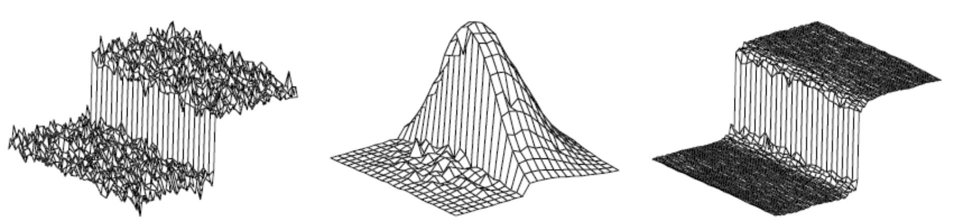
Fast Local Laplacian Filters: Theory and Applications

Mathieu Aubry (INRIA, ENPC),
Sylvain Paris (Adobe), Sam Hasinoff (Google),
Jan Kautz (UCL), and Frédo Durand (MIT)

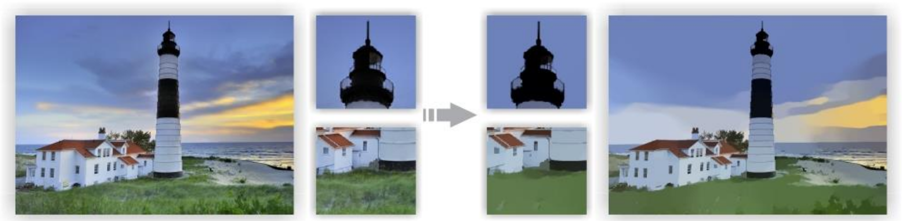
Input

Unsharp Mask, not edge-aware ☹

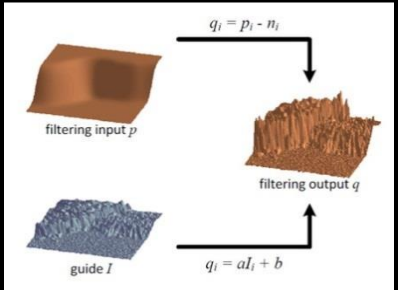
Edge-aware image processing



Bilateral Filter [Tomasi and Manduchi 1998]



L_0 **Gradient Minimization** [Xu et al. 2011]



Guided Image Filtering
[He et al. 2010]

Edge-aware wavelets
[Fattal 2009]

Adaptative Manifolds
[Gastal and Oliveira 2012]

See also [Fattal et al. 2002], [Farbman et al. 2008], [Subr et al. 2009], [Gastal and Oliveira 2011]…

Local Laplacian Filter, edge-aware ☺

[Paris et al. 2011]

- **No halos or gradient inversion**
- **Even for extreme edits**

Some limitations…

- Too slow for interactive editing: 4s/Mpixel
- Unknown relationship to other filters
- Only detail manipulation and tone mapping

Our contributions

- Too slow for interactive editing: 4s/Mpixel
  - **20x speed up**
- Unknown relationship to other filters
  - **Formal analysis and relation to Bilateral Filter**
- Only detail manipulation and tone mapping
  - **General gradient manipulations and style transfer**

Background on Gaussian Pyramids

- Resolution halved at each level using Gaussian kernel

level 0

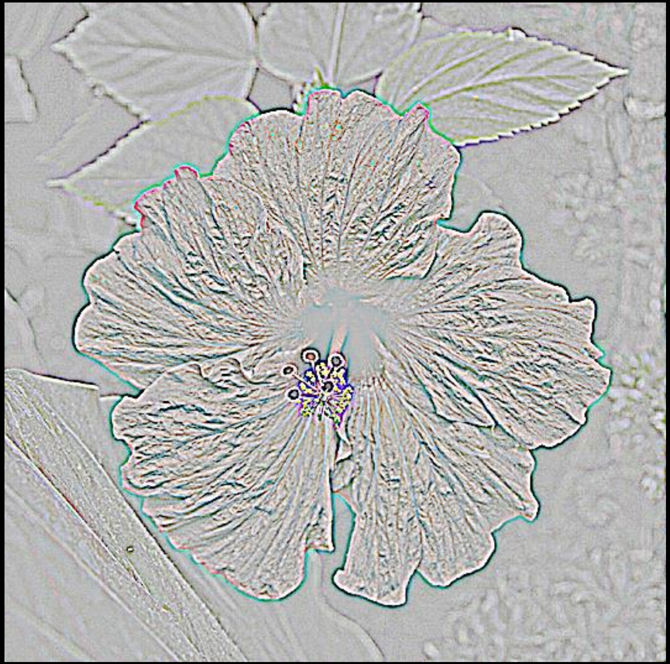
level 1

level 2

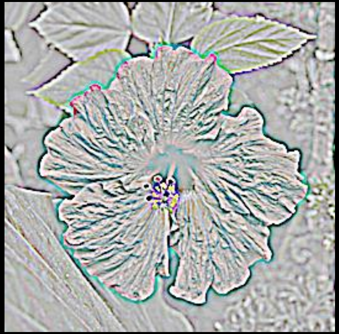
level 3
(residual)

Background on Laplacian Pyramids

- Difference between adjacent Gaussian levels



level 0



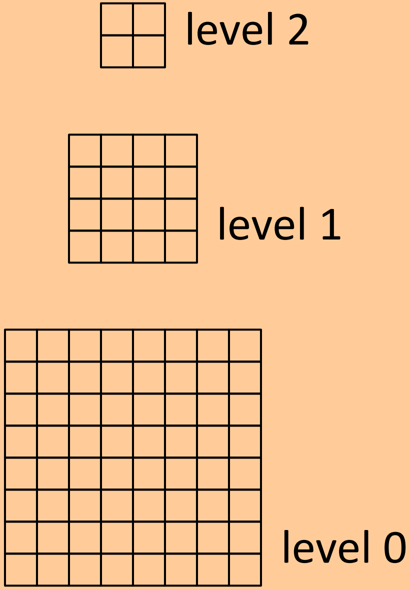
level 1

level 2

level 3
(residual)

