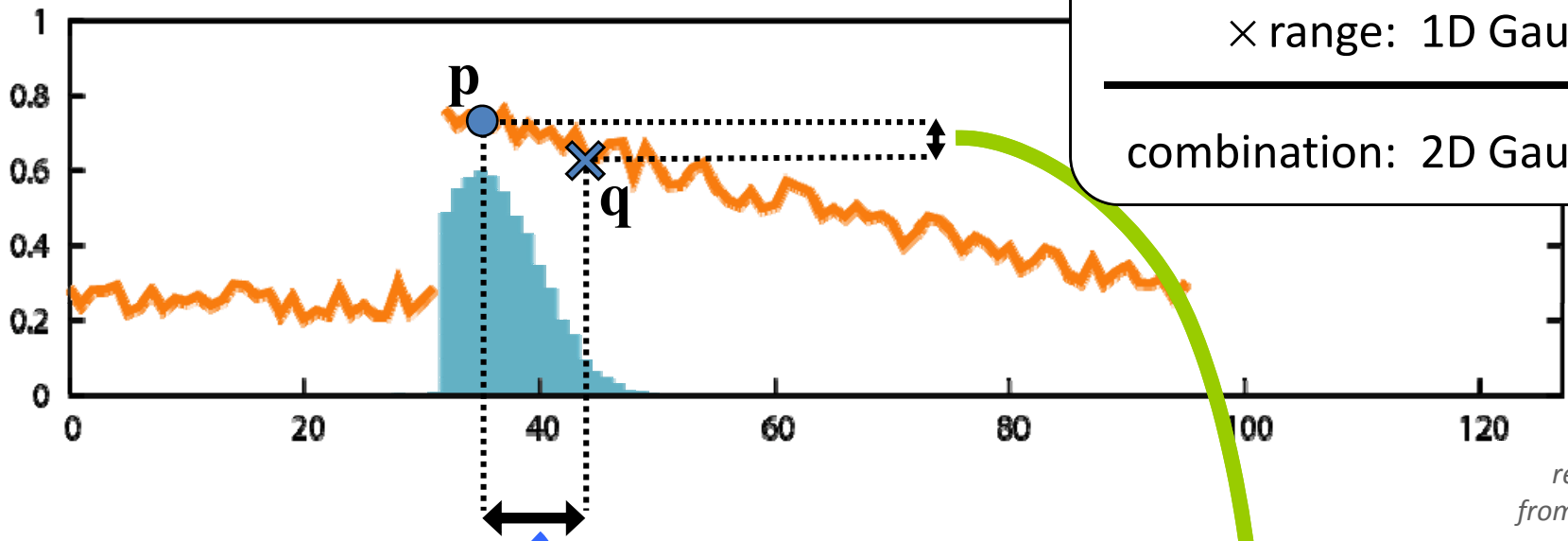
Key idea:
Nonlinear filtering → Linear filtering in a **higher** dimension



Example on 1D Gray Signal

Bilateral Filtering on 1D dimension
= Gaussian Linear Filtering on 2D dimension

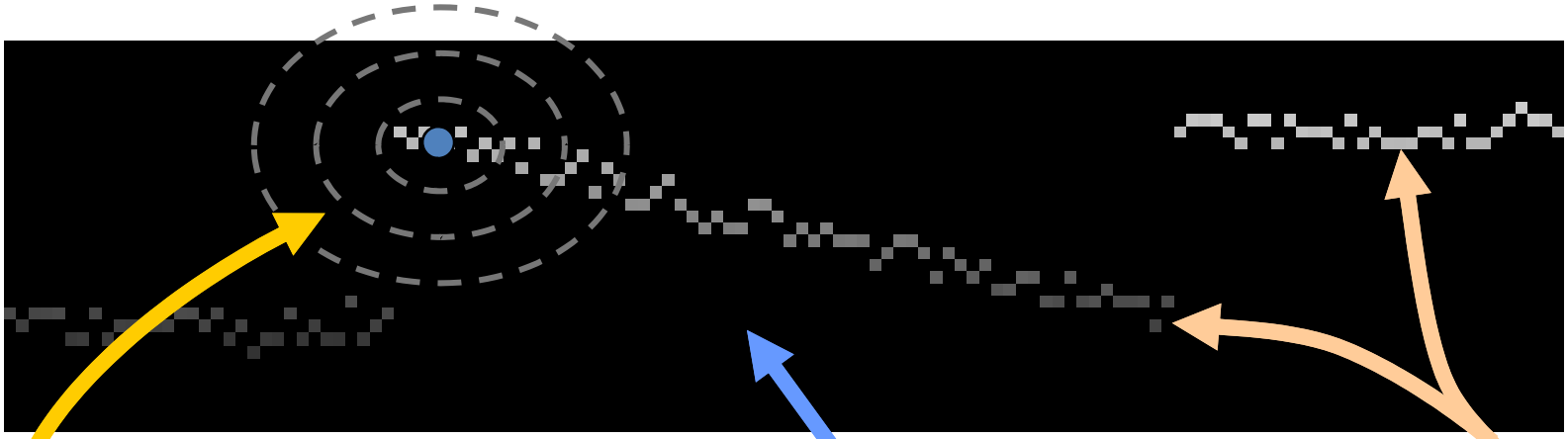
space: 1D Gaussian
× range: 1D Gaussian
---
combination: 2D Gaussian



reproduced from [Paris06]

$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{bf}} I_{\mathbf{p}}^{\mathrm{bf}} \\ W_{\mathbf{p}}^{\mathrm{bf}} \end{pmatrix} = \sum_{\mathbf{q}\in\mathcal{S}} \underbrace{G_{\sigma_\mathrm{s}}(\|\mathbf{p}-\mathbf{q}\|)}_{\text{space}} \underbrace{G_{\sigma_\mathrm{r}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)}_{\text{range}} \begin{pmatrix} I_{\mathbf{q}} \\ 1 \end{pmatrix}$$

Example on 1D Gray Signal



sum all values

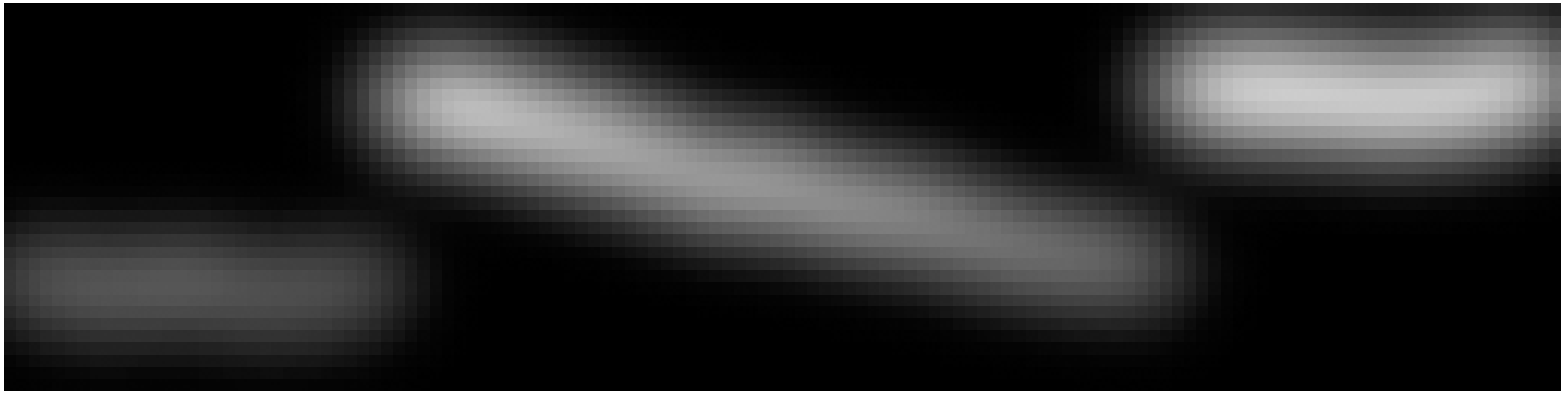
black = zero

reproduced from [Paris06]

$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{bf}} I_{\mathbf{p}}^{\mathrm{bf}} \\ W_{\mathbf{p}}^{\mathrm{bf}} \end{pmatrix} = \sum_{(\mathbf{q},\zeta)\in\mathcal{S}\times\mathcal{R}} \text{space-range Gaussian} \begin{pmatrix} I_{\mathbf{q}} \\ 1 \end{pmatrix}$$

sum all values multiplied by kernel ⇨ convolution

Example on 1D Gray Signal



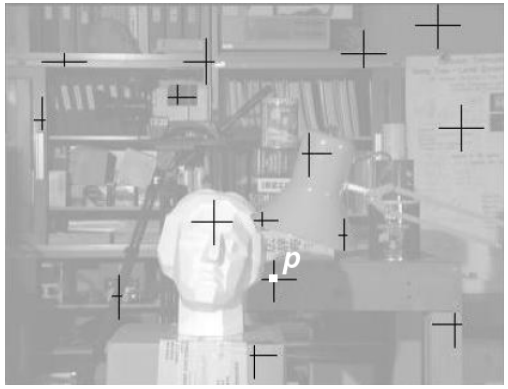
result of the convolution

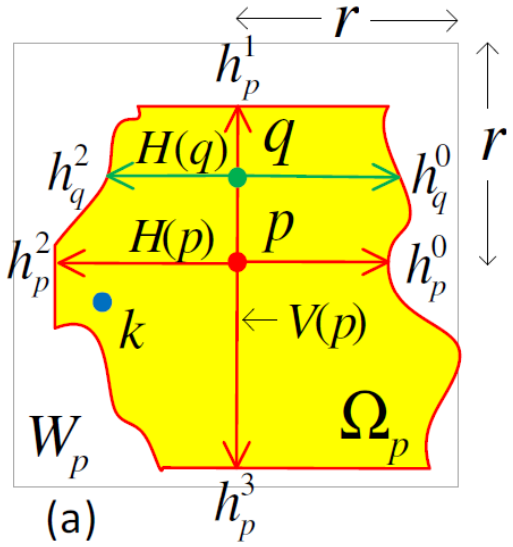
reproduced from [Paris06]

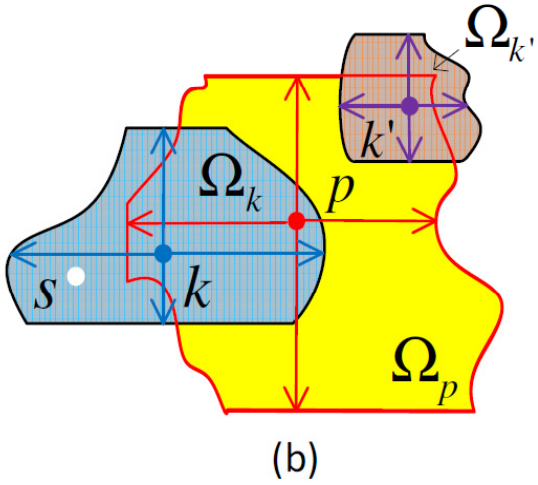
$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{bf}} I_{\mathbf{p}}^{\mathrm{bf}} \\ W_{\mathbf{p}}^{\mathrm{bf}} \end{pmatrix} = \sum_{(\mathbf{q},\zeta)\in\mathcal{S}\times\mathcal{R}} \text{space-range Gaussian} \begin{pmatrix} I_{\mathbf{q}} \\ 1 \end{pmatrix}$$

Cross-based Local Multipoint Filter (CLMF)

- **Preprocessing step**: decide for each pixel *p*, four varying support arm lengths
- **Multipoint estimation**: compute the estimates for all points in a local support
- **Aggregation**: fuse a number of multipoint estimates available for each point

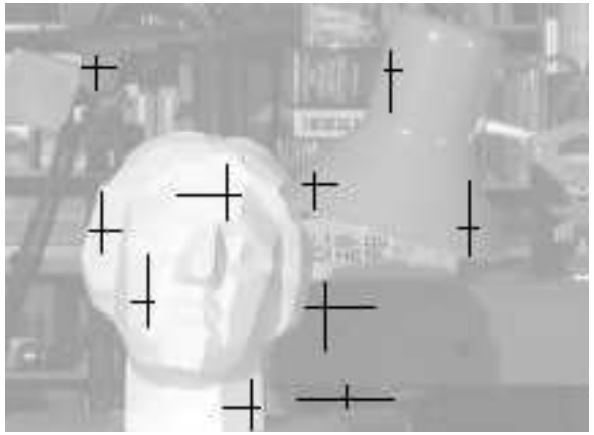


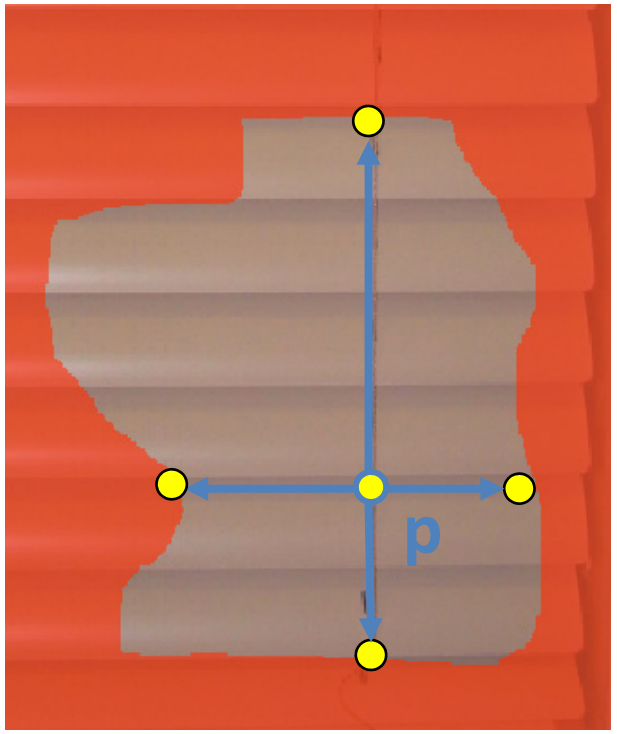




J. Lu, et al., "Cross-based local multipoint filtering," *CVPR*, 2012.

Scale decision (a simple example)





F(p)

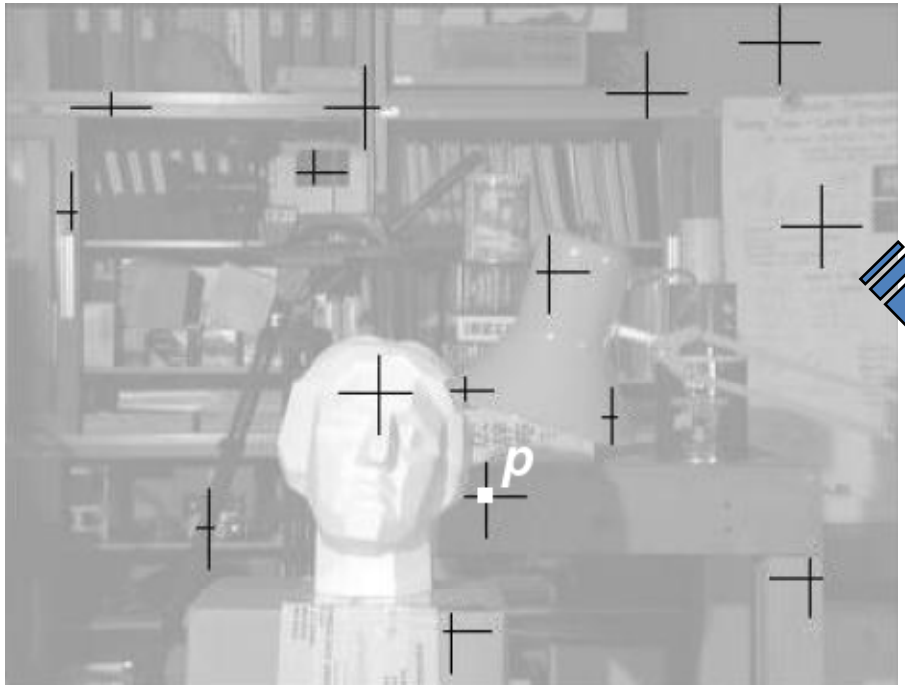
- Piecewise constant assumption (zero order)
  - Search for the largest span r^* (L is a preset maximum search range)

$$\forall r \leq r^*, \delta(p_r, p) = 1, \text{ and } \delta(p_{r^*+1}, p) = 0 \text{ (if } r^* < L)$$

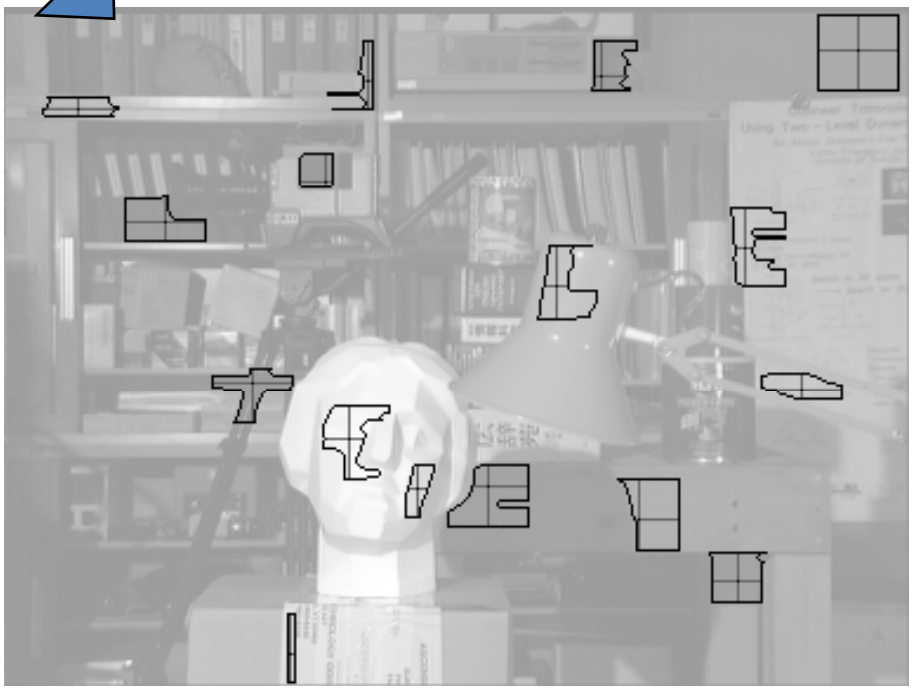
$$\delta(p_1, p_2) = \begin{cases} 1, & \max\limits_{c \in \{R,G,B\}} \left(|I_c(p_1) - I_c(p_2)| \right) \leq \tau \\ 0, & \text{otherwise,} \end{cases}$$

[Zhang et al., TCSVT'09]

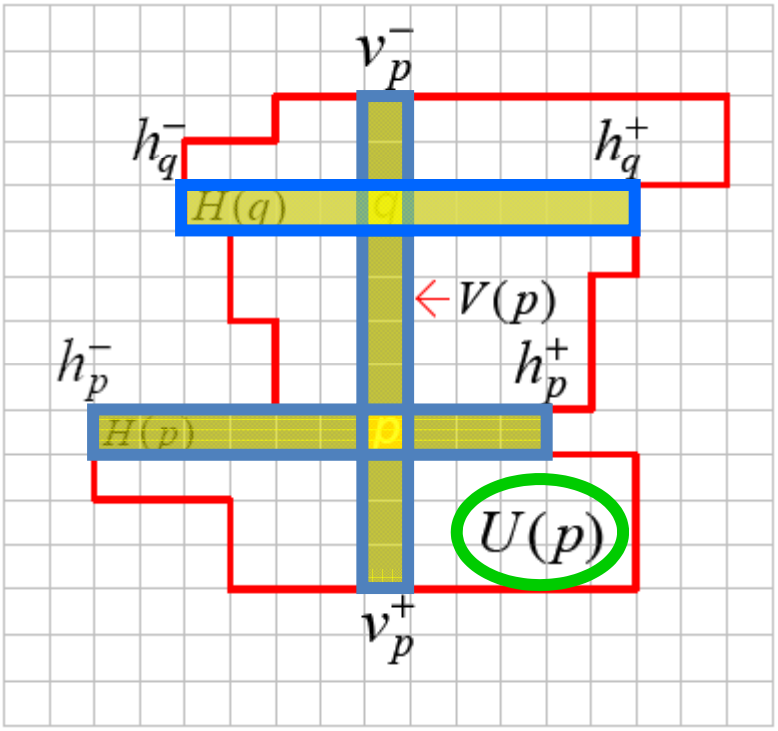
From compact cross to irregular full support

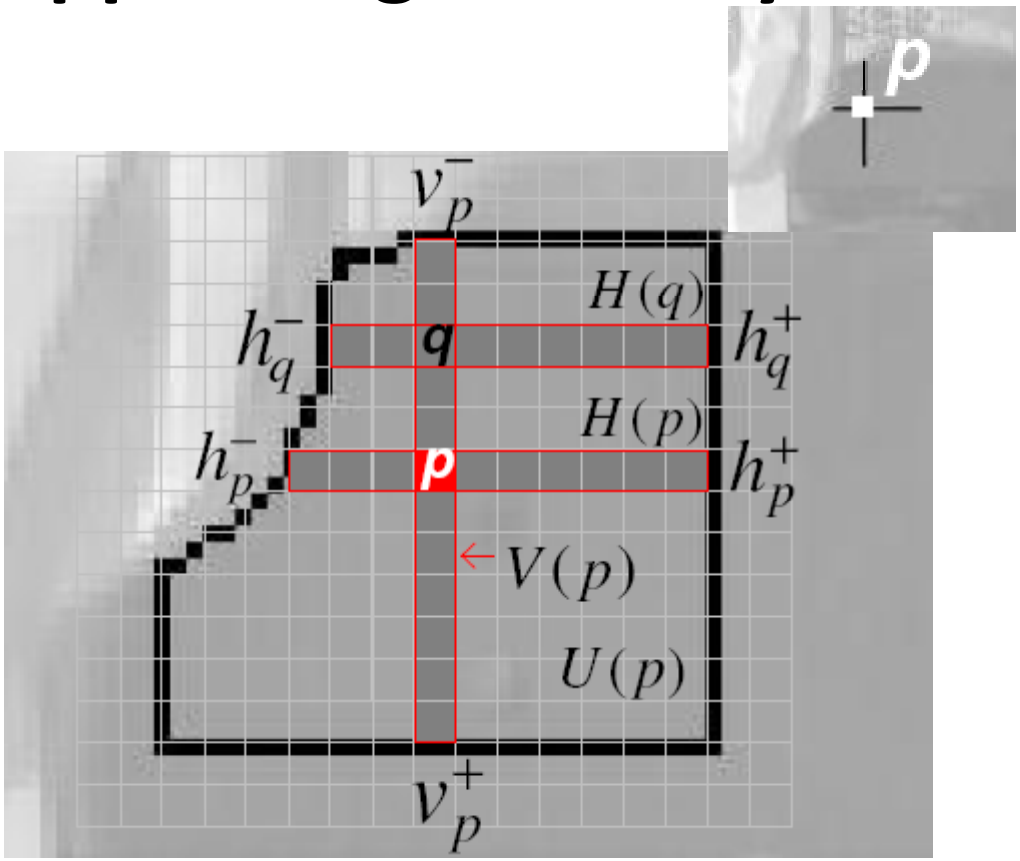


How?