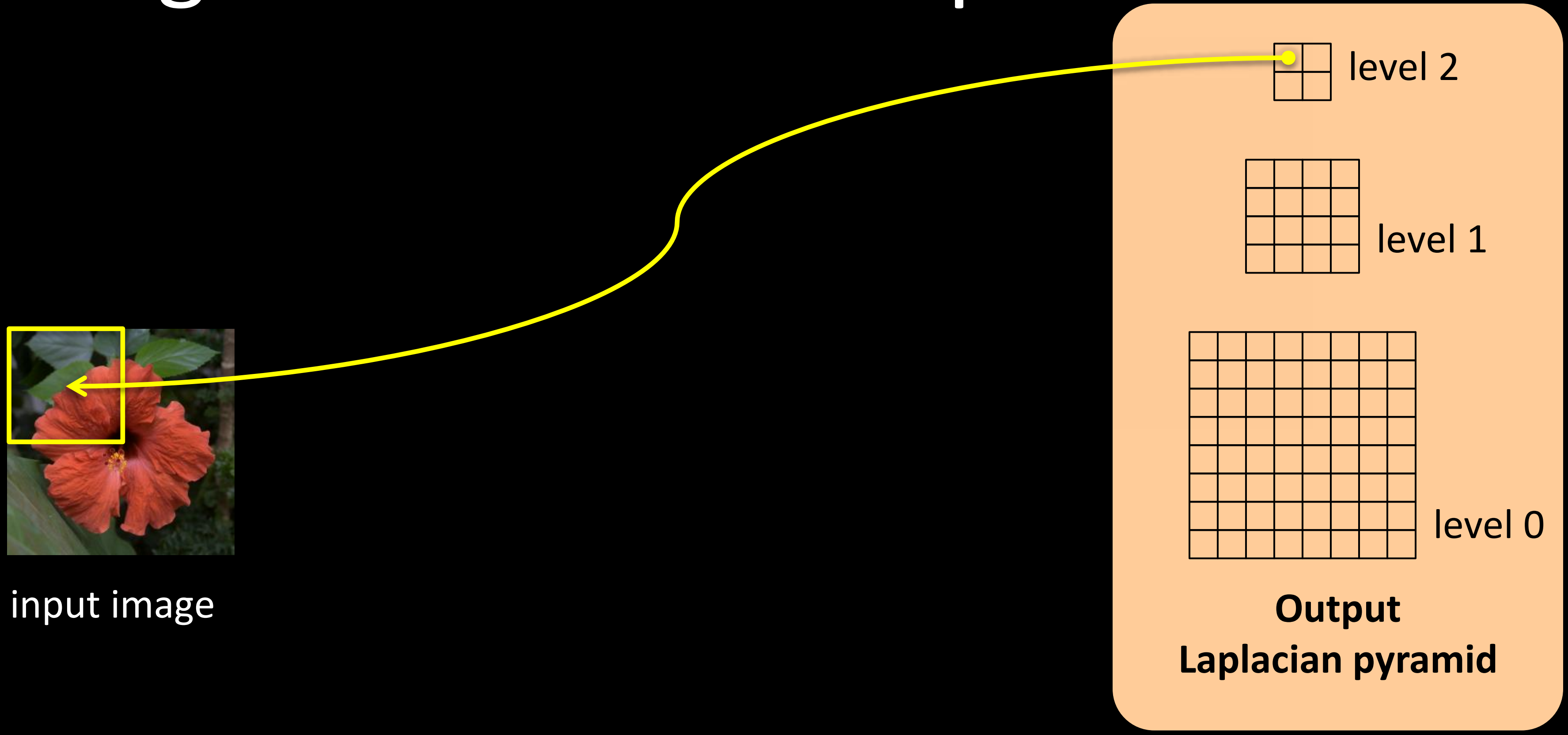
Background on Local Laplacian Filters

input image



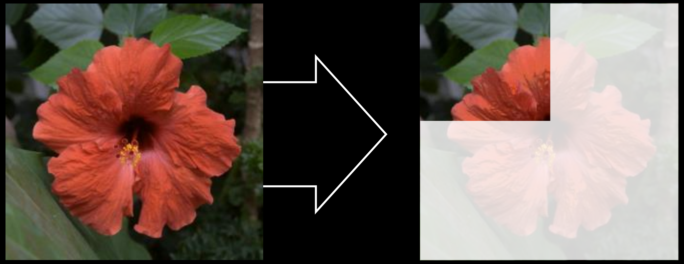
Output
Laplacian pyramid

Background on Local Laplacian Filters



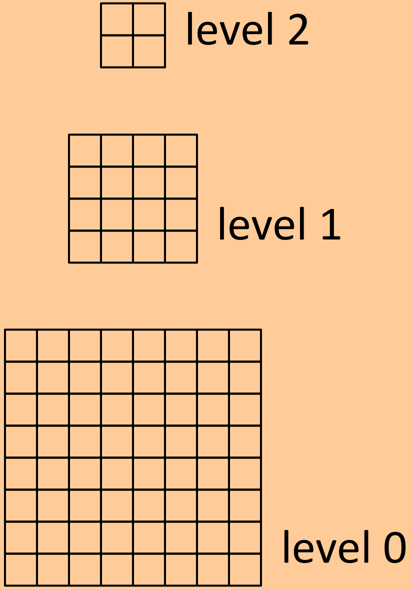
Background on Local Laplacian Filters

Local contrast manipulation