Construct a full support region easily





Two orthogonal line segments

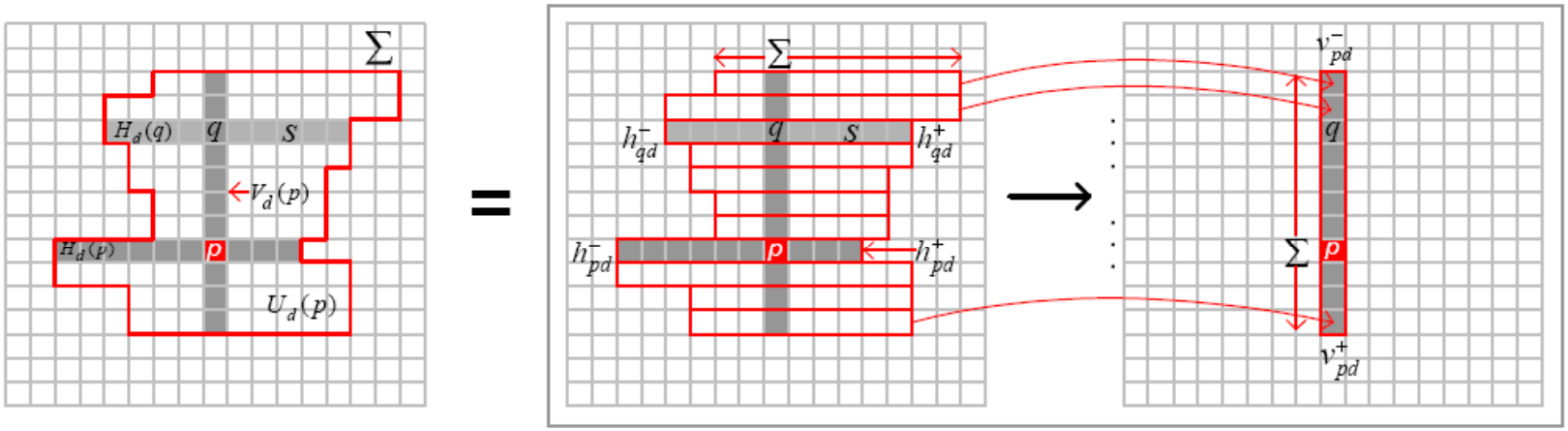
$$\begin{cases} H(p) = \{(x,y) \mid x \in [x_p - h_p^-, x_p + h_p^+], y = y_p\} \\ V(p) = \{(x,y) \mid x = x_p, y \in [y_p - v_p^-, y_p + v_p^+]\} \end{cases}$$

Area integral

$$U(p) = \bigcup_{q \in V(p)} H(q)$$

Fast integral over shape-adaptive regions

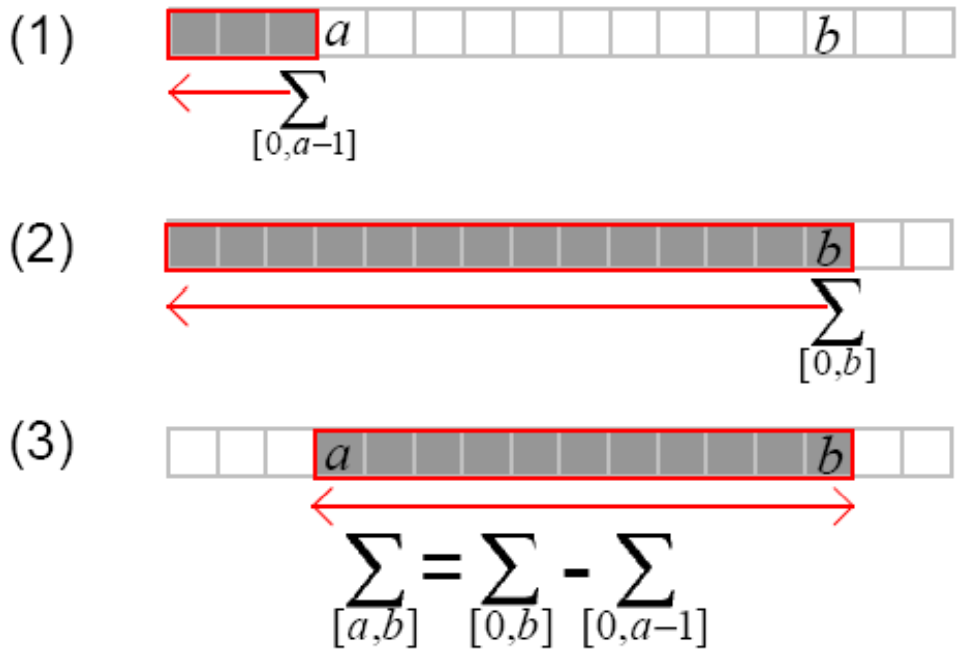
- Two separable/orthogonal 1D integration steps
  - Because $\boldsymbol{H_d(q)}$ is invariant for all anchor pixels, e.g., ***p*** and ***q***
  - Complexity linear to the span of segments, not the area



$$E_d(p) = \sum_{s \in U_d(p)} e_d(s) = \sum_{q \in V_d(p)} \Big(\sum_{s \in H_d(q)} e_d(s) \Big) = \sum_{q \in V_d(p)} E_d^H(q)$$

Integral images for 1D horizontal/vertical step

- Constant load for segments of varying lengths
- 5x – 15x faster than the straightforward computation



Local Filter

- ✓ Explicit kernel
- ✓ Efficient
- ✓ Artifacts, e.g., halo
- ✓ Limited applications

Optimization

- ✓ Implicit kernel, Ax=b
- ✓ Various applications
- ✓ No artifacts
- ✓ Not efficient

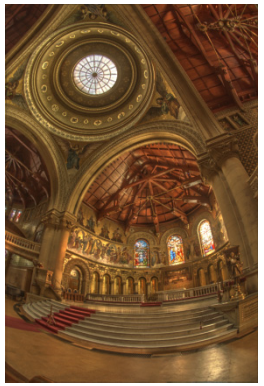
Edge-Preserving Smoothing (EPS) Operators

Fast Global Smoothing (FGS)

- ✓ Efficient
- ✓ No artifacts
- ✓ Various applications

Poisson image editing

Multi-scale tone manipulation



HDR tone mapping

Colorization

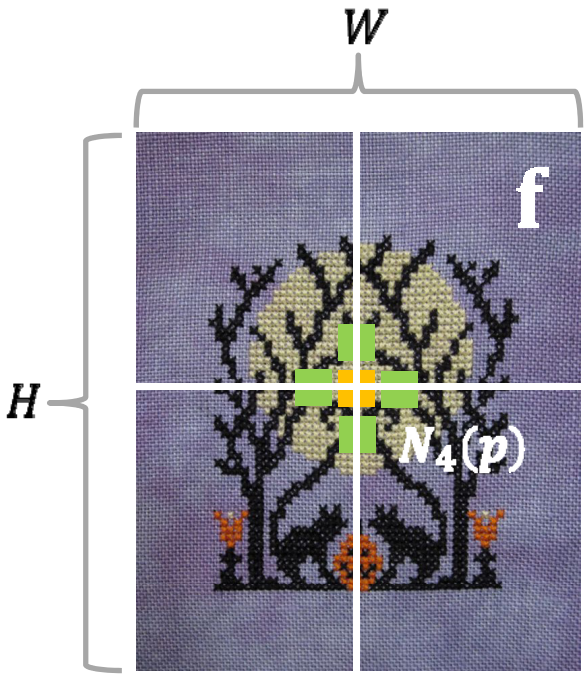
D. Min, et al., "Fast global image smoothing based on weighted least squares," *TIP*, 2014.

Image Smoothing using Weighted Least Squares (1/2)

- **Optimization on 2D signal (image)**
  - Given an input image f and a guidance image g, A desired output u **minimizes** the following

✓ No artifacts
✓ Various applications
✓ Not efficient

$$J(u) = \sum_p \left((u_p - f_p)^2 + \lambda \sum_{q \in \mathcal{N}(p)} w_{p,q}(g)(u_p - u_q)^2 \right) \Rightarrow (\mathbf{I} + \lambda \mathbf{A})\mathbf{u} = \mathbf{f}$$



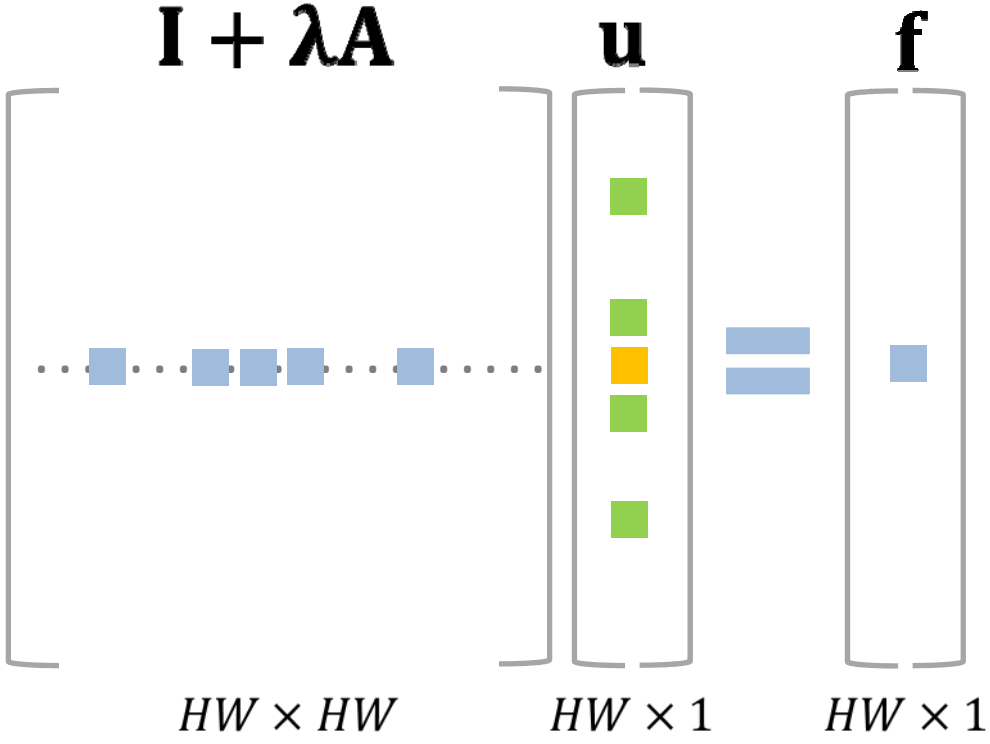


Image Smoothing using Weighted Least Squares (2/2)

- Optimization on 2D signal

$$\mathbf{A}(m,n) = \begin{cases} \sum_{l \in \mathcal{N}(m)} w_{m,l}(g) & n = m \\ -w_{m,n}(g) & n \in \mathcal{N}(m) \\ 0 & \text{otherwise} \end{cases}$$

I:Identity matrix
A: Spatially-varying Laplacian matrix

$$(\mathbf{I} + \lambda \mathbf{A})\mathbf{u} = \mathbf{f} \quad \Rightarrow ? \quad \mathbf{u}(m) = ((\mathbf{I} + \lambda \mathbf{A})^{-1}\mathbf{f})(m)$$

Q: How to solve this? A: Sparse matrix solver

State-of-the-arts solver: multi-level and multigrid preconditioning method

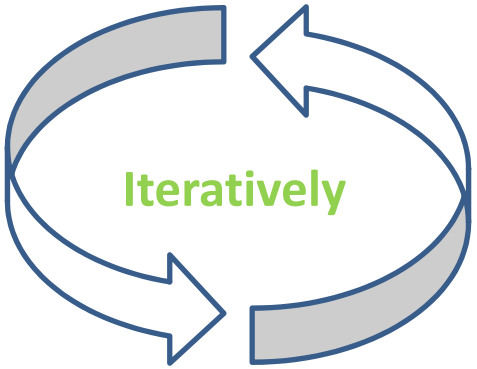
But, these approaches are still very slow.

Energy Decomposition: Horizontal and Vertical Lines

Our key idea: Update solution with **two separable** 1D global smoother **iteratively**

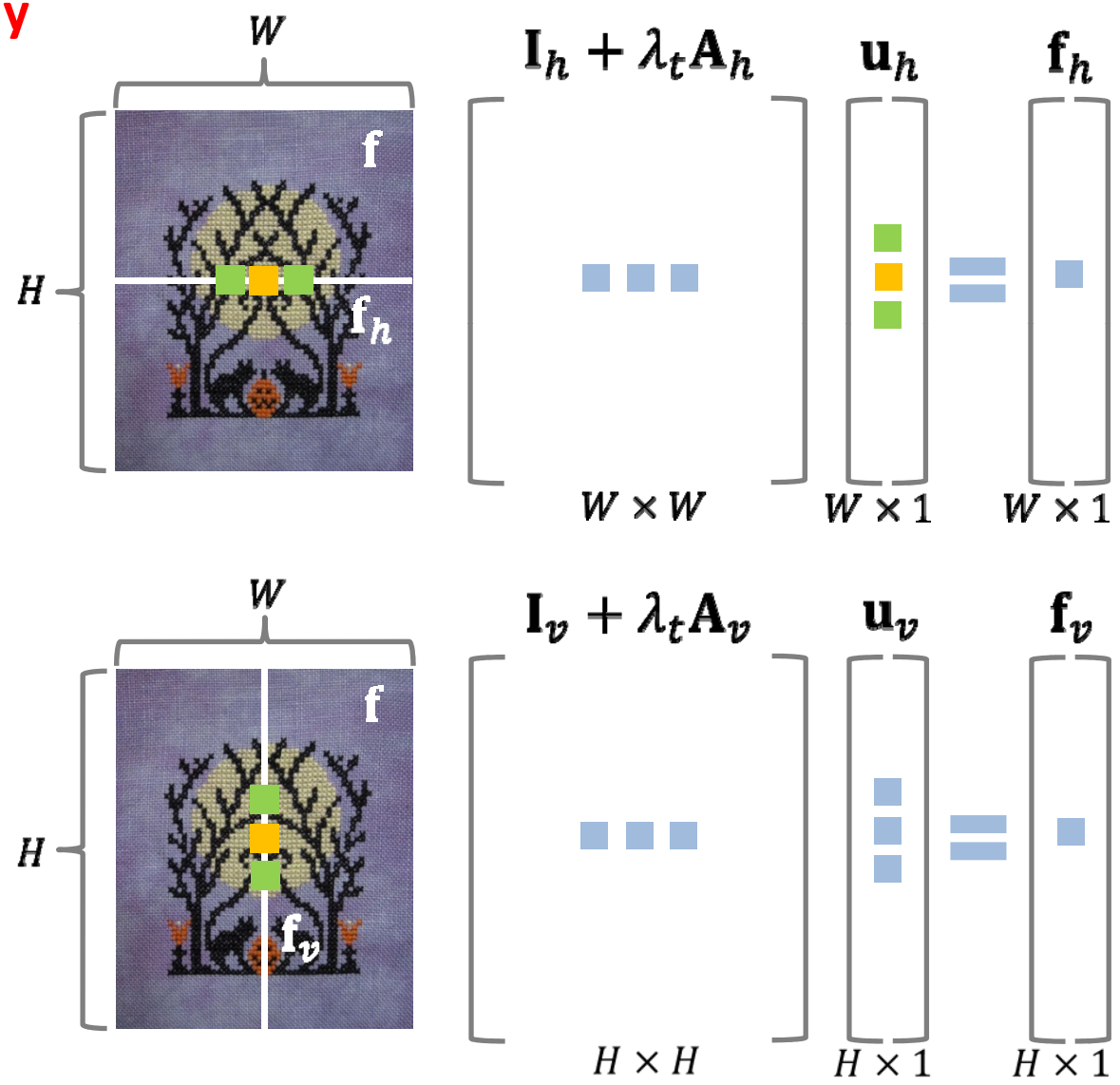
Horizontal 1D global smoother

$$(\mathbf{I}_h + \lambda_t \mathbf{A}_h)\mathbf{u}_h = \mathbf{f}_h$$



$$(\mathbf{I}_v + \lambda_t \mathbf{A}_v)\mathbf{u}_v = \mathbf{f}_v$$

Vertical 1D global smoother



Energy Decomposition: 1D Smoother (1/2)

- **For horizontal line**
  - Given a 1D horizontal f_x and a 1D guidance signal g, A desired output u **minimizes** the following

$$\sum_x \left((u_x^h - f_x^h)^2 + \lambda_t \sum_{i \in \mathcal{N}_h(x)} w_{x,i}(g^h)(u_x^h - u_i^h)^2 \right)$$

$$\Rightarrow \quad (\mathbf{I}_h + \lambda_t \mathbf{A}_h)\mathbf{u}_h = \mathbf{f}_h$$

$\mathbf{A}_h$: Three-point Laplacian matrix (Tridiagonal Matrix)

$$\begin{bmatrix} b_0 & c_0 & 0 & \cdots & 0 \\ \ddots & \ddots & \ddots & 0 & 0 \\ 0 & a_x & b_x & c_x & 0 \\ 0 & 0 & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & a_{W-1} & b_{W-1} \end{bmatrix} \begin{bmatrix} u_0^h \\ \vdots \\ u_x^h \\ \vdots \\ u_{W-1}^h \end{bmatrix} = \begin{bmatrix} f_0^h \\ \vdots \\ f_x^h \\ \vdots \\ f_{W-1}^h \end{bmatrix}$$

$$\begin{aligned} a_x &= \lambda_t \mathbf{A}_h(x, x-1) = -\lambda_t w_{x,x-1}, \\ b_x &= 1 + \lambda_t \mathbf{A}_h(x, x) = 1 + \lambda_t (w_{x,x-1} + w_{x,x+1}), \\ c_x &= \lambda_t \mathbf{A}_h(x, x+1) = -\lambda_t w_{x,x+1}. \end{aligned}$$

Energy Decomposition: 1D Smoother (2/2)

- **Solving a linear system with tridiagonal matrix**
  - Using Gaussian elimination algorithm with a $O(W)$ time complexity (W: 1D signal size)

$$\begin{bmatrix} b_0 & c_0 & 0 & \cdots & 0 \\ \ddots & \ddots & \ddots & 0 & 0 \\ 0 & a_x & b_x & c_x & 0 \\ 0 & 0 & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & a_{W-1} & b_{W-1} \end{bmatrix} \begin{bmatrix} u_0^h \\ \vdots \\ u_x^h \\ \vdots \\ u_{W-1}^h \end{bmatrix} = \begin{bmatrix} f_0^h \\ \vdots \\ f_x^h \\ \vdots \\ f_{W-1}^h \end{bmatrix}$$

1) Forward process

$$\tilde{c}_0 = c_0/b_0 \text{ and } \tilde{f}_0^h = f_0^h/b_0$$

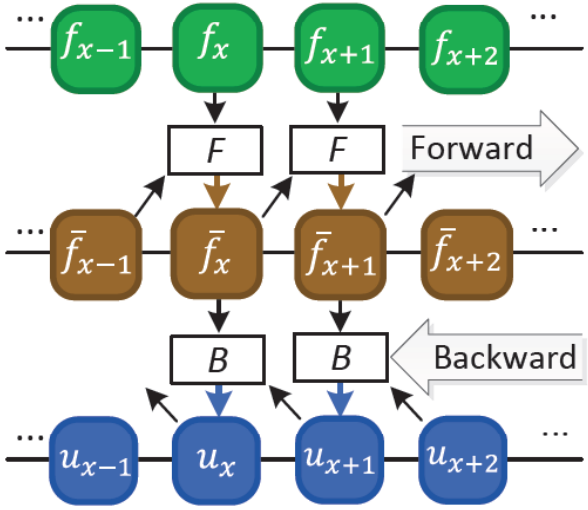
$$\begin{aligned} \tilde{c}_x &= c_x/(b_x - \tilde{c}_{x-1}a_x) \\ \tilde{f}_x^h &= (f_x^h - \tilde{f}_{x-1}^h a_x)/(b_x - \tilde{c}_{x-1}a_x) \end{aligned}, \quad (x = 1, \cdots, W-1)$$

2) Backward process

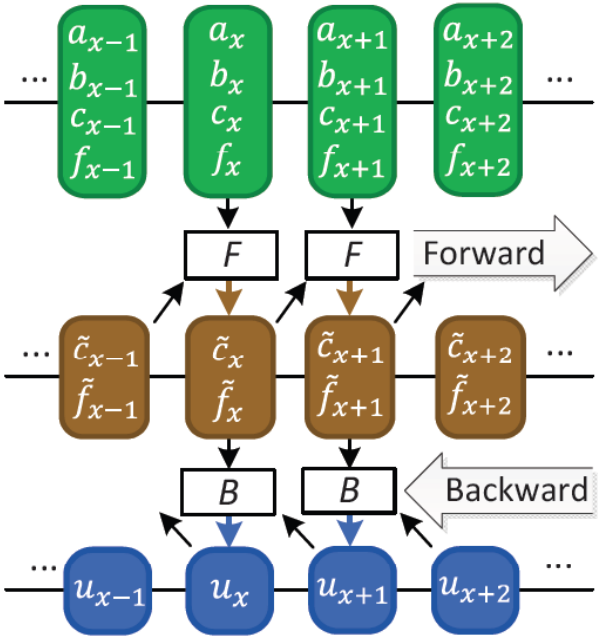
$$u_x^h = \tilde{f}_x^h - \tilde{c}_x u_{x+1}^h, \quad (x = W-2, \cdots, 0)$$

Recursive Propagation of FGS

- Local Edge-Preserving (EP) Recursive Filter vs. Our Global Smoother
  - The recursive data propagation scheme looks similar to each other
  - However, the smoothing quality of ours is MUCH better.