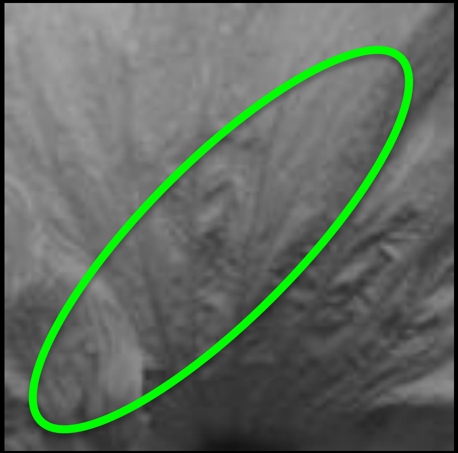
input image

locally processed image

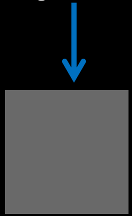


Output Laplacian pyramid

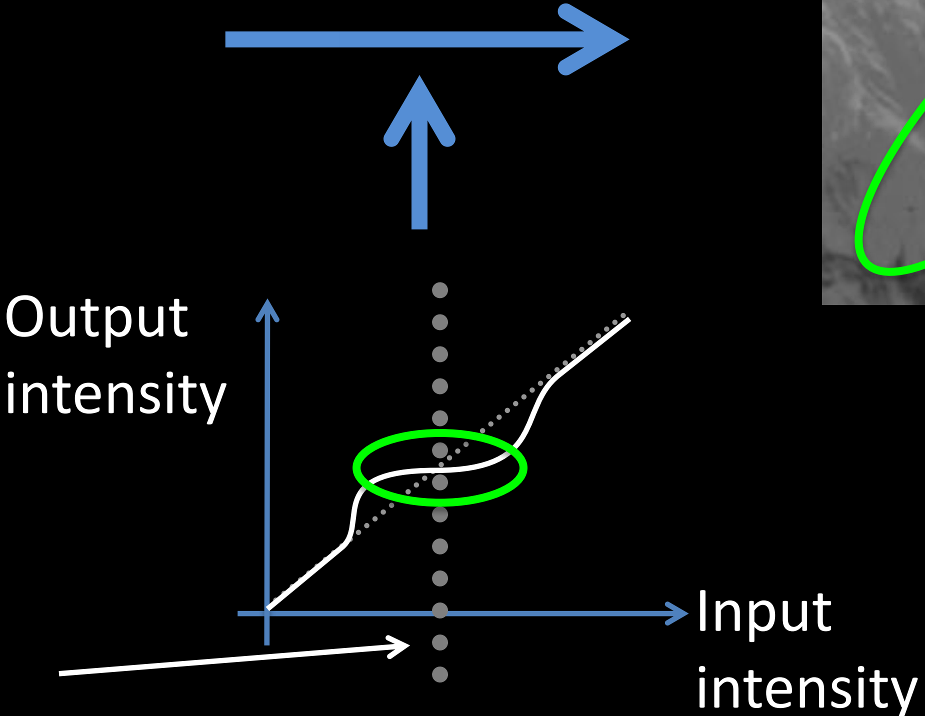
Background on Local Laplacian Filters

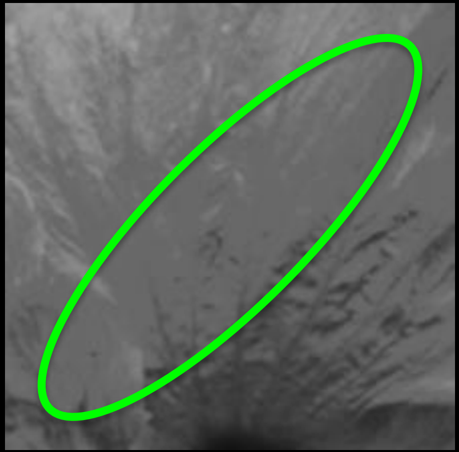


Input



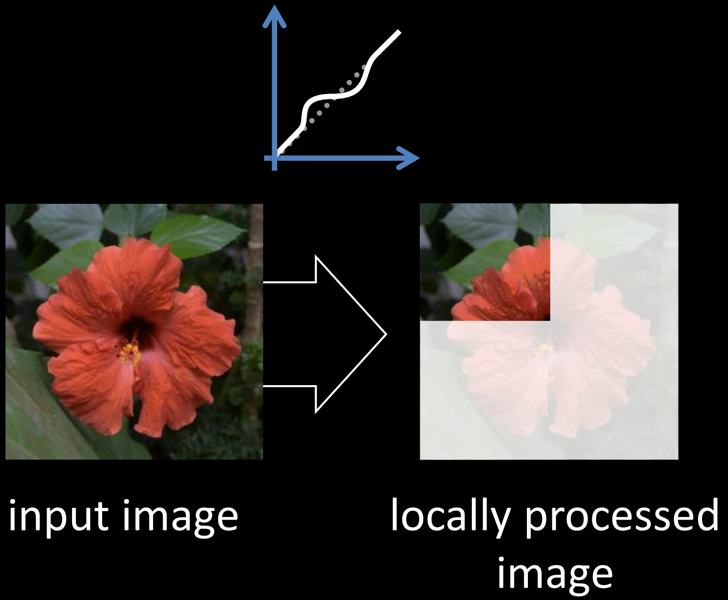