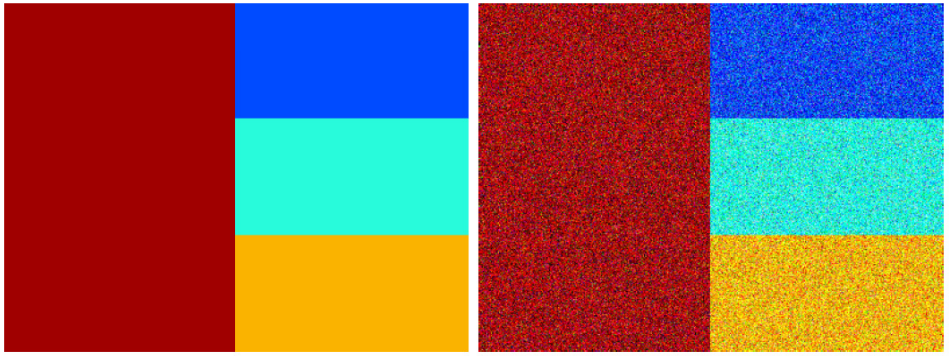


Local EP recursive filters

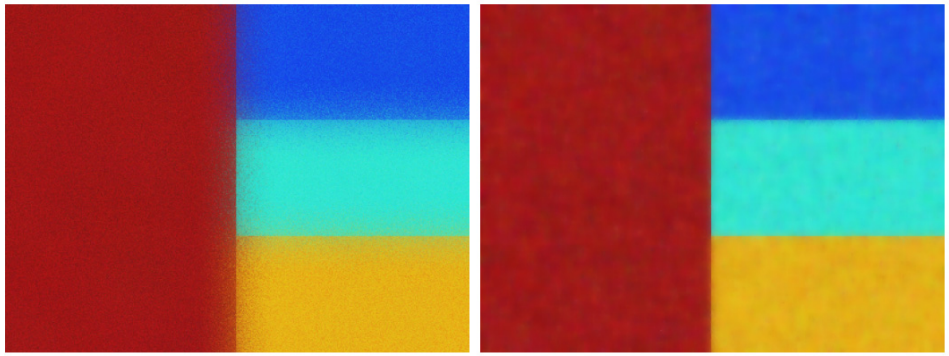


Global Smoother

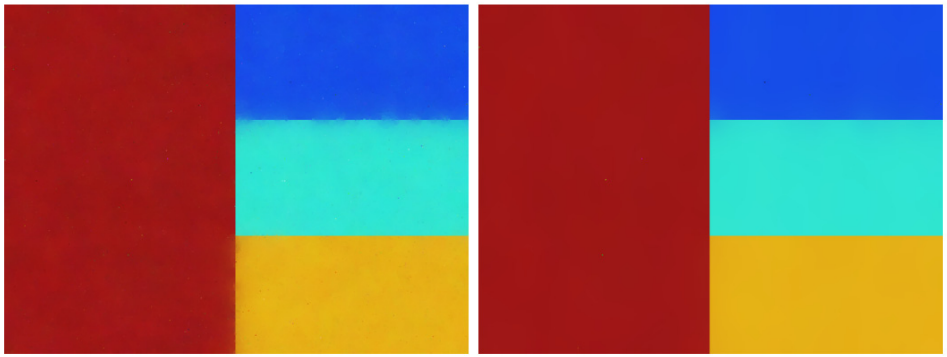
2D Image Smoothing Results (1/2)



(a) Original (b) Noisy input



(c) Guided Filter [8] (d) RF of DT [9]



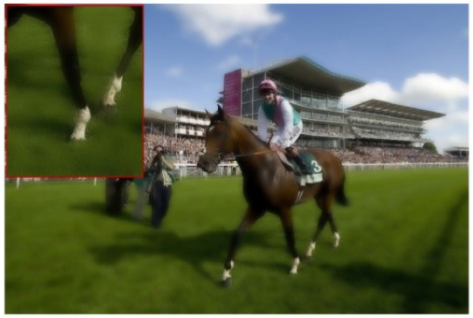
(e) Original WLS [15] (f) Proposed method

1) Smoothing results on highly-texture regions
2) Discontinuities-preserving capability

2D Image Smoothing Results (2/2)

Compare 1) smoothing results on highly-texture regions and 2) discontinuities-preserving capability

(a) Input

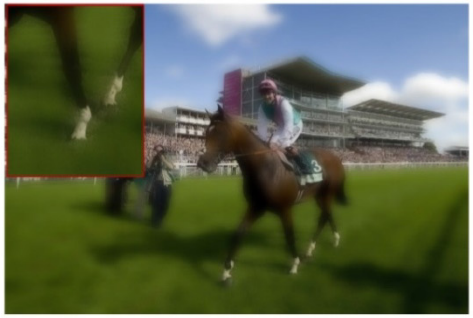


(b) GF ($r = 7$, $\epsilon = 0.1^2$)

(c) RF of DT ($\sigma_s = 50.0$, $\sigma_r = 0.5$)

(d) NC of DT ($\sigma_s = 50.0$, $\sigma_r = 0.5$)

(e) Ours ($\lambda = 25.0^2$, $\sigma_c = 0.035$)



(f) GF ($r = 11$, $\epsilon = 0.2^2$)

(g) RF of DT ($\sigma_s = 50.0$, $\sigma_r = 1.0$)

(h) NC of DT ($\sigma_s = 50.0$, $\sigma_r = 1.0$)

Extension to Sparse Data Interpolation

- **Start with 1D signal including an index function h**
  - h_p: indicates whether a pixel p has a valid input.

$$J(u) = \sum_p \left((u_p - f_p)^2 + \lambda \sum_{q \in \mathcal{N}(p)} w_{p,q}(g)(u_p - u_q)^2 \right)$$

h

$$\sum_p \left(h_p(u_p - f_p)^2 + \lambda \sum_{q \in \mathcal{N}_4(p)} w_{p,q}(g)(u_p - u_q)^2 \right)$$

Solution

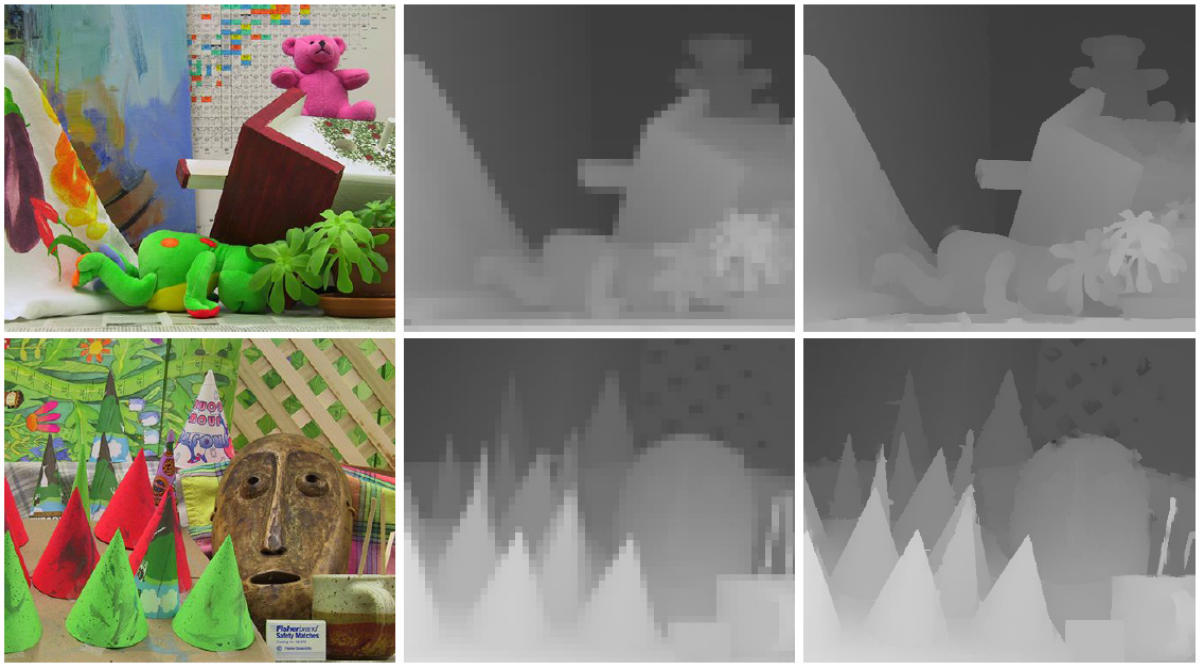
$$(\mathbf{H} + \lambda \mathbf{A})\mathbf{u} = \mathbf{H}\mathbf{f}$$

But, this linear system may be **unstable** since the sparse input data f_p is sparse (e.g. some rows or columns of $\mathbf{H}$ may be all zeros)

Alternatively, we use $$\mathbf{u}(m) = \frac{((\mathbf{I} + \lambda \mathbf{A})^{-1}\mathbf{f})(m)}{((\mathbf{I} + \lambda \mathbf{A})^{-1}\mathbf{h})(m)}$$

Results of Sparse Data Interpolation

- Depth Up-sampling



(a) Color image (b) Input depth (c) Upsampled depth

Algo.	*Tsukuba*		*Venus*		*Teddy*		*Cone*	
	all	disc	all	disc	all	disc	all	disc
Input	10.4	46.2	3.26	36.6	11.9	35.5	14.7	36.4
2-D JBU	9.04	40.4	2.04	22.1	14.0	37.6	14.7	34.8
3-D JBU	7.89	35.0	1.67	17.8	10.7	30.6	12.1	30.0
WMF	4.35	20.2	0.61	5.73	9.51	23.7	9.43	19.2
Ours	4.66	18.4	0.33	3.70	6.70	16.9	4.54	10.3

Runtime Comparison

WMF (state-of-the-arts): **0.48 sec**

FGS: **0.02 sec**

Using Aggregated Data Term (1/2)

Our Original Formulation: Per-pixel Data Constraint

$$J(u) = \sum_p \left((u_p - f_p)^2 + \lambda \sum_{q \in \mathcal{N}(p)} w_{p,q}(g)(u_p - u_q)^2 \right)$$

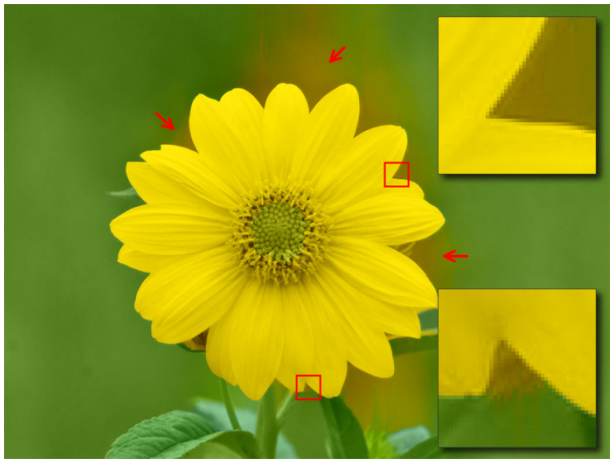
$$\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}$$

$$(\mathbf{I} + \lambda \mathbf{A})\mathbf{u} = \mathbf{f}$$

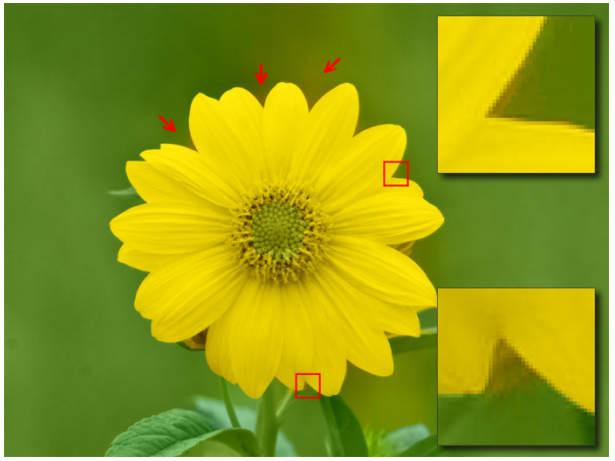
Uniform reliability

Initial input

Input



After 3 iter.



After 10 iter.

Color bleeding artifacts even after many iterations!

Using Aggregated Data Term (2/2)

Our New Formulation: Aggregated Data Constraint

For handling imprecise input

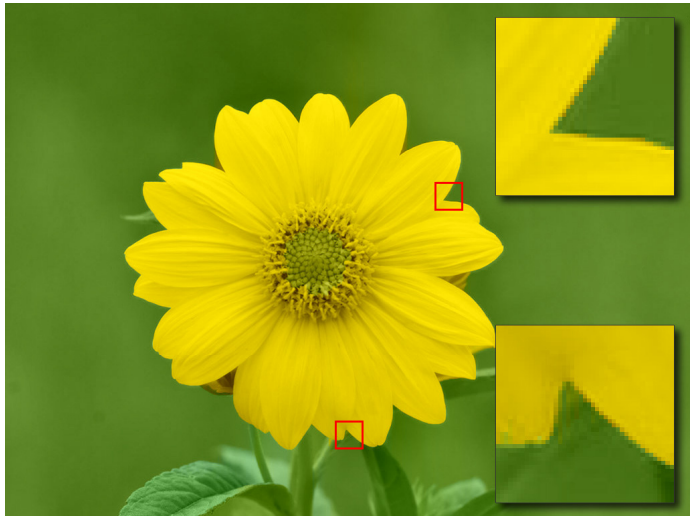
$$\sum_p \left(\sum_{r \in \mathcal{N}_D} c_{p,r}(g)(u_p - f_r)^2 + \lambda \sum_{q \in \mathcal{N}_4} w_{p,q}(g)(u_p - u_q)^2 \right)$$

$$\begin{pmatrix} d_{11} & & & \\ & d_{22} & & \\ & & \ddots & \\ & & & d_{SS} \end{pmatrix}$$

$$(\mathbf{D} + \lambda \mathbf{A})\mathbf{u} = \mathbf{C}\mathbf{f}$$

Adaptive reliability

Regularized input



No color bleeding artifacts!

Handling Imprecise Input in Image Colorization

(a) Input

(b) JBF (50 iter.)

(c) GF (30 iter.)

(d) RF of DT (50 iter.)

(e) Levin et al. 2004

(f) Ours with per-pixel data (3 iter.)

(g) Ours with per-pixel data (10 iter.)

(h) Ours with aggregated data (3 iter.)

(i) Enlarged image of (b)

(j) Enlarged image of (e)

(k) Enlarged image of (f)

(l) Enlarged image of (g)

(m) Enlarged image of (h)

Performance of Proposed Method (1/2)

Smoothing a 1 M pixel RGB image on a single core of a CPU

- ✓ **Local Filter: 0.15s - Guided Filter [He et al. PAMI 2013]**
- ✓ **Local Filter: 0.05s - Domain Transform [Gastal and Oliveira, SIGGRAPH 2011]**
- ✓ **Optimization: 3.3s - WLS method [Farbman et al. SIGGRAPH 2008]**
- ✓ **Our smoother: 0.10s**

Single core CPU: 3.4 GHz CPU, 8G memory

Efficiency: Our smoother ≈ Local filter >> Optimization

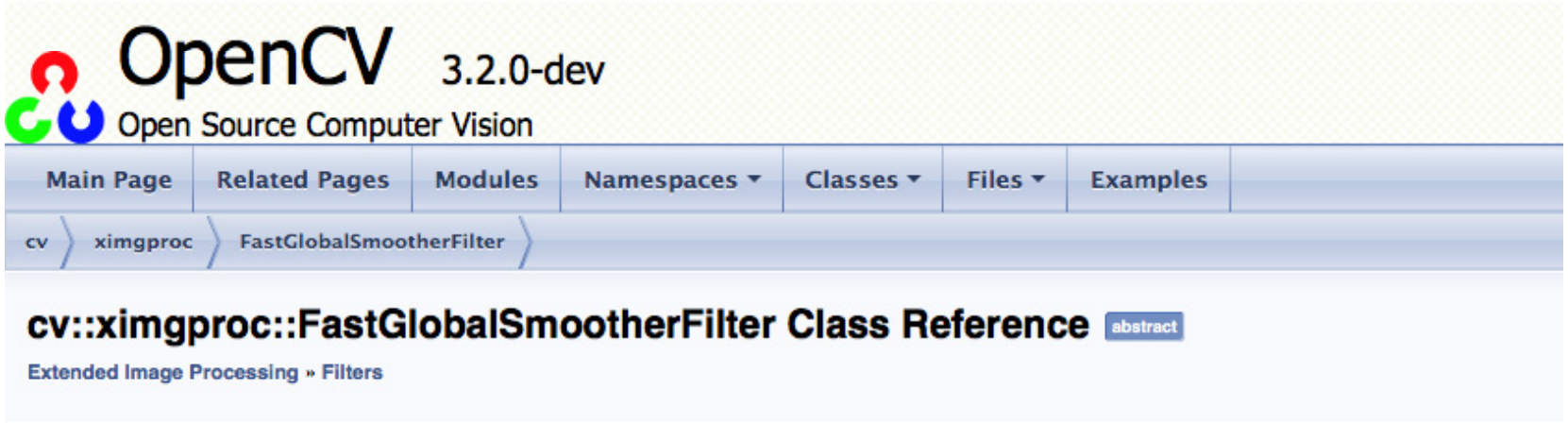
Quality: Our smoother ≈ Optimization >> Local filter

Applicability: Our smoother ≈ Optimization >> Local filter

Our Global Smoother
= Quality & Applicability of Optimization + Efficiency of Local filter

Performance of Proposed Method (2/2)

Properties	GF [8]	DT [9]	WLS [15]	Ours
Runtime efficiency [Sec. V.C]	0.15s	0.05s	3.3s	0.10s
Smoothing quality [Sec. V.A]	halo	halo	no halo	no halo
L_γ norm smoothing $(0 < \gamma < 2)$ [Sec. IV.D]	N.A.	N.A	N.A.	Yes
Using aggregated data [Sec. IV.E]	N.A.	N.A.	N.A.	Yes