average

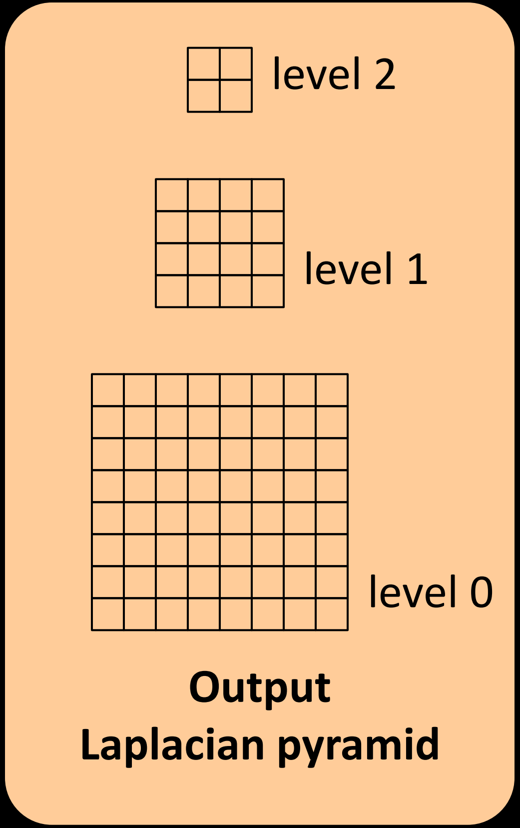




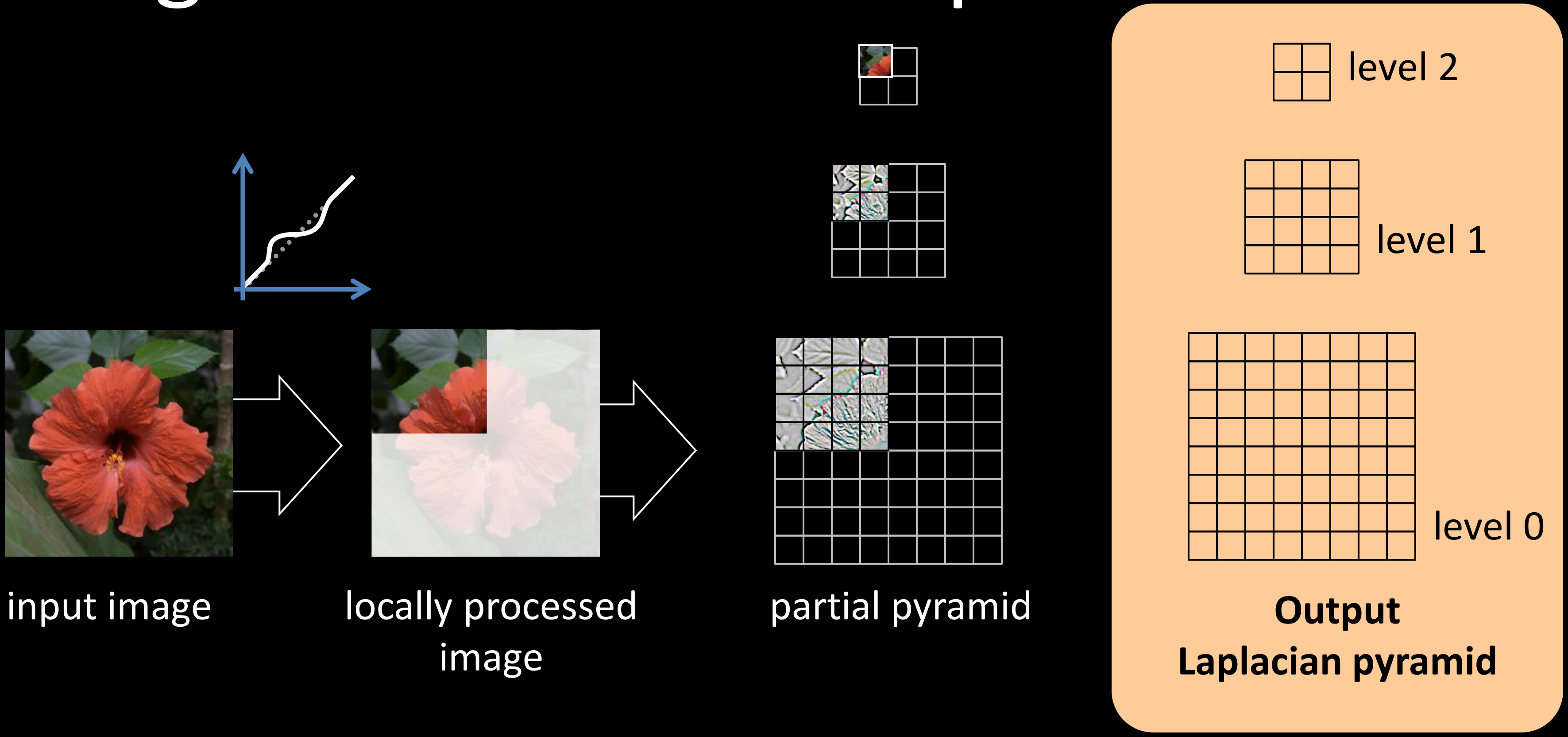
Output

Background on Local Laplacian Filters

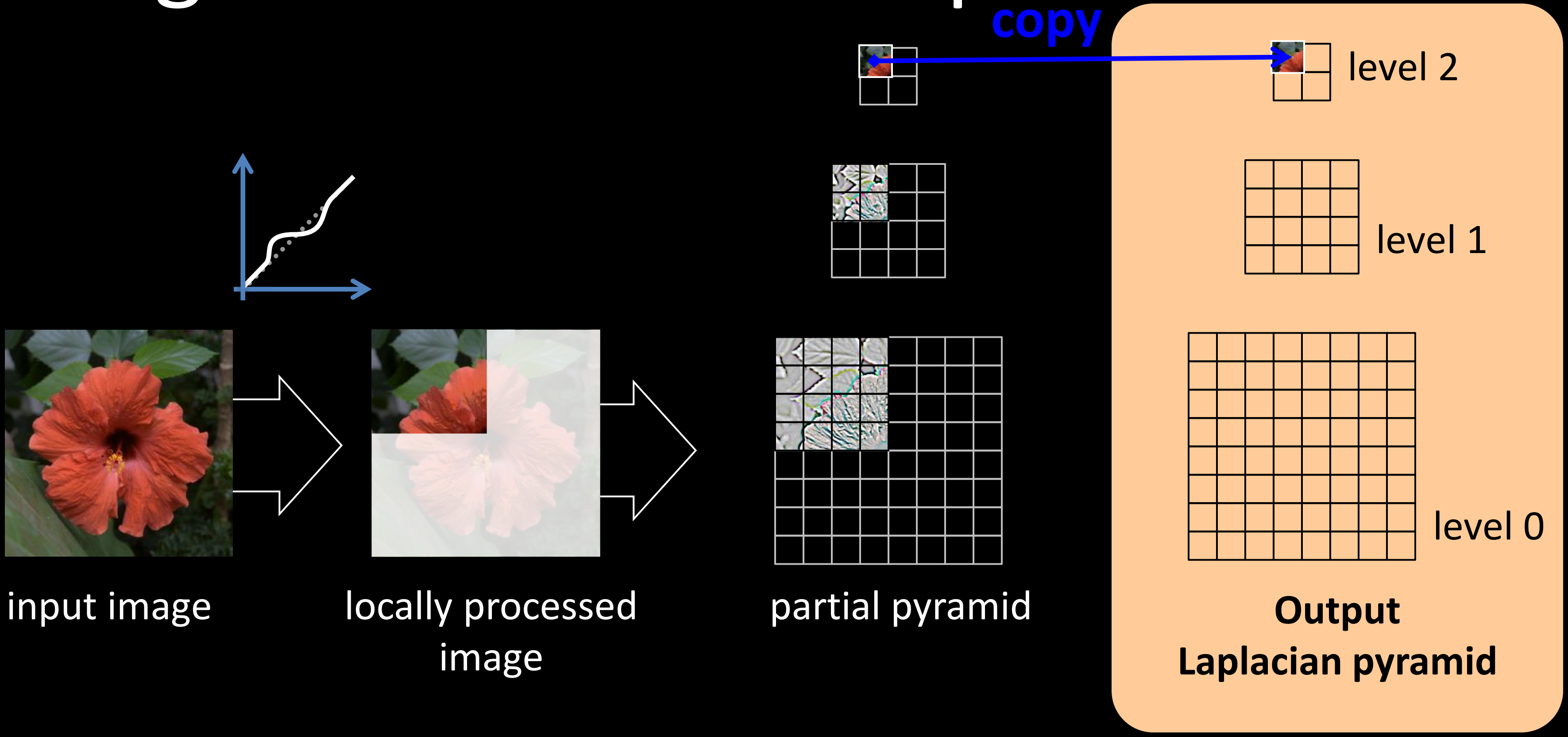




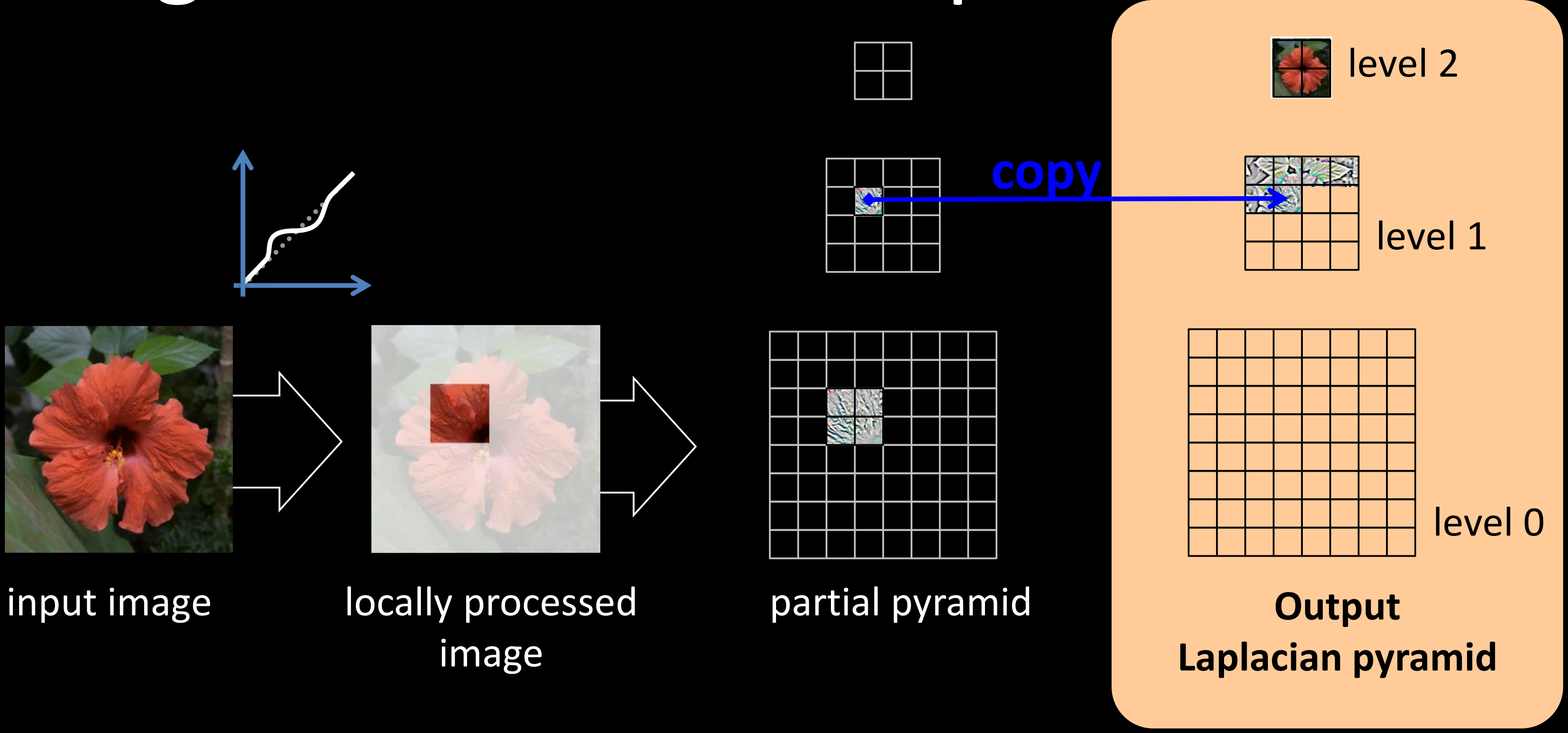
Background on Local Laplacian Filters



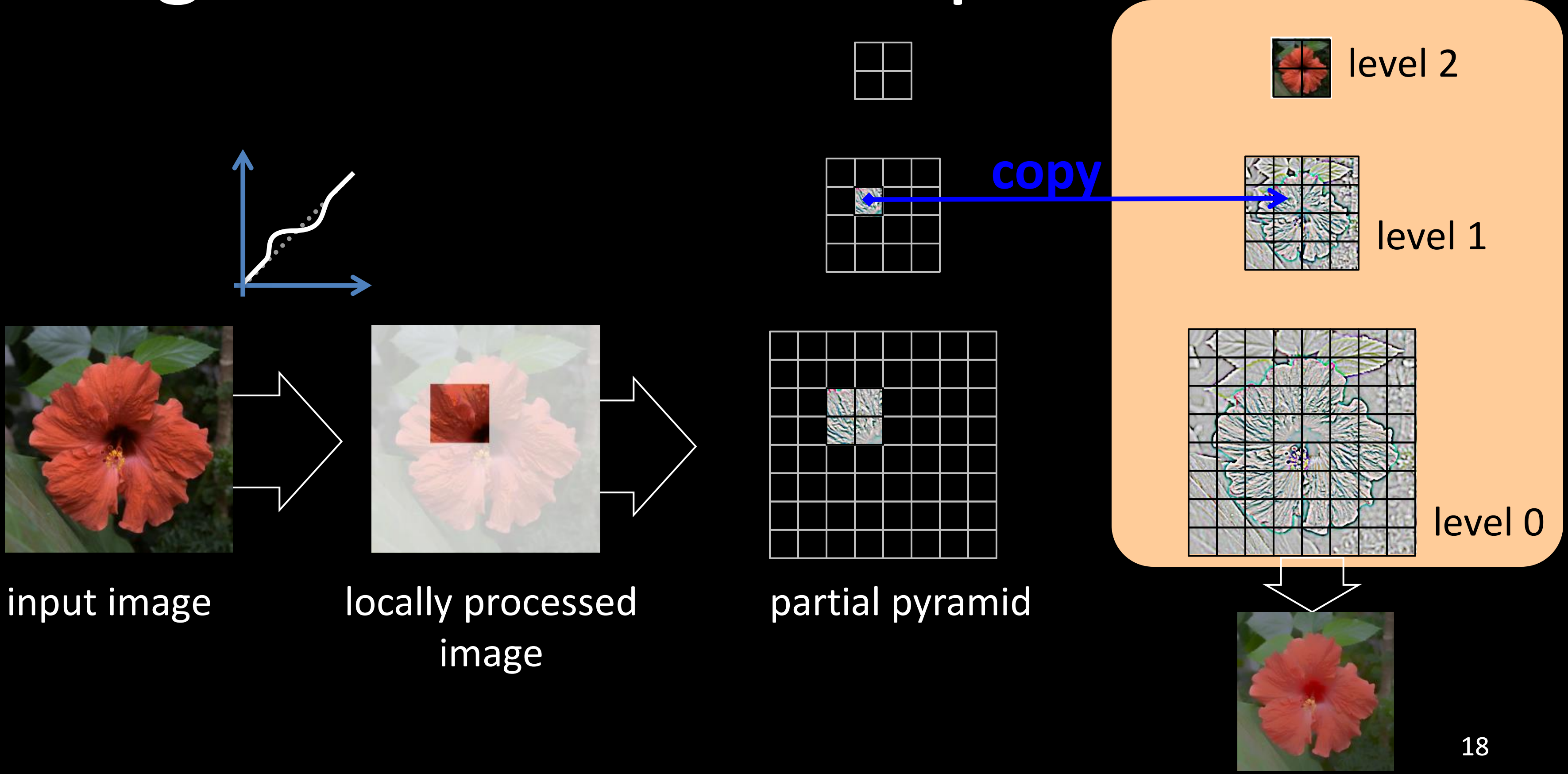