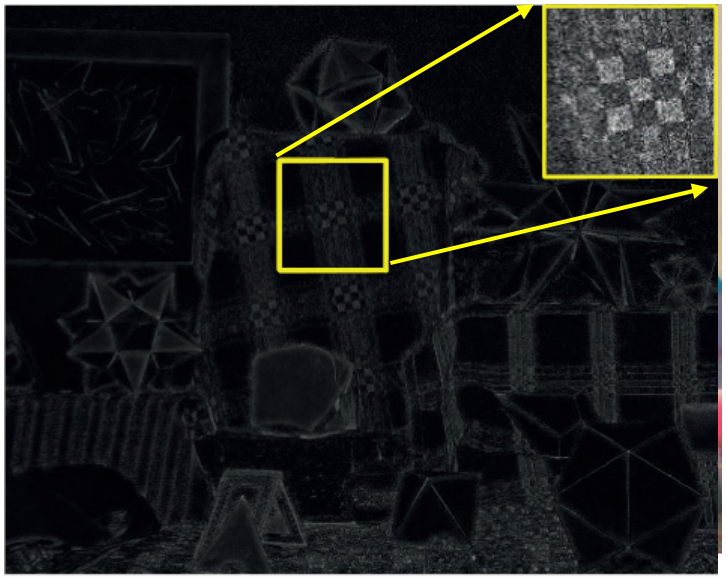


Applications

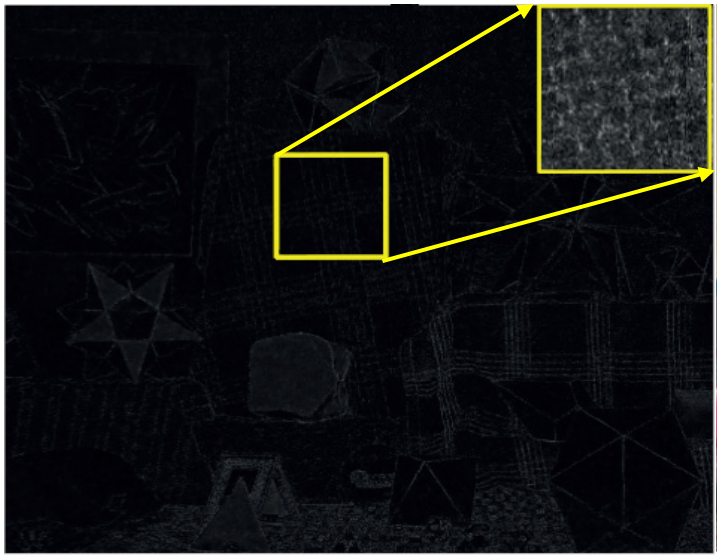
- **Color Image Denoising, Segmentation**
- **Joint/Cross Filtering Applications**
  - Depth denoising
  - Depth upsampling
  - Flash/no-flash denoising
  - Interactive image segmentation
- **Detail manipulation**
- **Saliency Detection**
- **Extension into Video**

Example Application: Image Denoising

- Not the best solution, but very fast and relatively good at low-level noise



GF



CLMF

Example Application: Color Image Segmentation

Mean-shift

CLMF

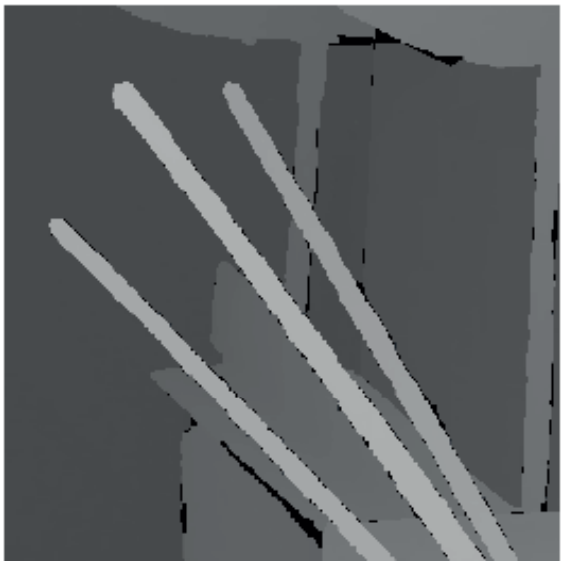
Iterative CLMF filtering → Use local support arms to label neighboring pixels → Region merging and pruning small-sized clusters

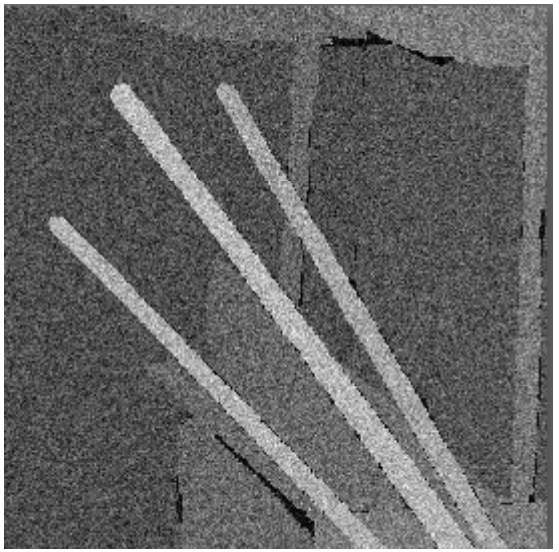
Example Application: Depth Denoising

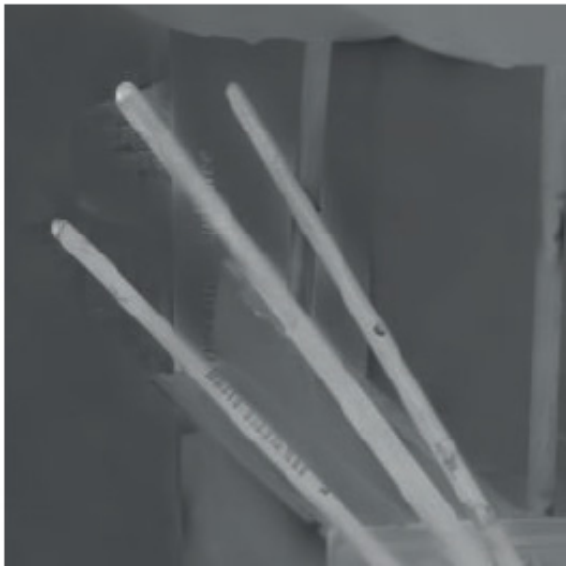
Depth Maps – Important but Usually not Good

- Wide applications, e.g.
  - View interpolation
  - 3D television
  - Robot navigation
- Acquiring methods
  - Passive stereo methods
  - Active depth sensors
    - ToF sensors (Mesa Imaging)
    - Structured lighting (Kinect)
- Low resolution, noisy, blurred, occlusion, inaccurate
  - Physical limitations
  - Real-time capturing constraint

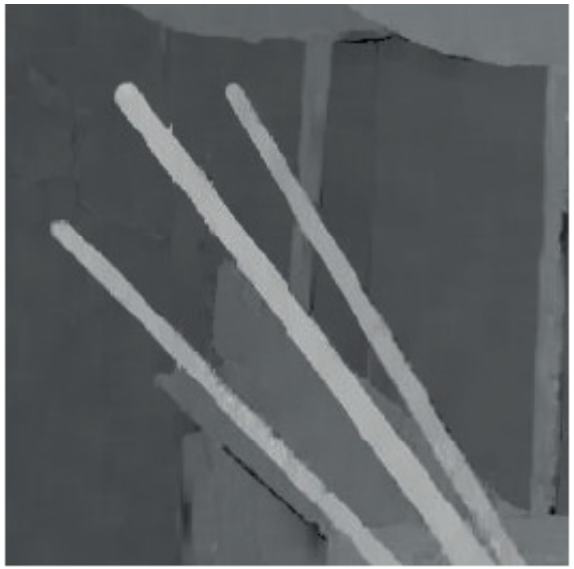
Example Application: Depth Denoising





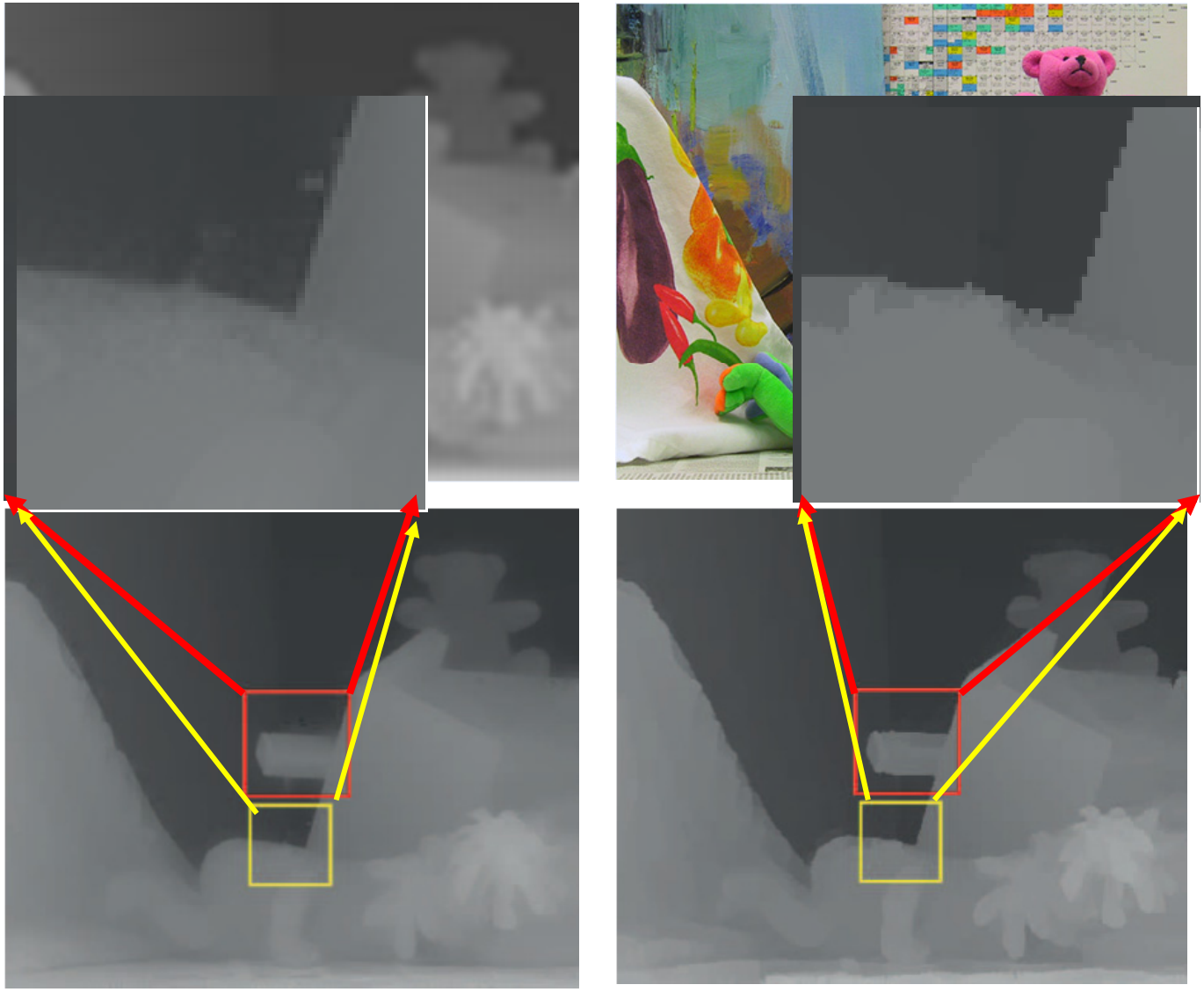


GF, RMSE=15.48



CLMF-0, RMSE=13.28

Example Application: Depth Enhancement



Example Application: Depth Enhancement

Ours is quite similar to state-of-the art work

Error rate %		BF	GF	WMF [10]	CLMF-1	**CLMF-0**
Tsu.	Disc.	40.3	49.6	25.1	39.1	**19.7**
	All	8.94	11.3	6.10	8.75	**4.30**
Venus	Disc.	15.0	25.6	**6.14**	12.2	9.98
	All	1.48	2.60	**0.63**	1.14	0.93
Teddy	Disc.	30.8	39.5	**26.1**	31.0	28.5
	All	11.6	15.5	**10.1**	12.0	11.6
Cones	Disc.	27.3	35.2	**23.7**	26.5	25.3
	All	13.1	17.0	**11.8**	12.9	13.1

[10] D. Min, J. Lu, and M. N. Do, 'Depth Video Enhancement Based on Weighted Mode Filtering,' TIP 2012.

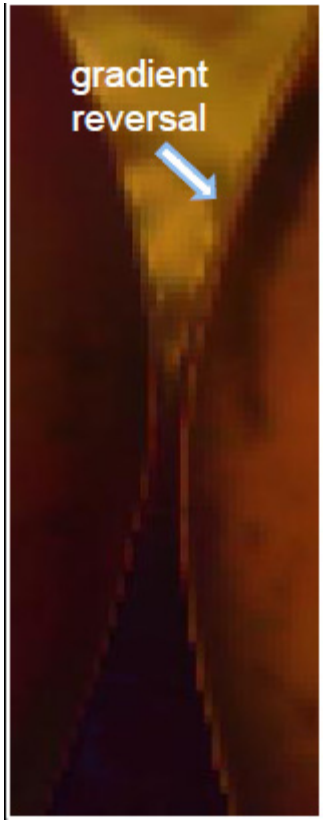
Example Application: Flash/no-flash denoising

No-flashed image

BF result

Flashed image

CLMF result



Bilateral Filter

CLMF

Example Application: Interactive Segmentation

Build FG and BG appearance models (GMM or histograms)

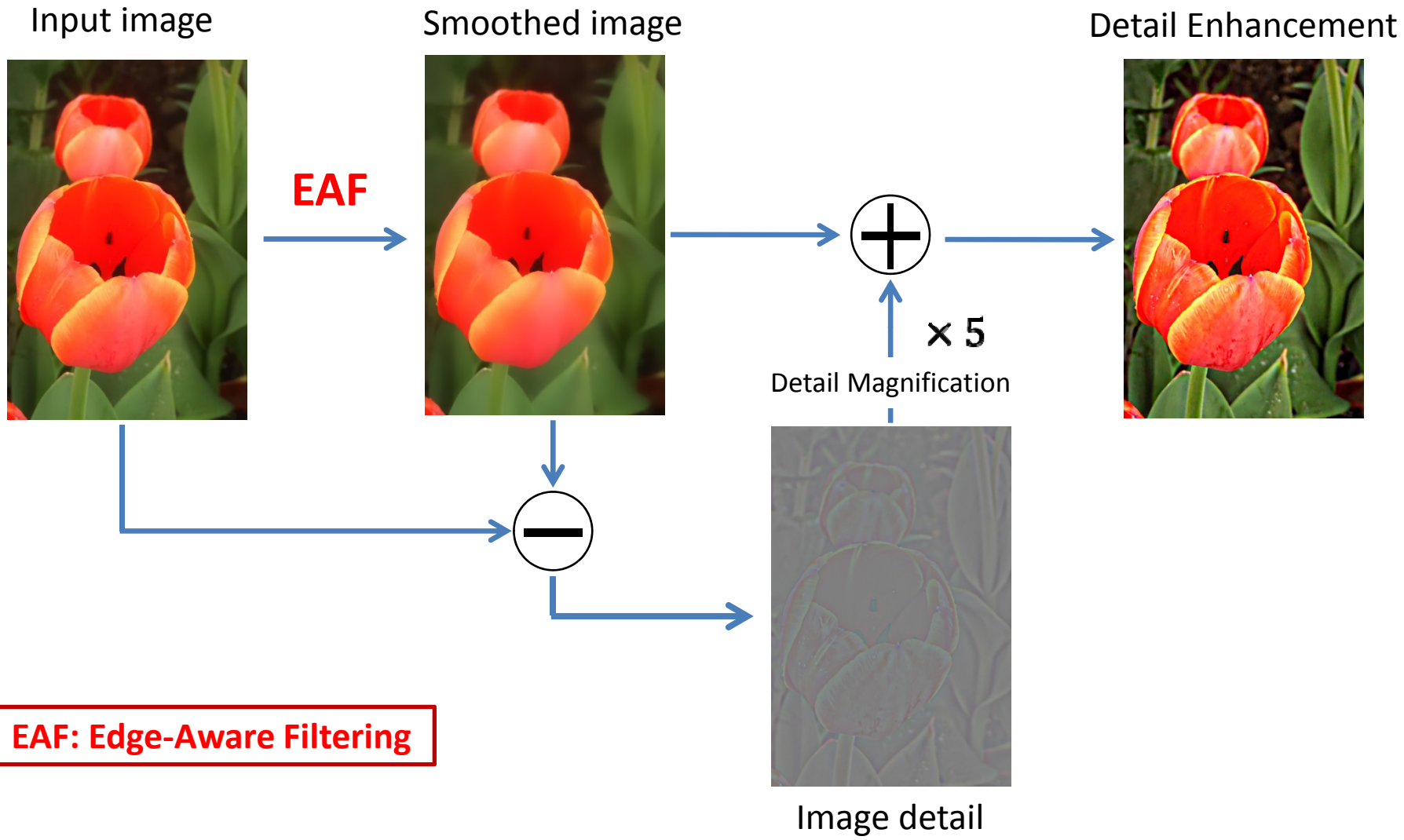
Evaluate each pixel's likelihood w.r.t. FG and BG models

Edge-aware filtering to the normalized likelihood map

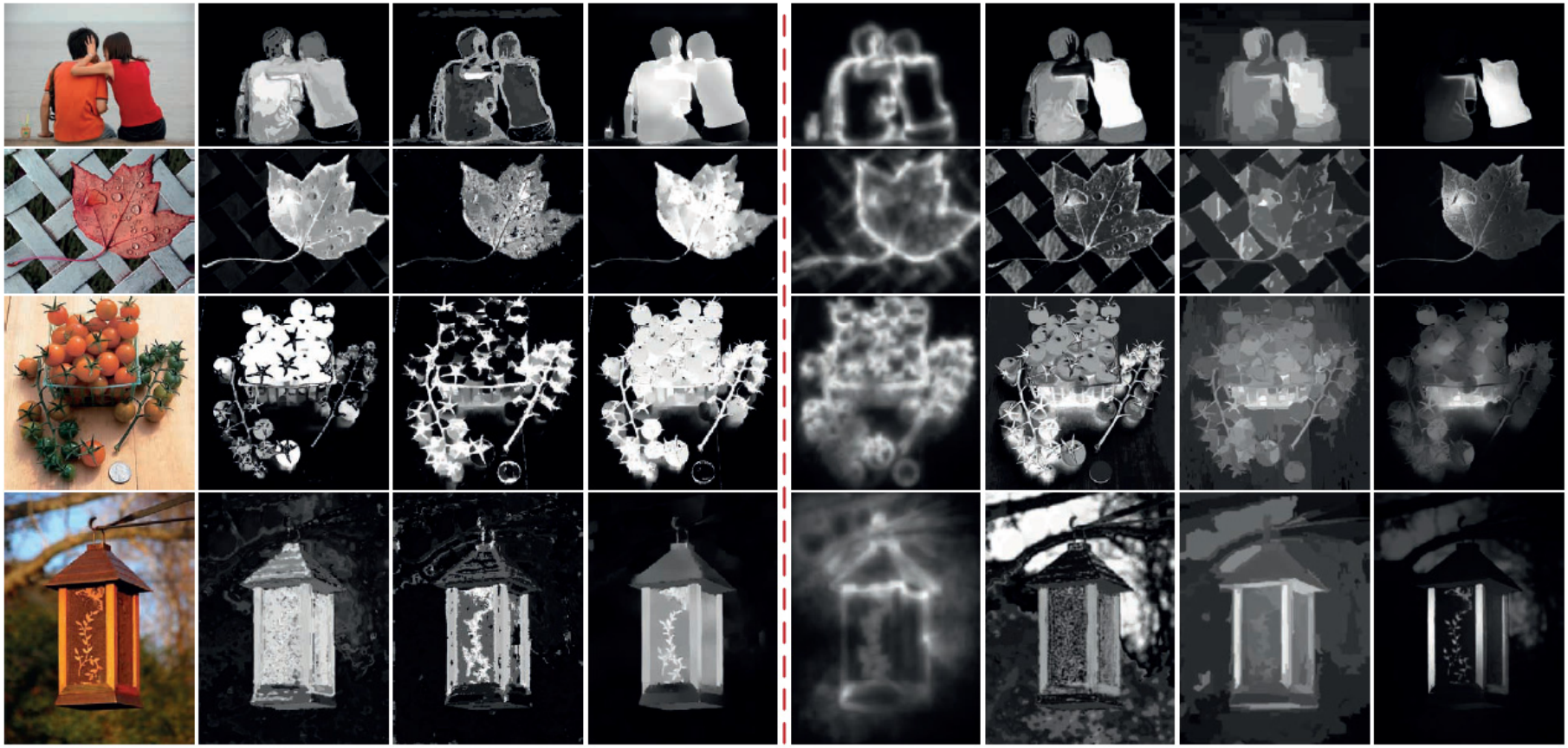
Thresholding and post-refinement

- Test on the database [Rother' 04], error rate for CLMF0 is 5.8%, and GF is 6.1%
- CLMF0 is over **2x** faster than GF

Example Application: Detail Manipulation



Saliency detection based on efficient filters
Saliency filters [CVPR'12] and PISA [CVPR'13]