Background on Local Laplacian Filters



Background on Local Laplacian Filters

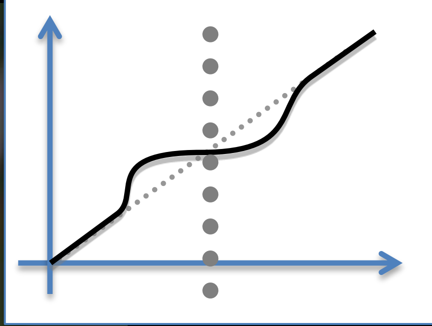


Background on Local Laplacian Filters

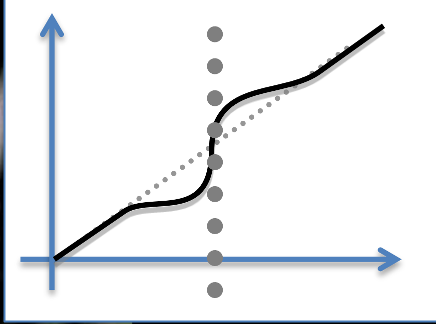



Input

Smoothing



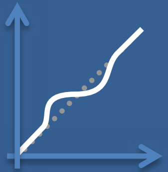
Enhancement



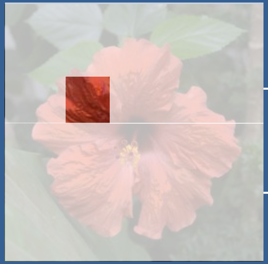
1. Speed up

One-level Local Laplacian Filter

$$i \to i - d(i - \boxed{g})$$



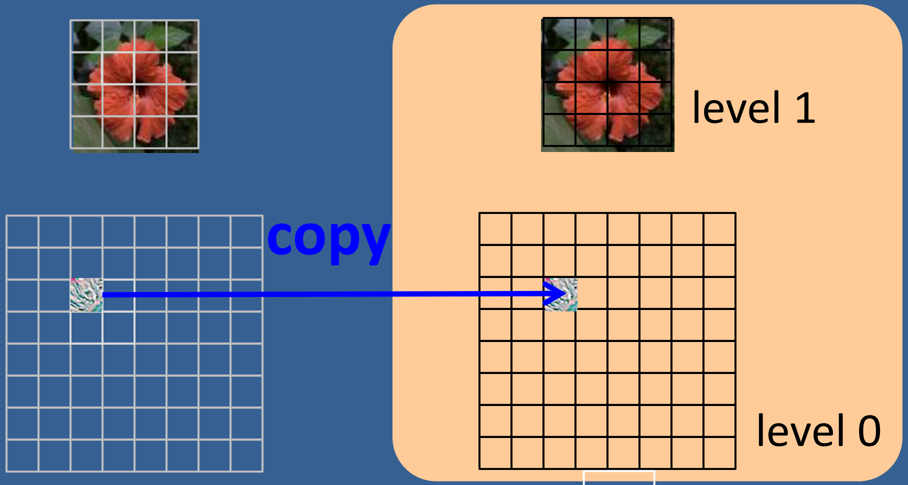
input image



locally processed image

STEP 1: INTENSITY REMAPPING

$$I \to I - G_\sigma * I$$



partial pyramid

STEP 2: PYRAMID

One-level Local Laplacian Filter

$$i \rightarrow i - d(i - g) \qquad\qquad I \rightarrow I - G_\sigma * I$$

$$\boxed{O_p} = \boxed{I_p} + \boxed{\textstyle\sum_q} \boxed{G_\sigma(q - p)} \boxed{d(I_q - I_p)}$$

Output image

Input image

Local sum

Gaussian spatial weight

Influence from intensity difference

Why is it slow?