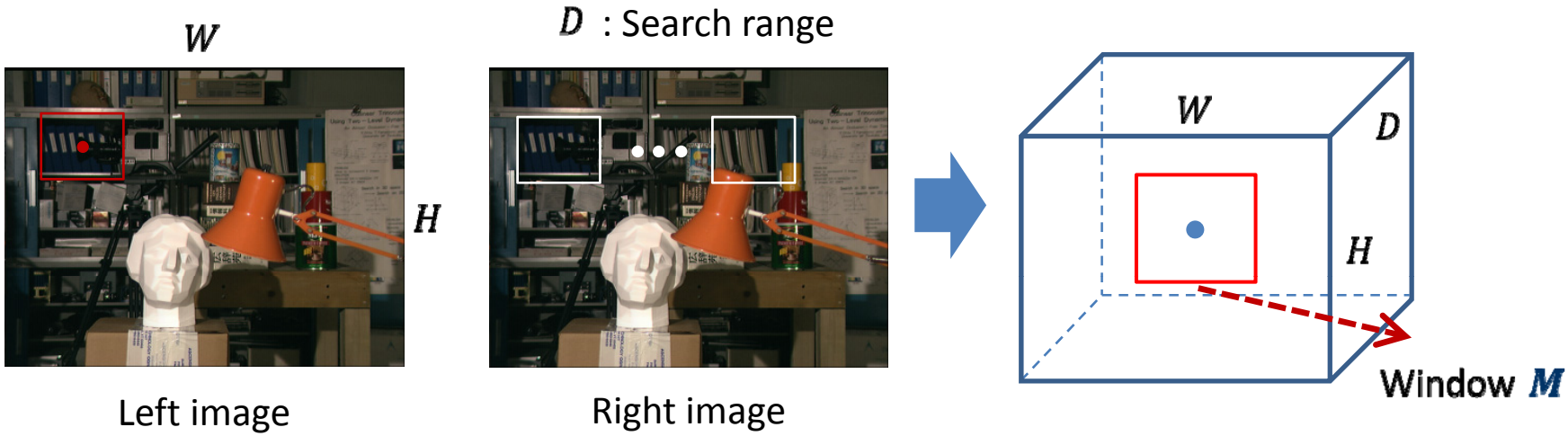


(a) Original (b) Color (c) Structure (d) PISA (e) CA[7] (f) HC[5] (g) RC[5] (h) SF[19]

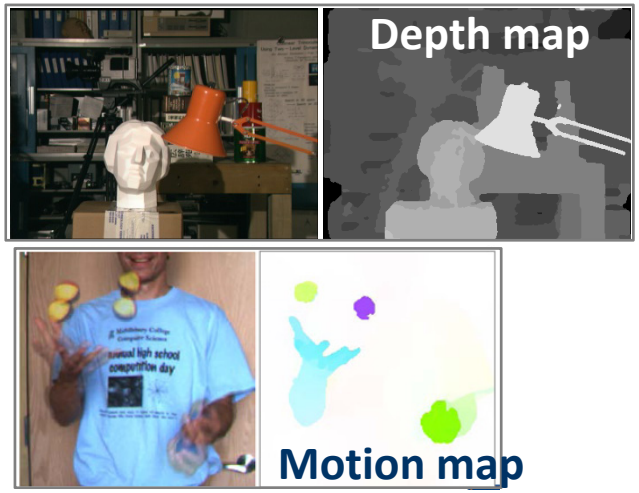
CA [6]	HC [7]	RC [7]	SF [4]	PISA	F-PISA
42.9	0.011	0.115	0.125	0.650	0.035

EAF for Discrete Labeling Optimization

- **Labeling: assigning a label for all pixels (e.g. depth, motion)**



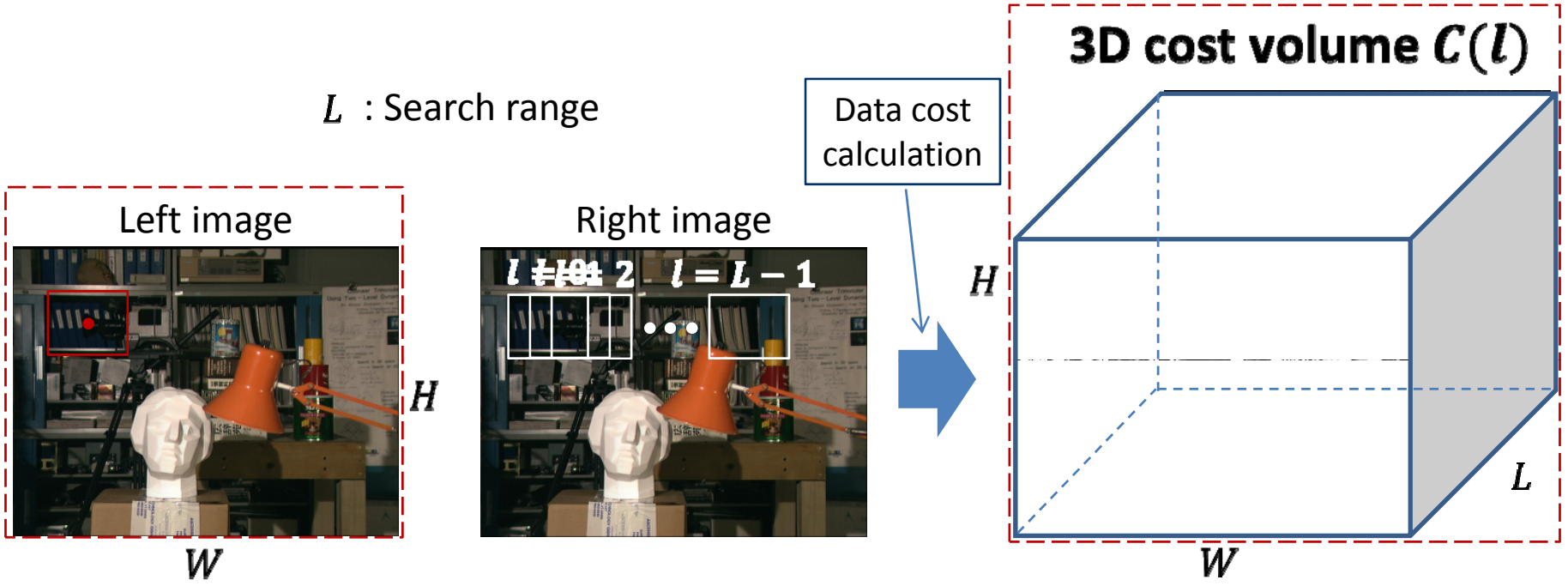
Examples of label maps



Applications using depth/motion

- View synthesis for 3DTV
- Frame up-conversion (30→ 60fps)
- 3D scene reconstruction
- Scene understanding
- 3D video editing

EAF for Discrete Labeling Optimization



JOINT Edge-Aware Filtering

for each slice of $C(l)$

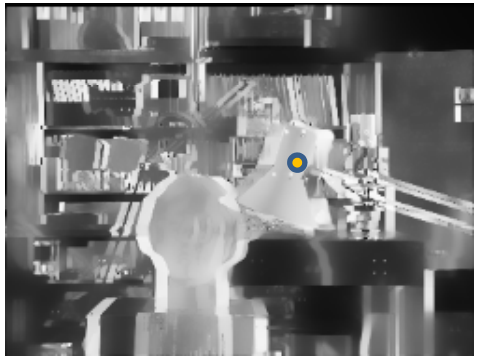
EAF for Discrete Labeling Optimization

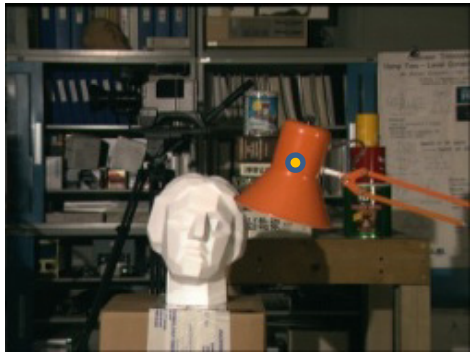
- Based on cross/joint (bilateral) filtering principles
- Cost volume filtering – repeated cross/joint filtering

$$Output_{Cost} = \mathbf{EAF}(Color, Ini_{Cost})$$

$$\tilde{C}_p(l) = \sum_{q \in W_p(r)} \omega_{q,p}(I) C_q(l)$$

While a label
$d = 0 \rightarrow D - 1$





[Yoon & Kweon,PAMI06], [Rhemann et al.,CVPR11]

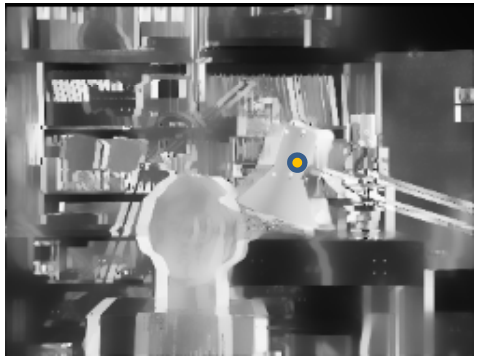
EAF for Discrete Labeling Optimization

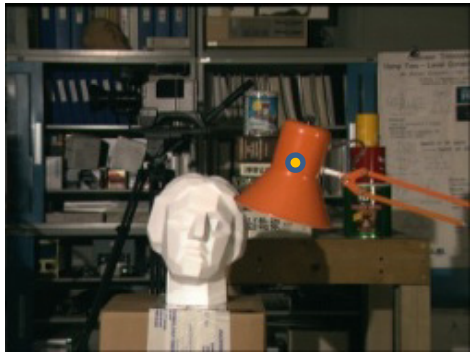
Simple WTA

$$d(p) = arg\min_{l} \tilde{C}_p(l)$$

Simple WTA

$$d(p) = arg\min_{l} \tilde{C}_p(l)$$

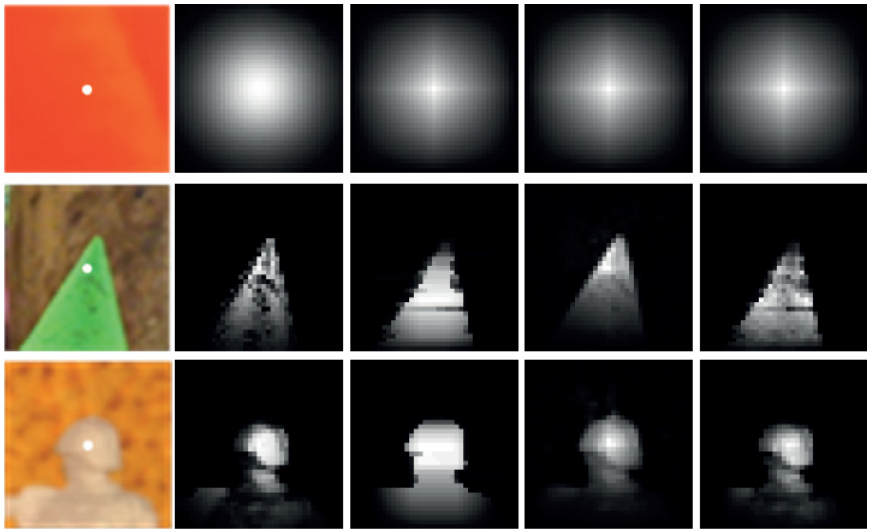




EAF for Discrete Labeling Optimization

- Based on cross/joint (bilateral) filtering principles
- Cost volume filtering – repeated cross/joint filtering
- Spatially coherent and edge-aligned labeling results

EAF for Discrete Labeling Optimization

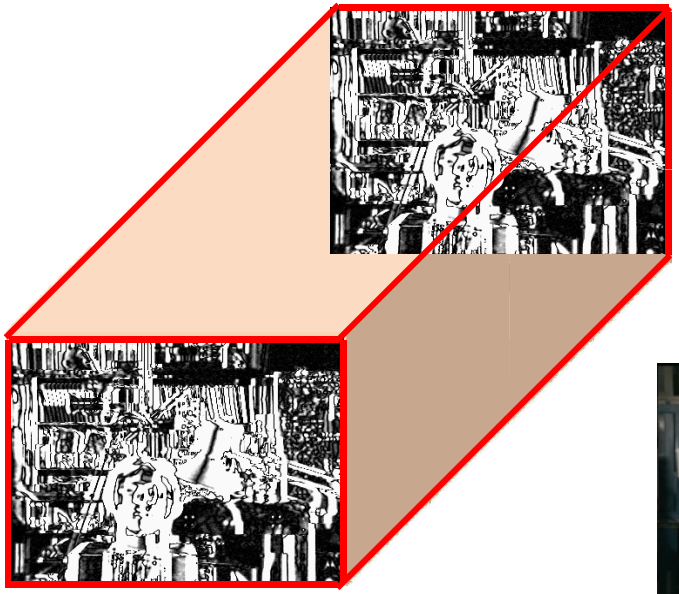


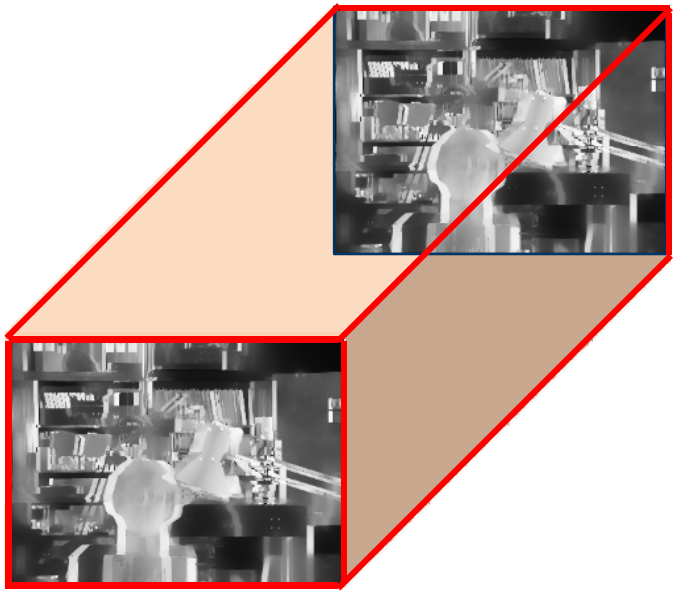
- Based on cross/joint (bilateral) filtering principles
- Cost volume filtering – repeated cross/joint filtering
- Spatially coherent and edge-aligned labeling results
- Runtime often independent of the filter kernel size *m*

But, the curse of the label search space

$$\mathcal{L} = \{0, 1, ..., L-1\}$$

O(*M*L*) !!**





Also said for recent *filter-based mean-field inference for random fields* [Vineet et al. ECCV12]

The label space can be HUGE

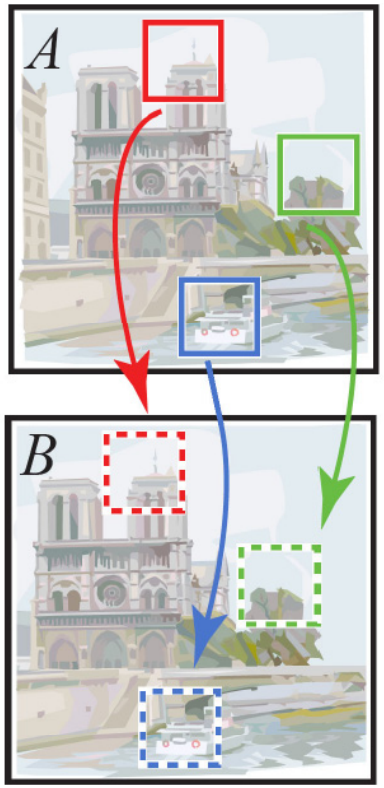
- Two-dimensional motion search
- Displacement in subpixel accuracy
- Over-parameterized surface or motion modeling
- …

- e.g. motion search range in [-40, 40]*[-40, 40] * 8 * 8 ➔ **410,000** labels! ➔ **410,000** joint filtering!

PatchMatch for Approximate Nearest-Neighbor Field (ANNF)

[Barnes et al., SIGGRAPH09, ECCV10]

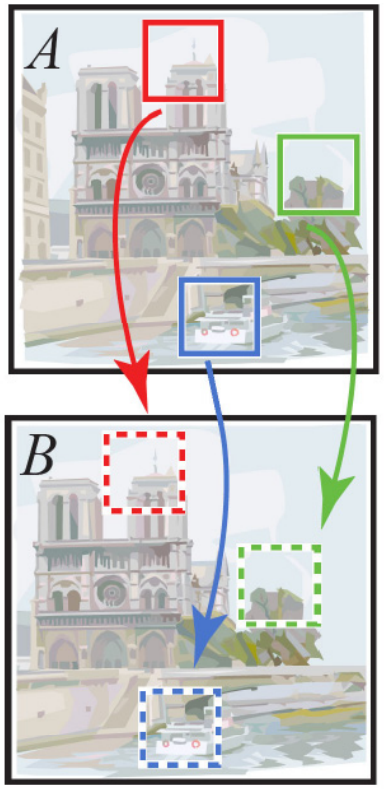
- Find for every patch in A the nearest neighbor in B under a patch distance metric

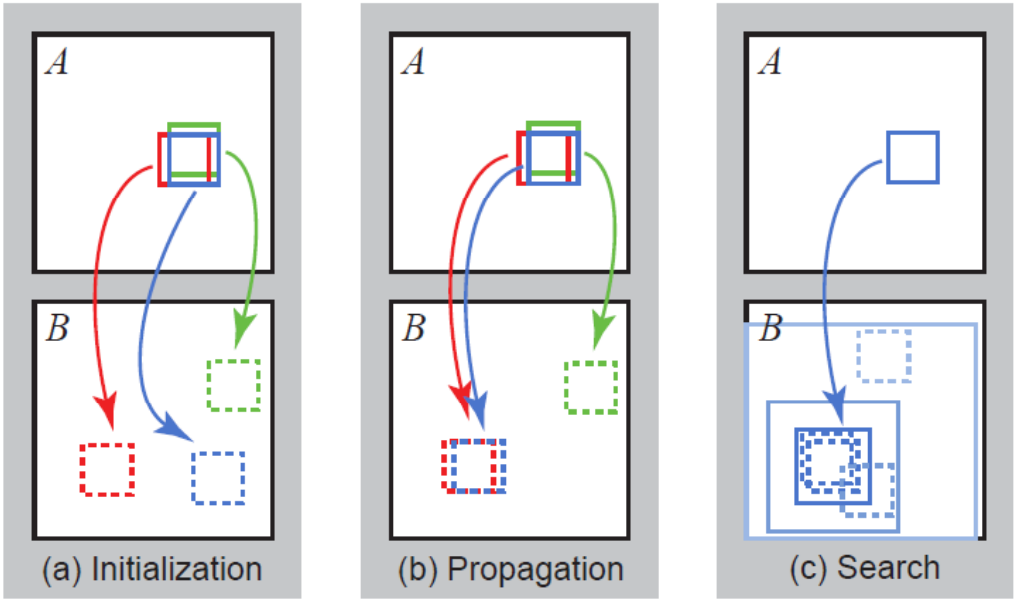


PatchMatch for Approximate Nearest-Neighbor Field (ANNF)

[Barnes et al., SIGGRAPH09, ECCV10]

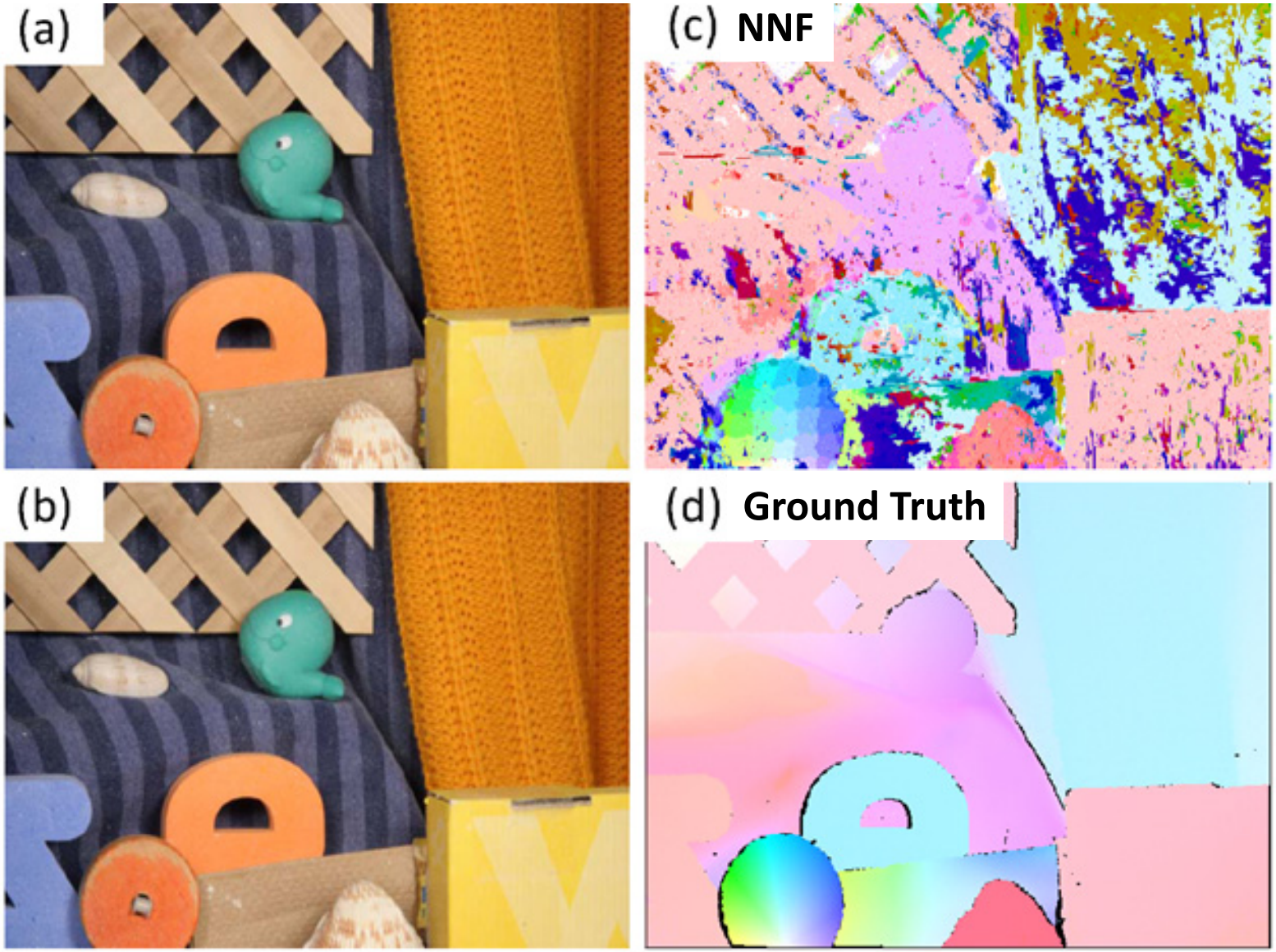
- Find for every patch in A the nearest neighbor in B under a patch distance metric
- Leveraging *natural structure of images* and *the law of large numbers*
- Iterative *propagation & random search*





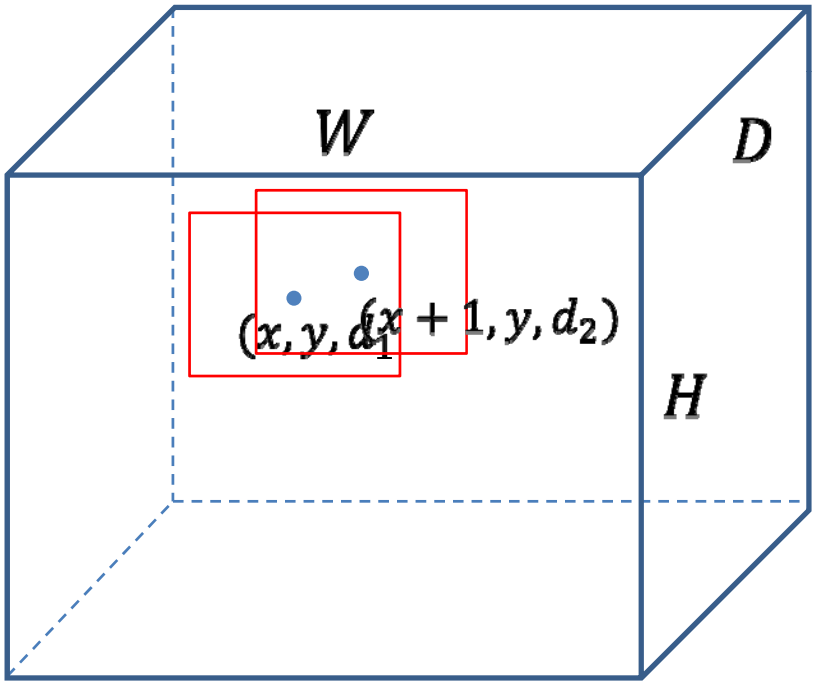
O(*m*M*logM*) !!

NNF != Visual correspondence field without enforcing spatial smoothness



Random Access on Label Space Makes Problem DIFFICULT

- Pixel-wise randomized search of original PatchMatch
  - Fragmental data access on 3D cost volume



3D cost volume

Matching window for aggregation (nonlinear filtering)

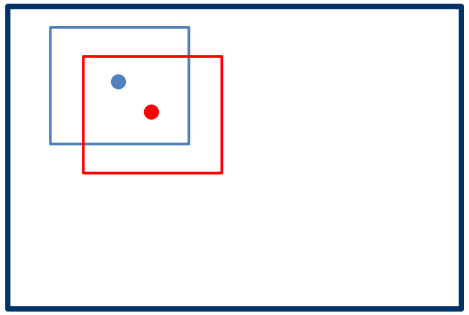
This random access hinders the application of efficient $O(N)$ filtering technique
- d_1 for (x, y) and d_2 for $(x+1, y)$

$$O(IMlogD)$$

I: image size ($H \times W$), M: filter size, D: label size

$O(N)$ filtering needs redundancy!

- Redundancy of simultaneously computing a weighted sum for ***all*** pixels
  - ***Guided filter*** (ECCV 2010, PAMI 2013): multiple number of integral sum (box filtering)
  - ***Recursive filter of Domain Transform*** method (SIGGRAPH 2011): Recursive propagation of aggregated data in causal and non-causal manners
  - ***$O(N)$ Bilateral Filter on bilateral grid*** (ECCV 2006): Linear Gaussian filtering on high dimensional volume



The filtered data of ● should be reused for filtering ●

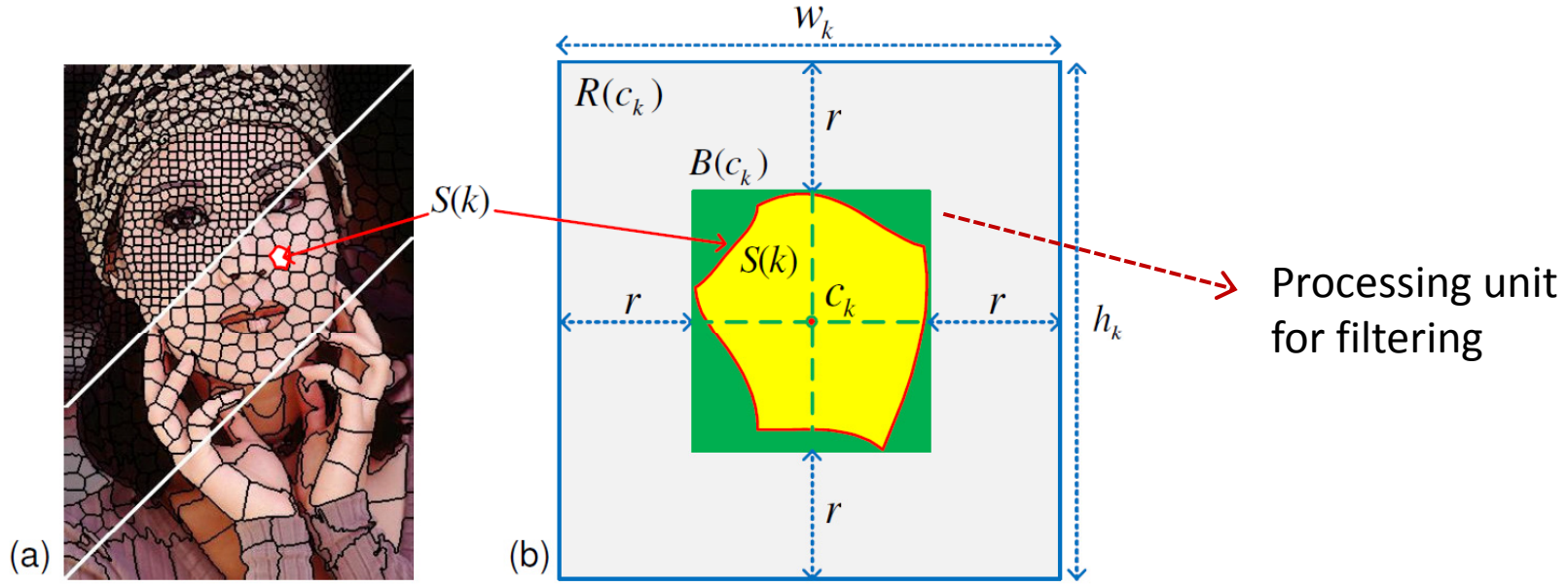
PatchMatch Filter (PMF)

- *Super-pixel* based randomized search algorithm
  - Collaborative filtering within a single super-pixel

- Efficient filtering + PatchMatch algorithm

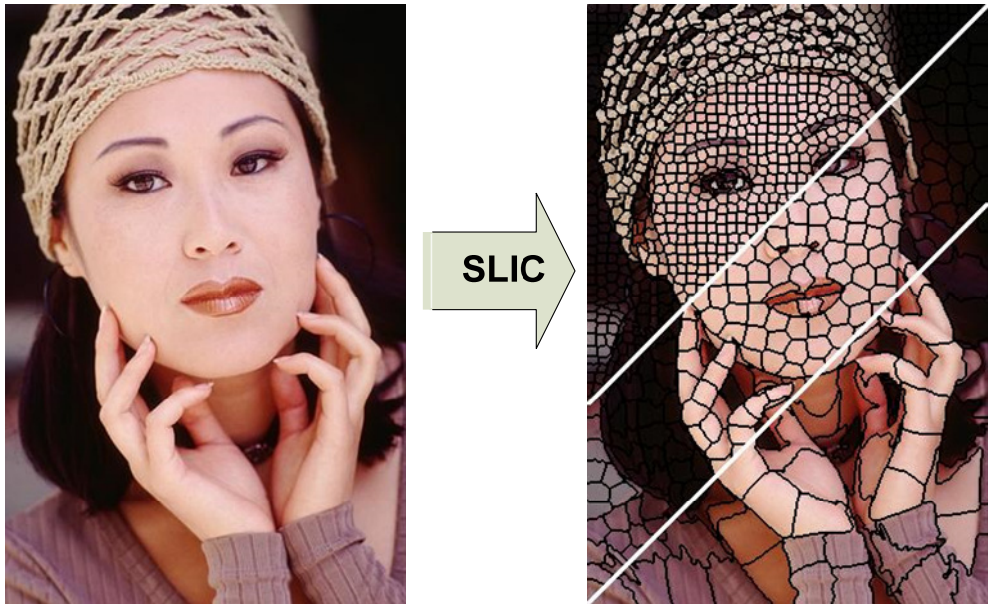
$$O(ID) \quad + \quad O(IMlogD) \quad \dashrightarrow \quad O(IlogD)$$

  - Collaborative randomized search



J. Lu, et al., "PatchMatch Filter: Efficient Edge-Aware Filtering Meets Randomized Search for Fast Correspondence Field Estimation," *CVPR*, 2013.

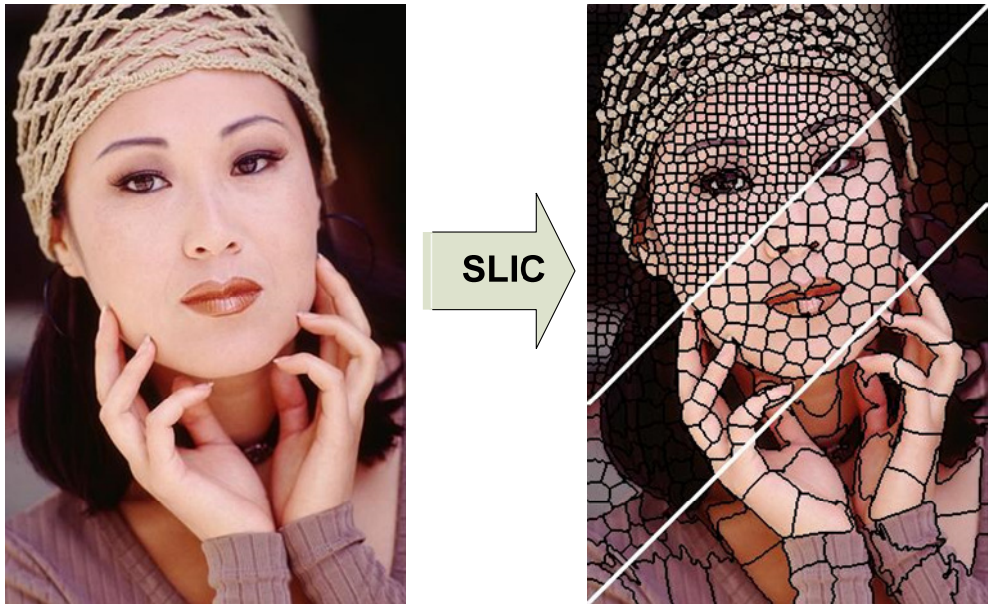
Segments as the bridge



• Labeling solutions are spatially smooth and discontinuities-aligned
➔ Collaborative label search and propagation
➔ Extends the propagation range

[Achanta et al. PAMI12]

Segments as the bridge



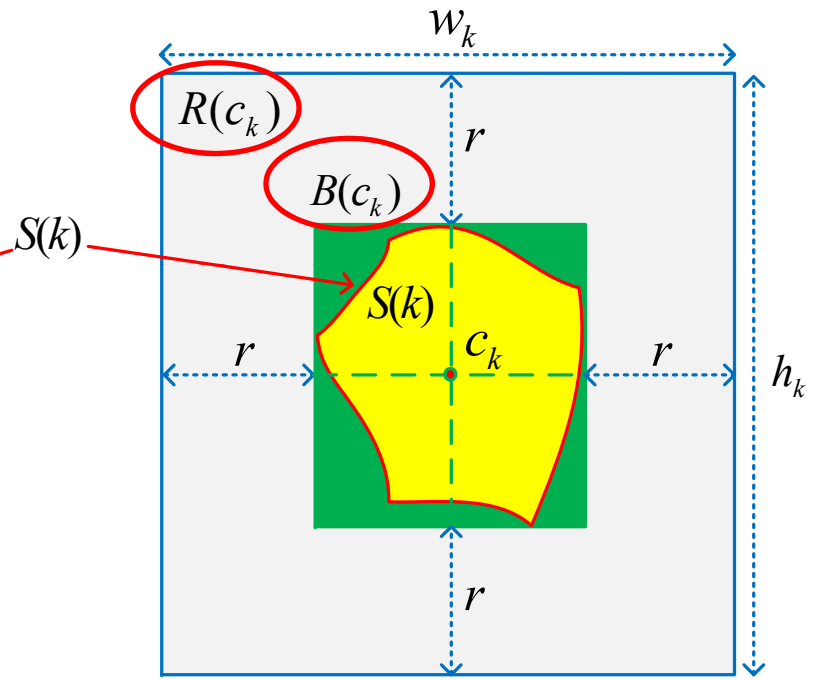
[Achanta et al. PAMI12]

• Labeling solutions are spatially smooth and discontinuities-aligned
➔ Collaborative label search and propagation
➔ Extends the propagation range

• The efficiency of EAF comes from high computational redundancy or vast opportunity for shared computation reuse

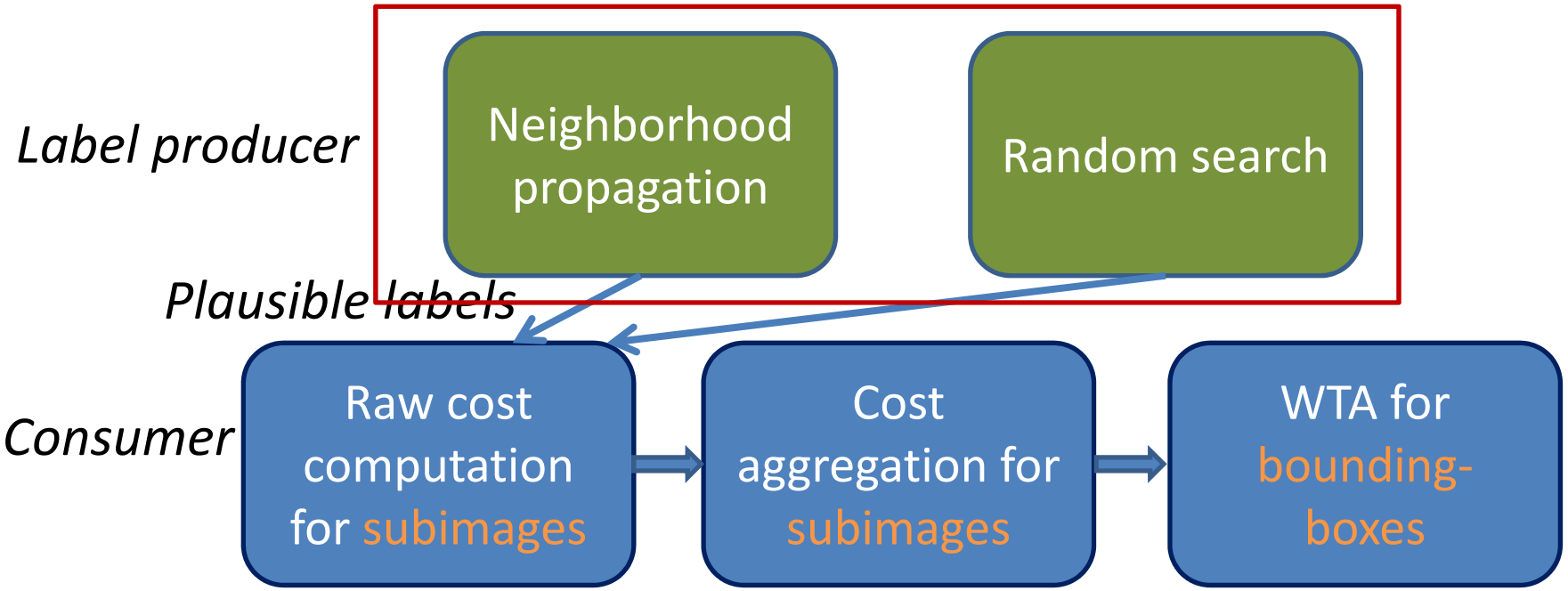
Segments as the bridge

SLIC

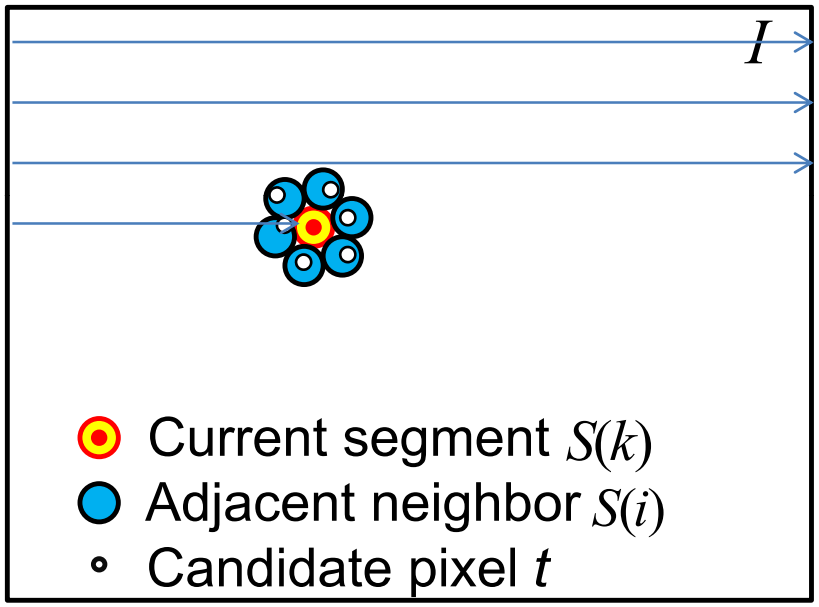


Baseline PMF algorithm: General recipe

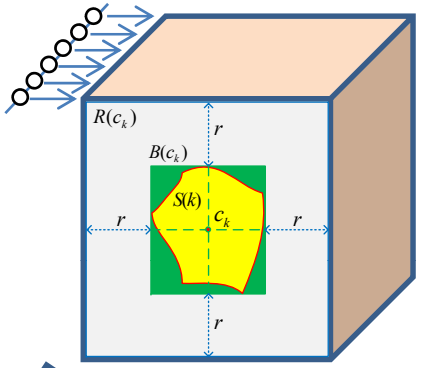
- Task-specific parameterization of the label space
- Initial label assignment to each segment
- **Process each segment in scan order iteratively**
  - For the current segment, evaluate the candidate labels generated from two sources: *propagation* & *random search*



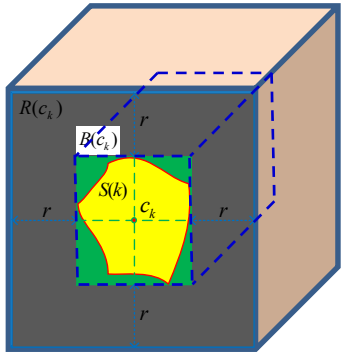
Neighborhood label propagation & evaluation



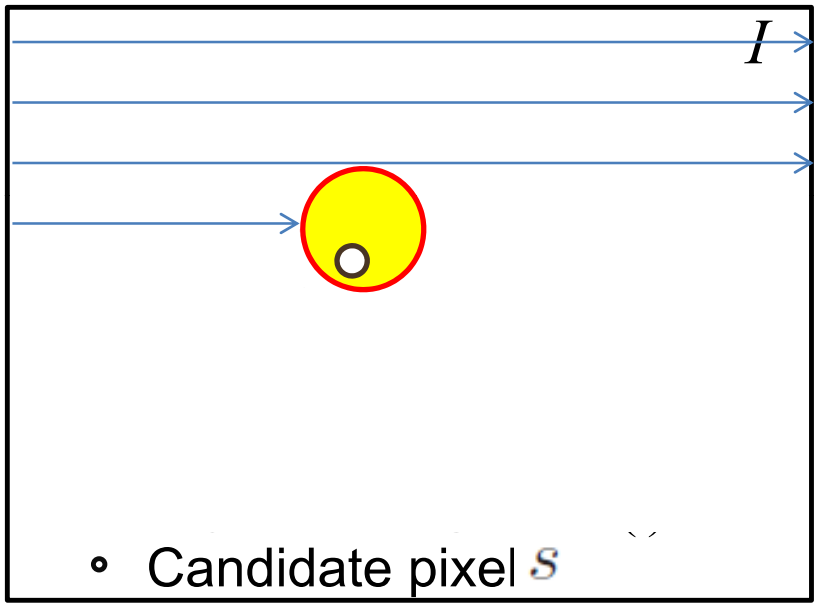
$$\mathcal{L}_t = \{l_t\}$$

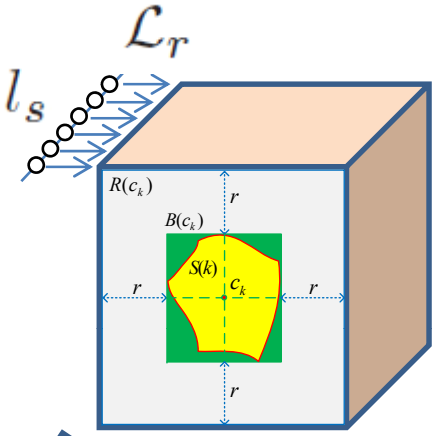


$$\mathbf{f} : \mathbf{C}\left(R(c_k), \{l \in \mathcal{L}_t\}\right) \mapsto \tilde{\mathbf{C}}\left(B(c_k), \{l \in \mathcal{L}_t\}\right)$$

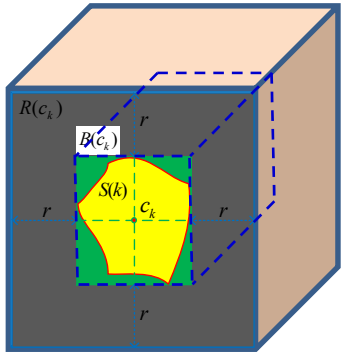


Random search & evaluation



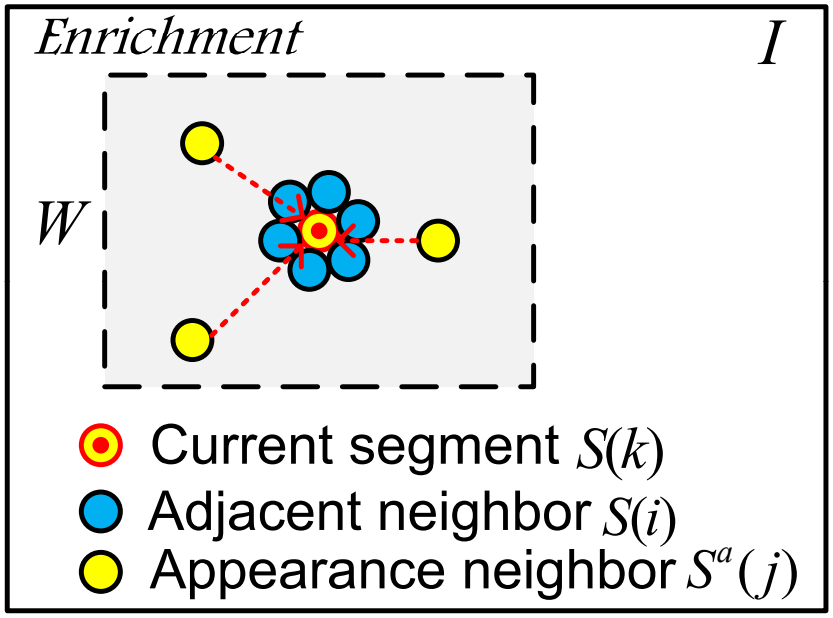


$$\mathbf{f} : \mathbf{C}\left(R(c_k), \{l \in \mathcal{L}_r\}\right) \mapsto \tilde{\mathbf{C}}\left(B(c_k), \{l \in \mathcal{L}_r\}\right)$$



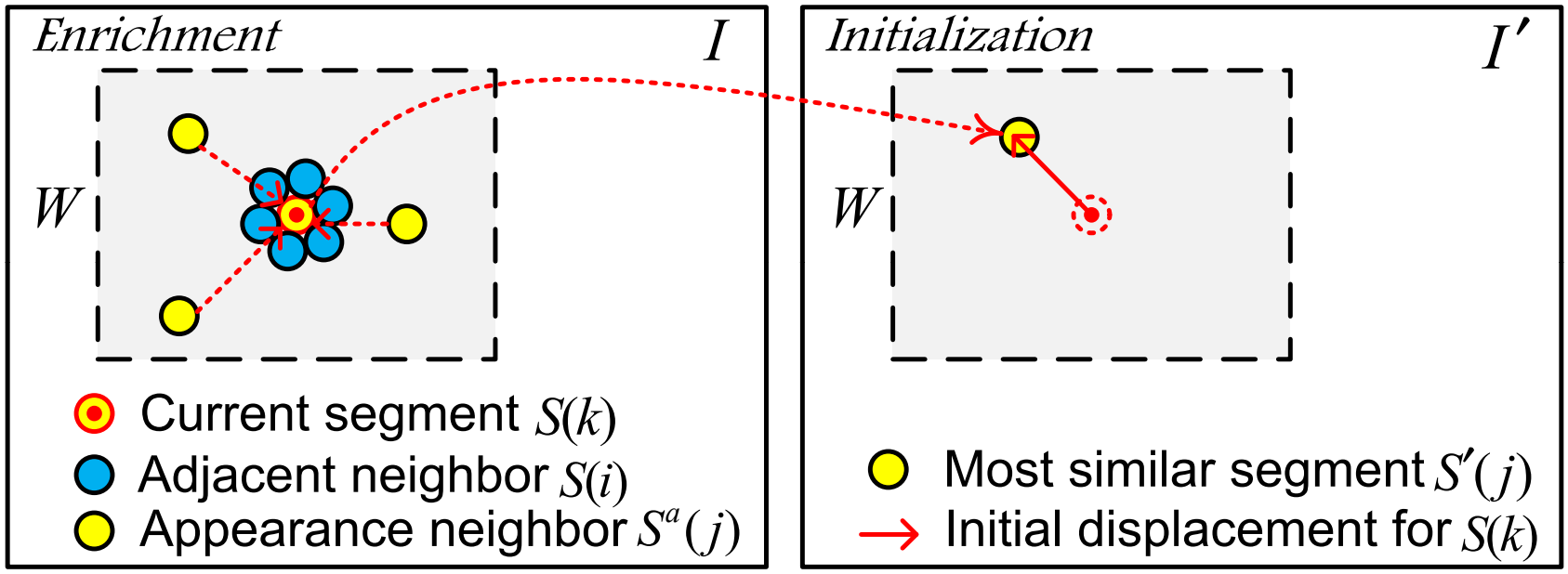
A list records the labels visited for each segment *S*(*k*), so NO subimage filtering for any revisited label.

Improved strategies for PMF: Enrichment



$$H(S(k),S(j)) = \sum_{s\in S(k),t\in S(j)} \exp\left(-\frac{\|s-t\|^2}{\sigma_s^2} - \frac{\|I_s-I_t\|^2}{\sigma_r^2}\right)$$

Improved strategies for PMF: Initialization



$$H(S(k), S(j)) = \sum_{s \in S(k), t \in S(j)} \exp\left(-\frac{\|s-t\|^2}{\sigma_s^2} - \frac{\|I_s - I_t\|^2}{\sigma_r^2} \right)$$

Complexity comparison

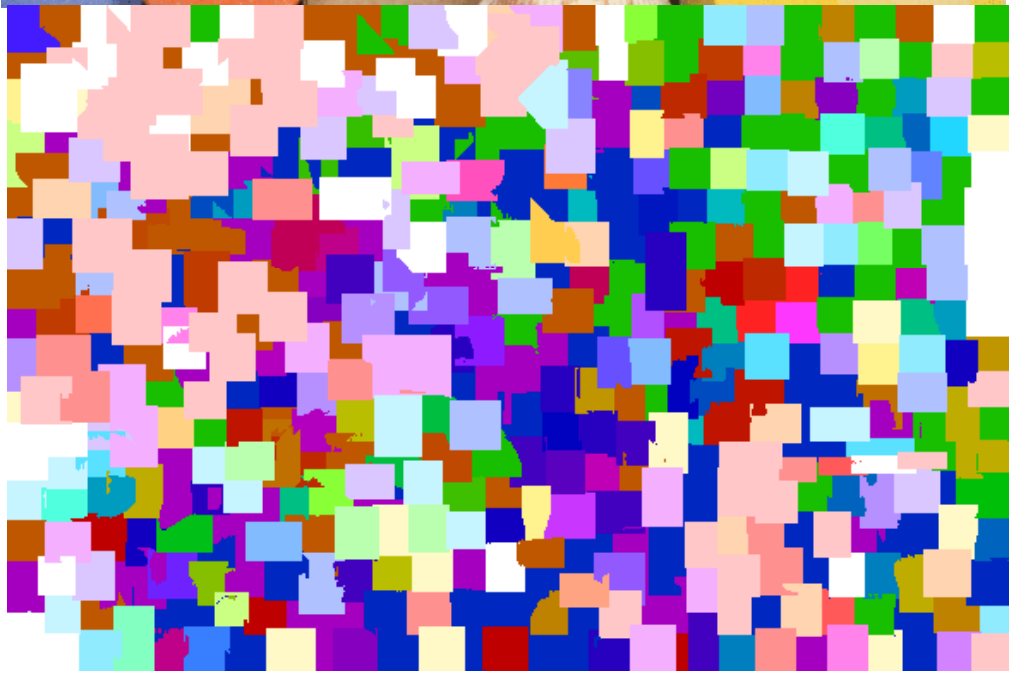
	CostFilter [17]	PatchMatch [7]	PMF
Complexity	$O(ML)$	$O(mM\log L)$	$O(M\log L)$
Memory	$O(M)$	$O(M)$	$O(M)$

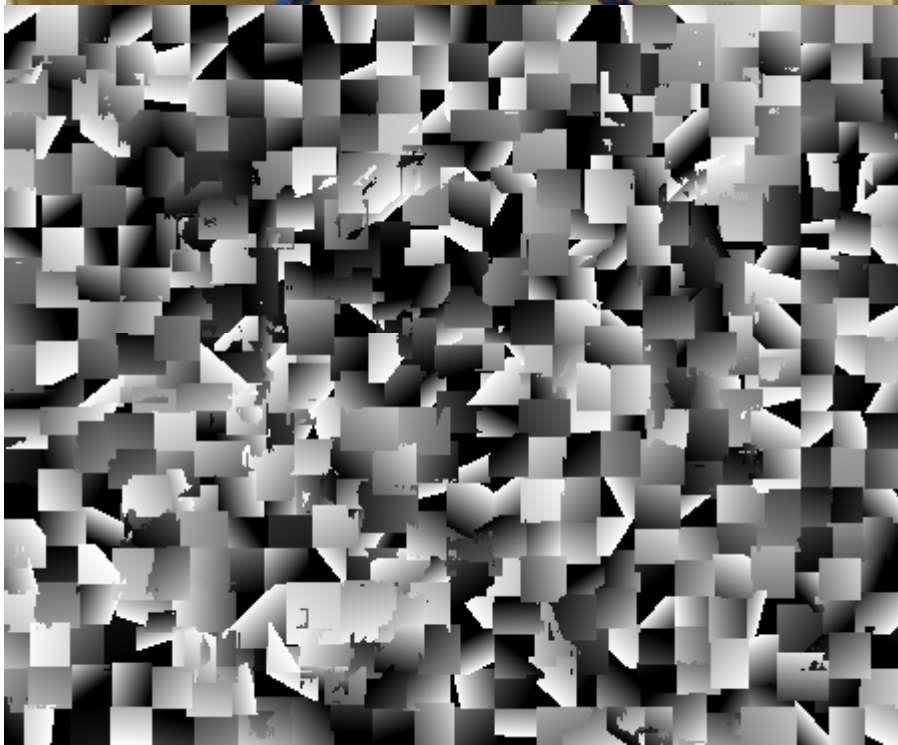
m – filter kernel size

M – number of pixels

L – label space size

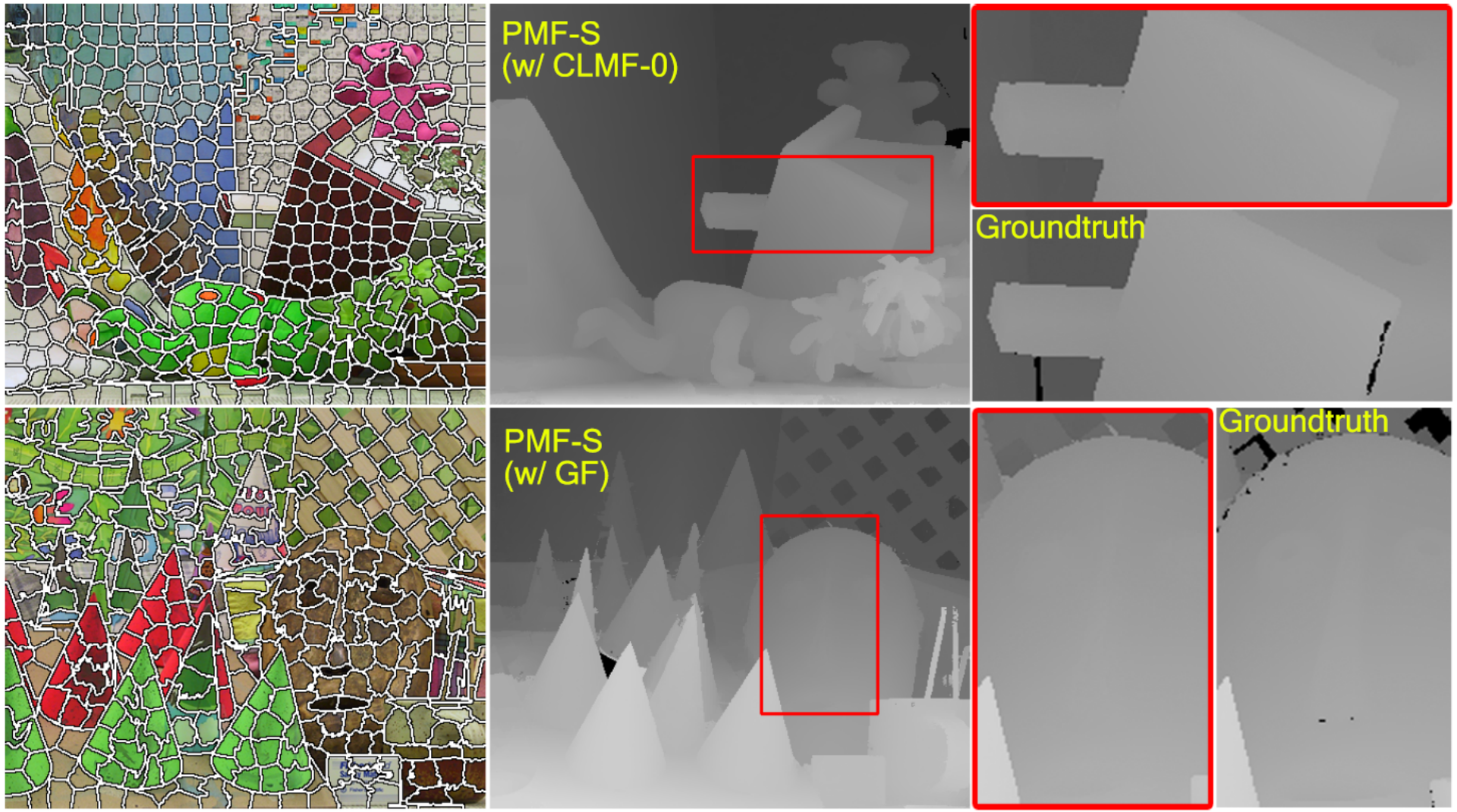

digi-key
www.digikey.com

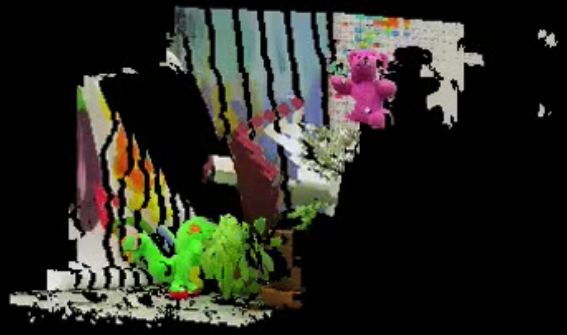


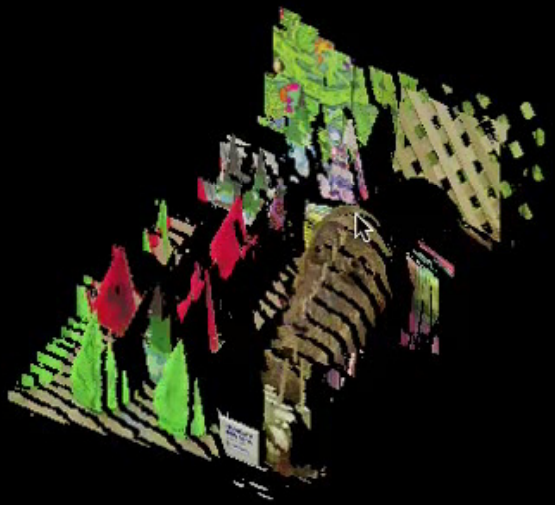


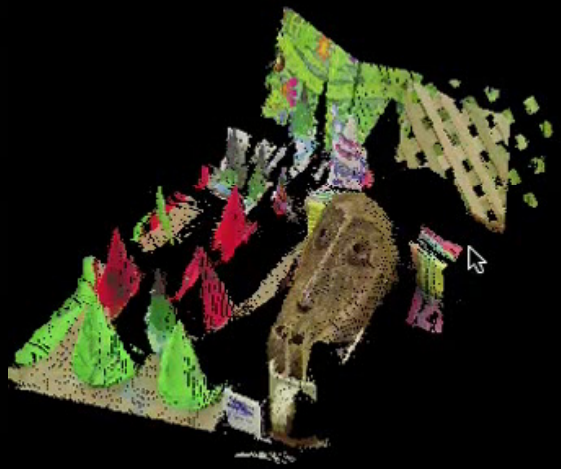
Algorithm	*Teddy*			*Cones*		
	nocc	all	disc	nocc	all	disc
PMF-S-GF	4.45_2	9.44_2	13.7_2	$\mathbf{2.89_1}$	8.31_2	$\mathbf{8.22_1}$
PMBP [6]	5.60_3	12.0_6	15.5_3	3.48_3	8.88_4	9.41_4
PMF-S-CLMF0	$\mathbf{4.07_1}$	10.5_3	$\mathbf{12.1_1}$	2.96_2	8.84_3	8.38_2
PatchMatch [7]	5.66_4	11.8_5	16.5_4	3.80_5	10.2_6	10.2_5

- **PMF-C** runs over **3-7x** faster for high-res. stereo images (e.g. 1M pixels) than CostFilter
- **PMF-S** over **10x** faster than PatchMatch Stereo[7]
- In any case, w/ CLMF-0 runs **2-3x** than w/ GF


PMF-S
(w/ CLMF-0)
Groundtruth
PMF-S
(w/ GF)
Groundtruth



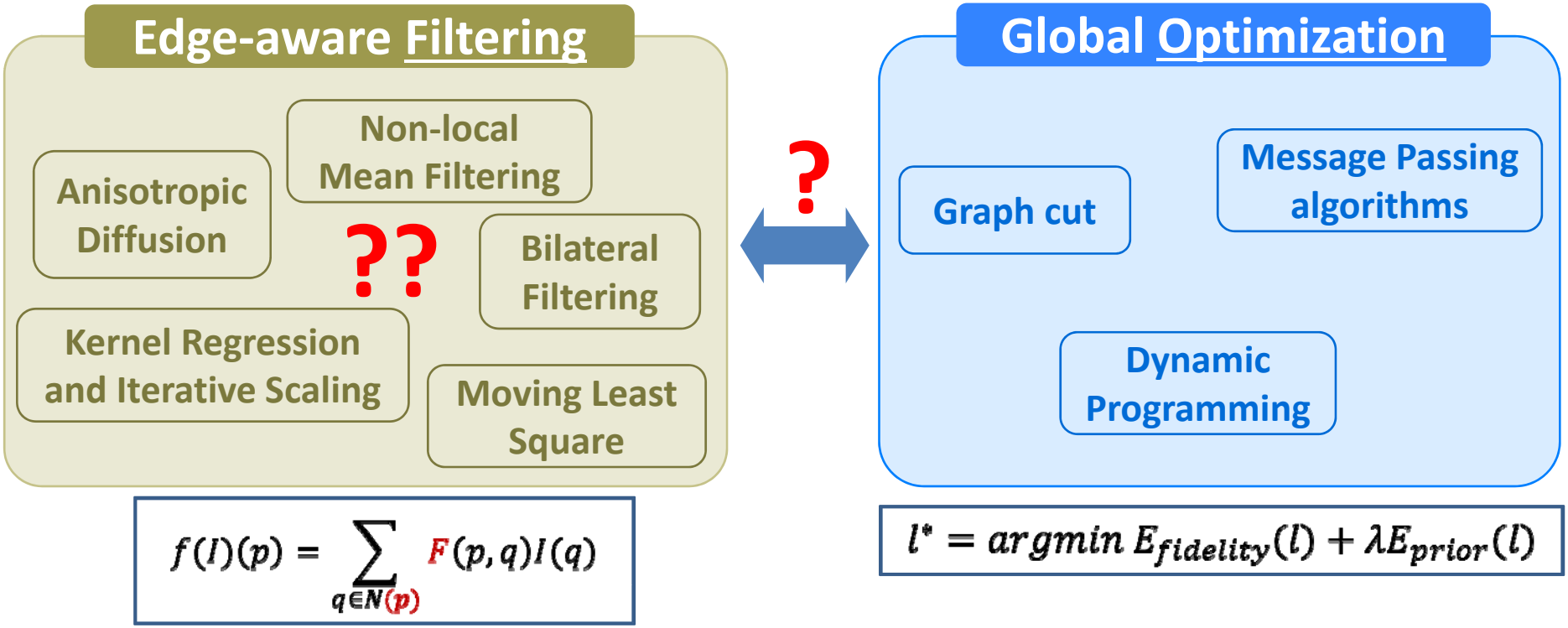




Efficient Edge-Aware Filters: Theory and Generalization

Goal of Filtering and Optimization: Efficient Propagation of High-dimensional Data

Better Theoretical Understanding → Better Filtering or Optimization Algorithms Design

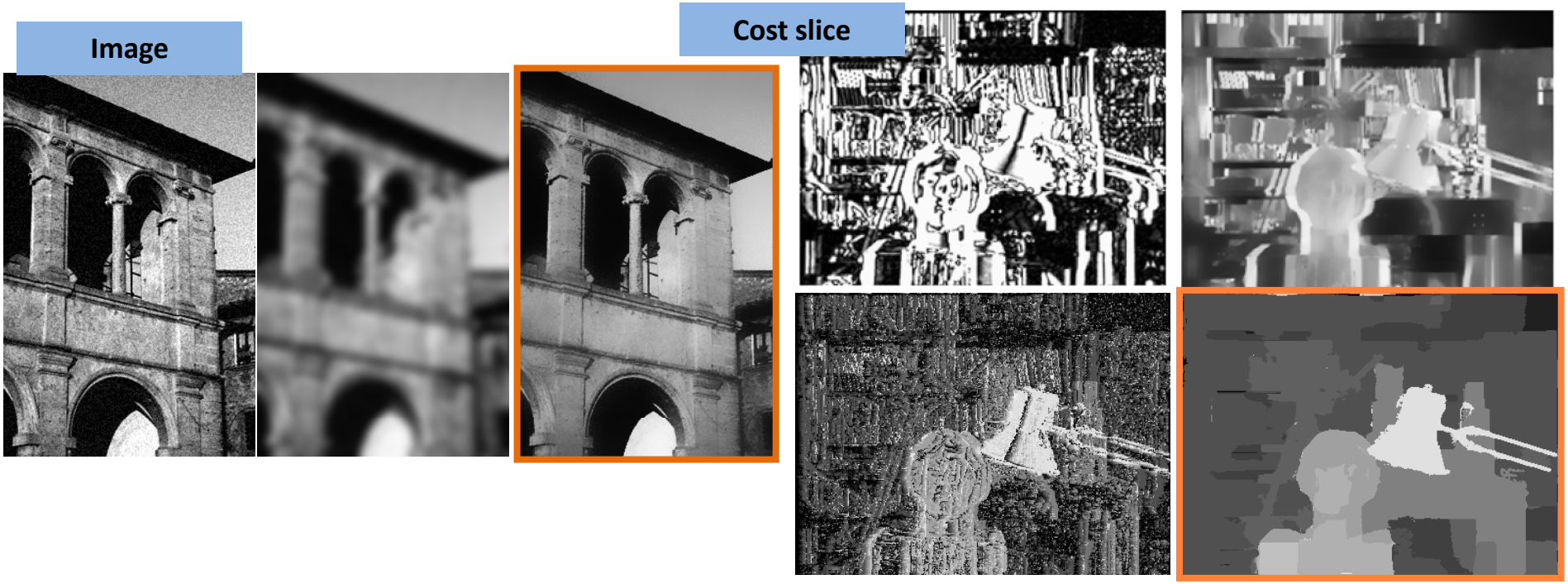


$$f(I)(p) = \sum_{q \in N(p)} F(p,q) I(q)$$

$$I^* = argmin\, E_{fidelity}(I) + \lambda E_{prior}(I)$$

General algorithmic requirements

- Obtaining spatially-smooth, contrast-sensitive pixel-labeling assignment
- Separating large-scale structures from small-scale variations/details
- Eliminating outliers when performing smoothing



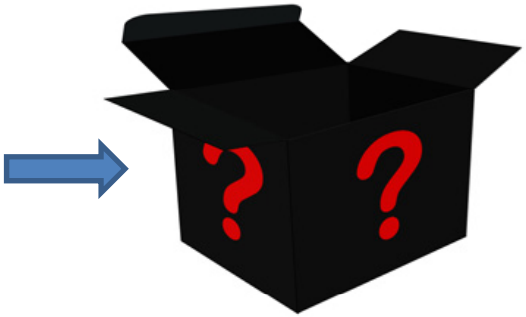
Ubiquitous Filtering: dealing with pixels, correspondences, segments, objects…

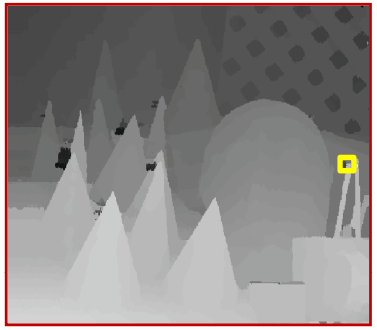
[Dabov et al., TIP'07]

[Durand & Dorsey, SIGGRAPH'02]

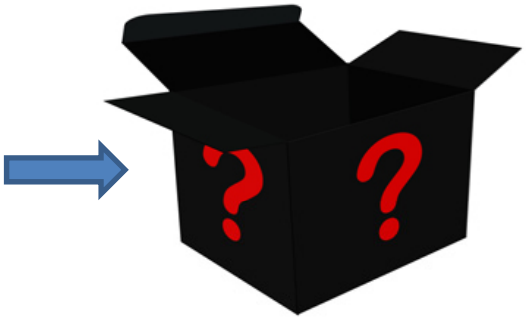
[Winnemoller et al., SIGGRAPY'06]

Ubiquitous Filtering: dealing with pixels, correspondences, segments, objects…



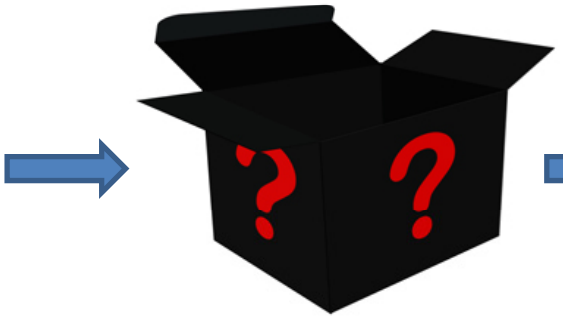


[Lu et al., CVPR'12]

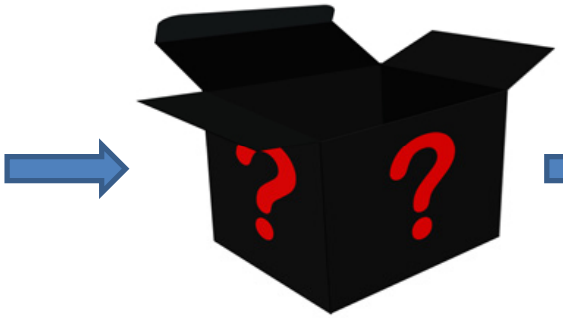


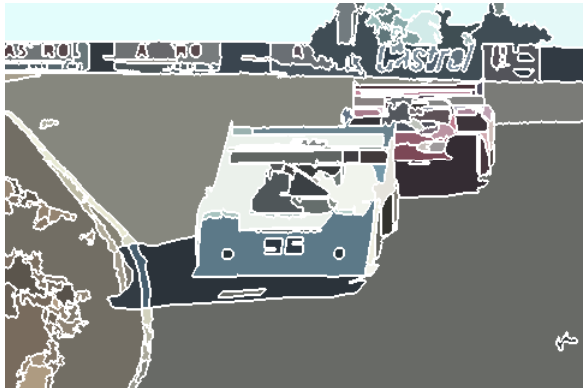
[Min et al., PAMI'13]

Ubiquitous Filtering: dealing with pixels, correspondences, **segments**, **objects**…



[Rhemann et al., CVPR'11]



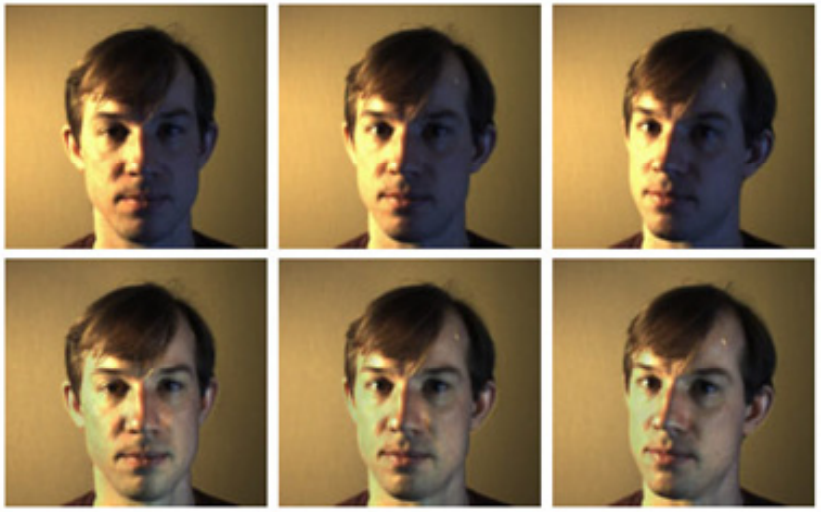


[Comaniciu & Meer, PAMI'02]

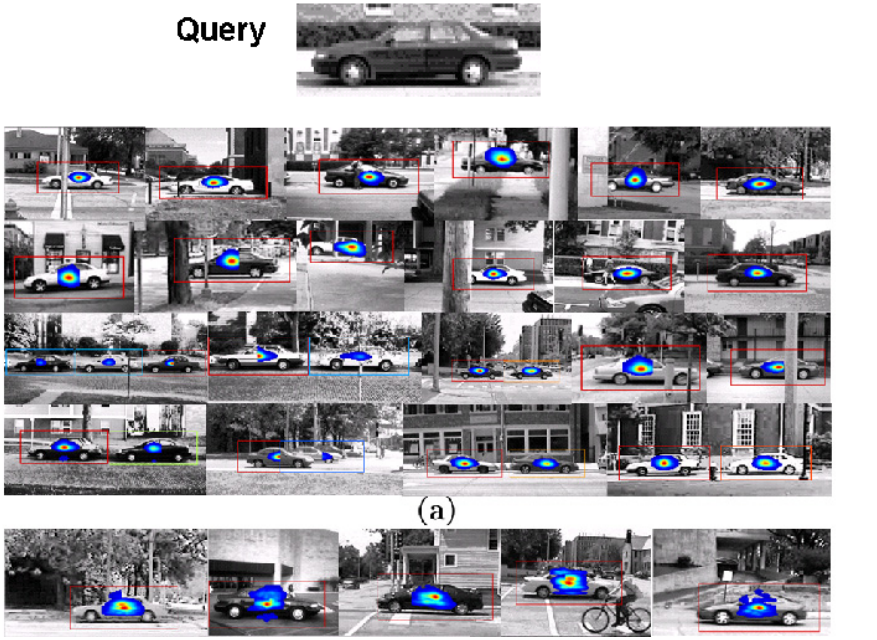
Ubiquitous Filtering: dealing with pixels, correspondences, **segments**, **objects**…

(a) Input (b) Output of Fleishman *et al.* [2003]

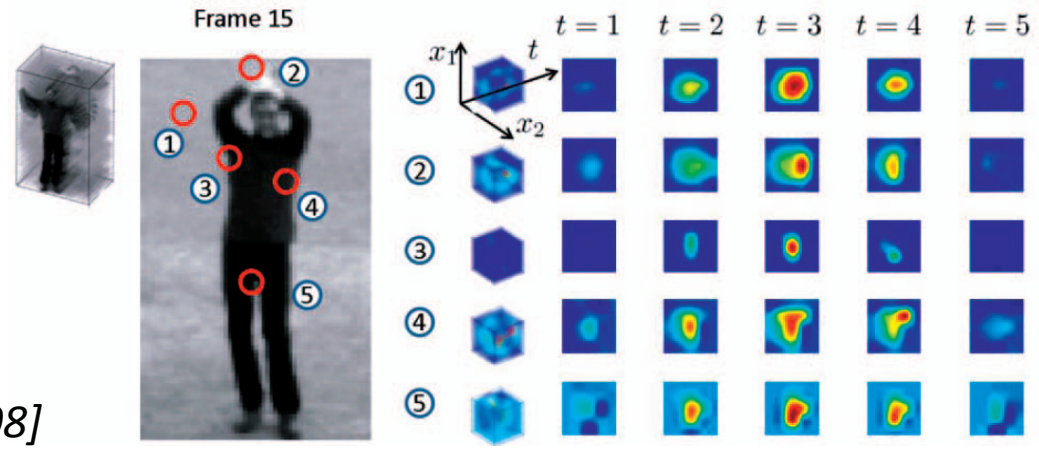
[Fleishman et al., SIGGRAPH'03]



[Wang et al., EG'08]



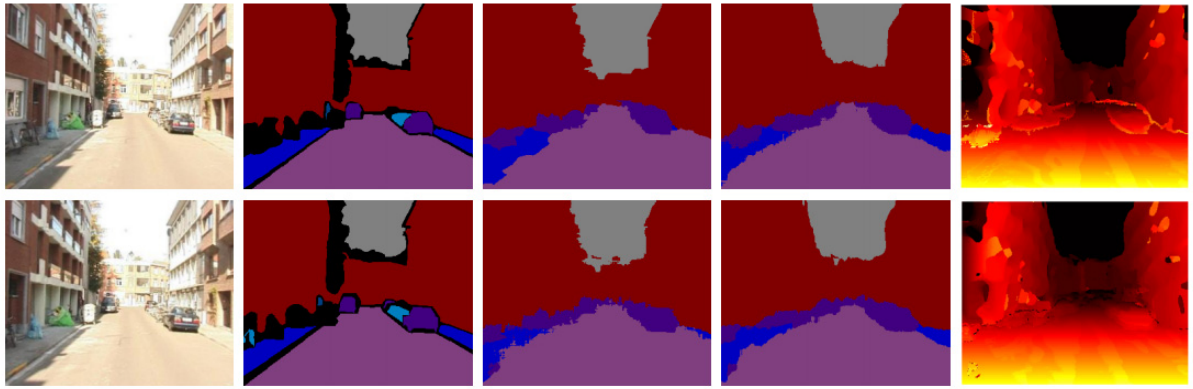
[Seo and Milanfar, PAMI'10 & 11]



Ubiquitous Filtering...

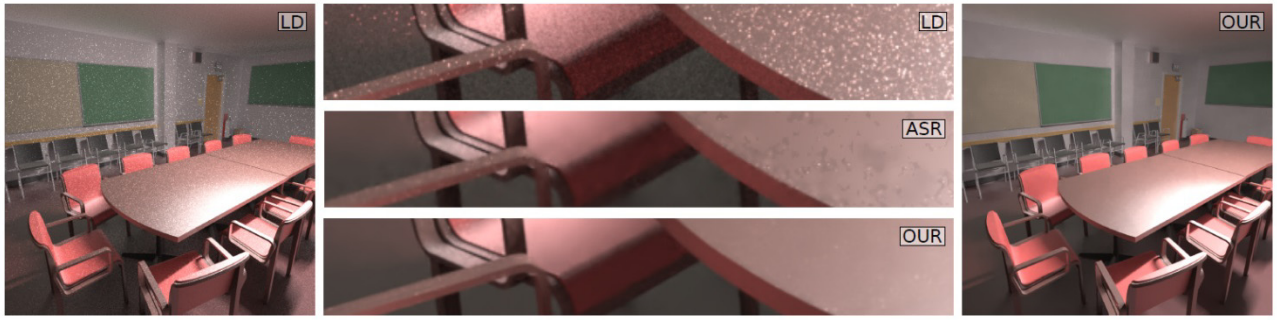
[Saliency Filters, CVPR'12]

High-order CRF terms: e.g. label consistency over large regions, global co-ocurrence relations. 10x-30x faster.



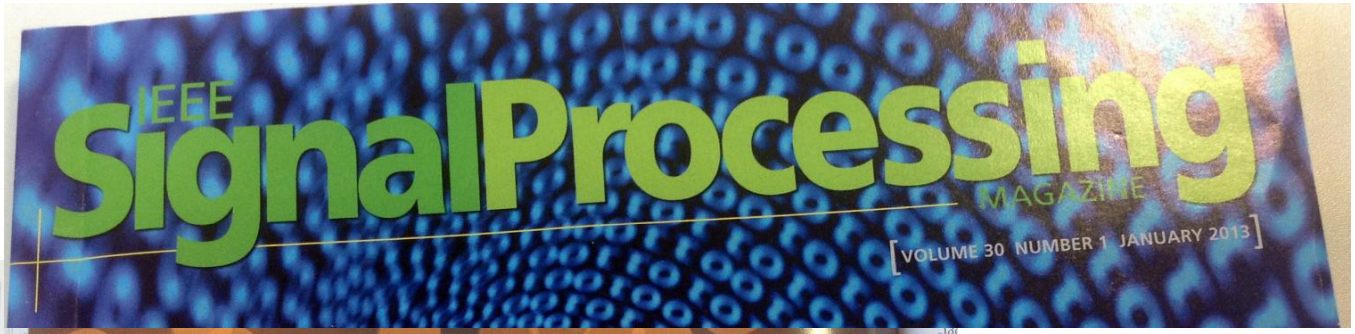
[Filter-based Mean-Field Inference, ECCV'12]

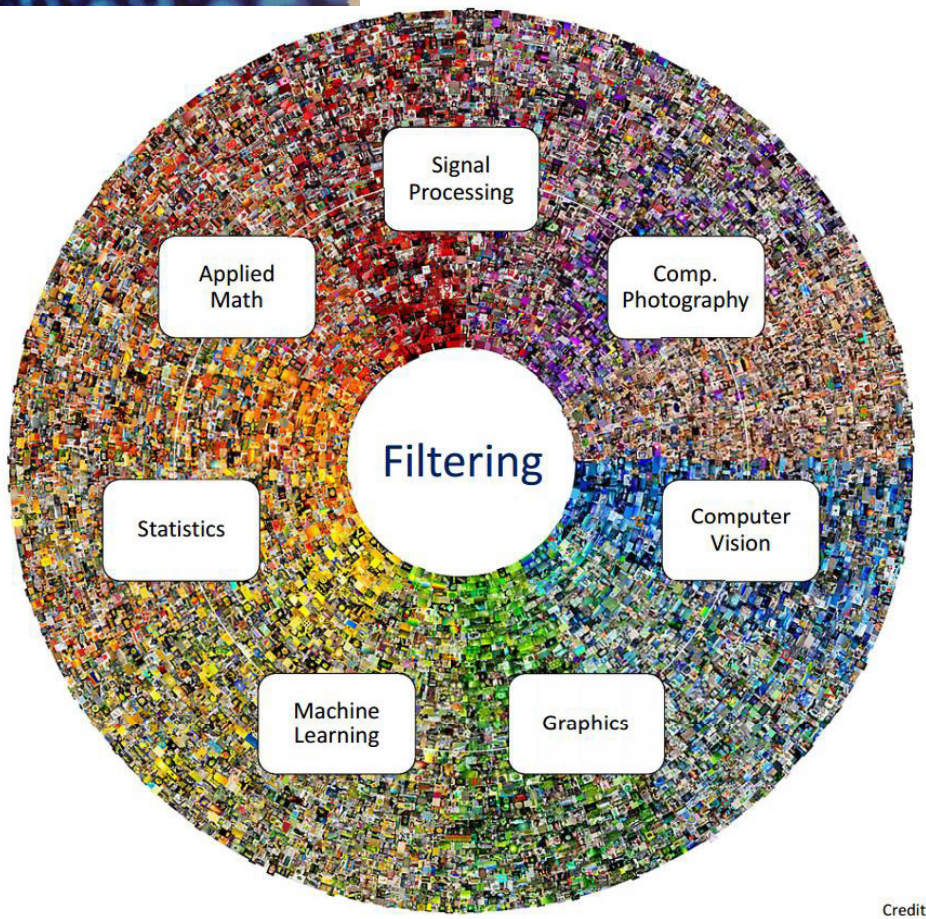
[Adaptive Manifold, SIG'12]



[Adaptive Rendering with NLM Filtering, SIG-Asia'12]

Image Filtering 2.0?





Credit