$$i \to i - \boxed{d(i - g)}$$

$$I \to I - G_\sigma * I$$

$$O_p = I_p + \sum_q G_\sigma(q - p)\boxed{d(I_q - I_p)}$$

For each neighborhood

For each pixel

➢ **Computed #neighborhood X #pixels**

Speed up

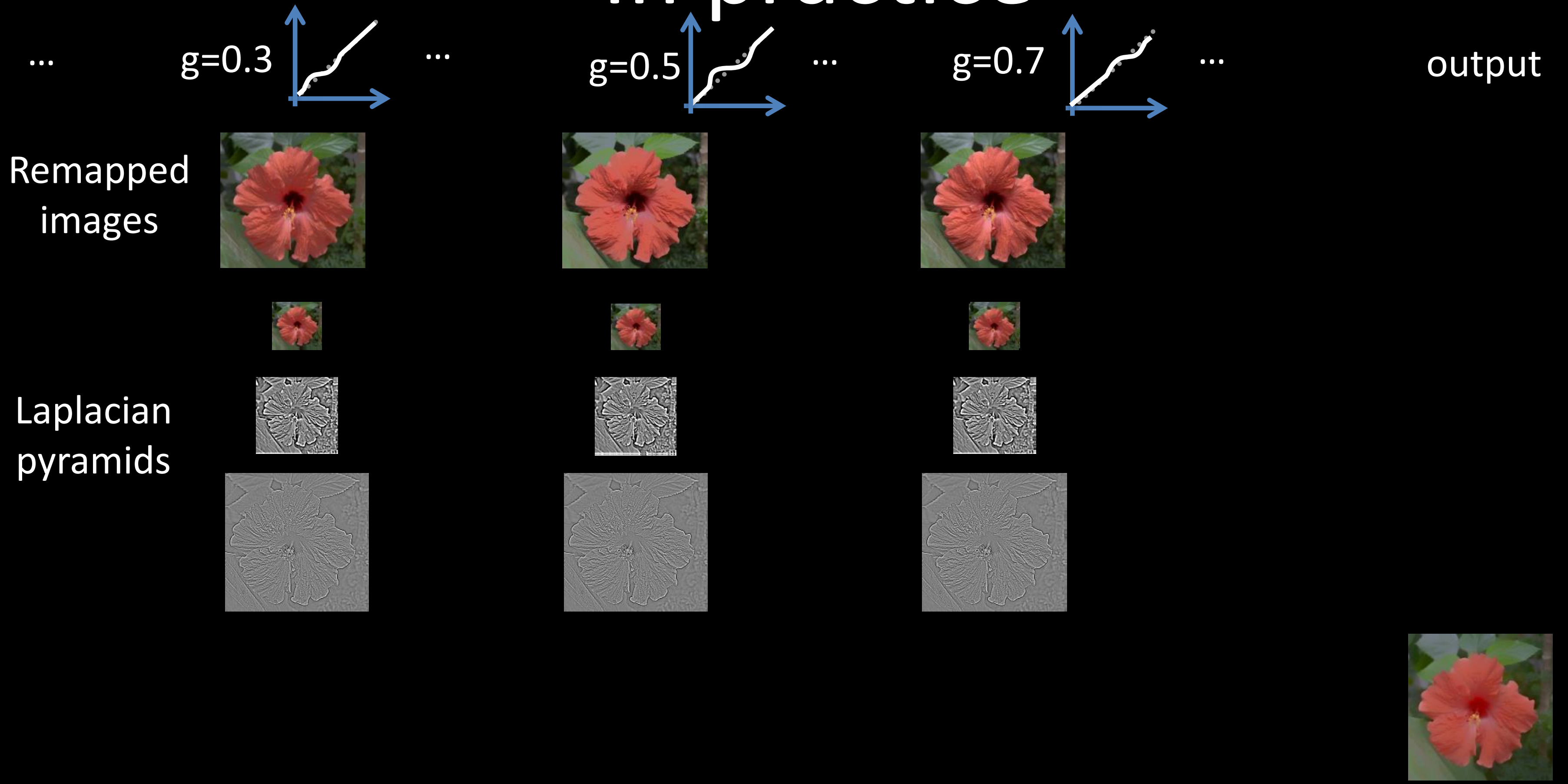
$$\boxed{d(i-g)}$$

Idea: if g were constant, we would need to compute d only once per pixel

➢Compute d only for a small set of values of g and interpolate

➢ Compute d **K x #pixels**

In practice



Performance

	[Paris 2011]	Our method	Speed up
1Mpixel CPU	15 s	350 ms	50x
4Mpixel GPU	1 s	49 ms	20x

Suitable for interactive editing

- implemented in Lightroom/Photoshop

Input Image

Ground truth enhancement

Our method with 20 values

Our method with 10 values

Our method with 5 values

2. Relation to Bilateral Filter

Interpretation

Bilateral Filter

Spatial weight

Weighted intensities

$$BF_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}} \boxed{G_{\sigma_s}(\mathbf{q}-\mathbf{p})}\boxed{G_{\sigma_r}(I_{\mathbf{q}} - I_{\mathbf{p}})I_{\mathbf{q}}}$$

One-level Local Laplacian Filter

$$O_{\mathbf{p}} = I_{\mathbf{p}} + \sum_{\mathbf{q}} \boxed{G_{\sigma}(\mathbf{q}-\mathbf{p})}\boxed{d(I_{\mathbf{q}} - I_{\mathbf{p}})}$$

Spatial weight from pyramid

Remapping function

Interpretation

Bilateral Filter

Spatial weight

Weighted intensities

$$BF_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}} \boxed{G_{\sigma_s}(\mathbf{q}-\mathbf{p})}\boxed{G_{\sigma_r}(I_{\mathbf{q}} - I_{\mathbf{p}})I_{\mathbf{q}}}$$

One-level Local Laplacian Filter

$$O_{\mathbf{p}} = I_{\mathbf{p}} + \sum_{\mathbf{q}} \boxed{G_{\sigma}(\mathbf{q}-\mathbf{p})}\boxed{G_{\sigma_r}(I_{\mathbf{q}} - I_{\mathbf{p}})(I_{\mathbf{q}} - I_{\mathbf{p}})}$$

Spatial weight from pyramid

Remapping function


Power function


Gaussian

Interpretation

Bilateral Filter

Spatial weight

Weighted intensities

$$BF_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}} \boxed{G_{\sigma_s}(\mathbf{q}-\mathbf{p})} \boxed{G_{\sigma_r}(I_{\mathbf{q}} - I_{\mathbf{p}}) I_{\mathbf{q}}}$$

One-level Local Laplacian Filter

$$O_{\mathbf{p}} = \boxed{I_{\mathbf{p}}} + \sum_{\mathbf{q}} \boxed{G_{\sigma}(\mathbf{q}-\mathbf{p})} \boxed{G_{\sigma_r}(I_{\mathbf{q}} - I_{\mathbf{p}})(I_{\mathbf{q}} \boxed{- I_{\mathbf{p}}})}$$

Spatial weight from pyramid

Remapping function

Rewriting the bilateral filter

Bilateral Filter

Weights sum to 1

$$BF_{\mathbf{p}} = \boxed{\frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}} G_{\sigma_s}(\mathbf{q} - \mathbf{p}) G_{\sigma_r}(I_{\mathbf{q}} - I_{\mathbf{p}})} I_{\mathbf{q}}$$

$$BF_{\mathbf{p}} = \boxed{I_{\mathbf{p}}} + \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}} G_{\sigma_s}(\mathbf{q} - \mathbf{p}) G_{\sigma_r}(I_{\mathbf{q}} - I_{\mathbf{p}})(I_{\mathbf{q}} \boxed{- I_{\mathbf{p}}})$$

Interpretation

Bilateral Filter

Spatial weight

Weighted intensities

One-level Local Laplacian Filter

$$O_{\mathbf{p}} = \boxed{I_{\mathbf{p}}} + \sum_{\mathbf{q}} \boxed{G_{\sigma}(\mathbf{q} - \mathbf{p})} \boxed{G_{\sigma_r}(I_{\mathbf{q}} - I_{\mathbf{p}})(I_{\mathbf{q}} - I_{\mathbf{p}})}$$

Original image

Spatial weight from pyramid

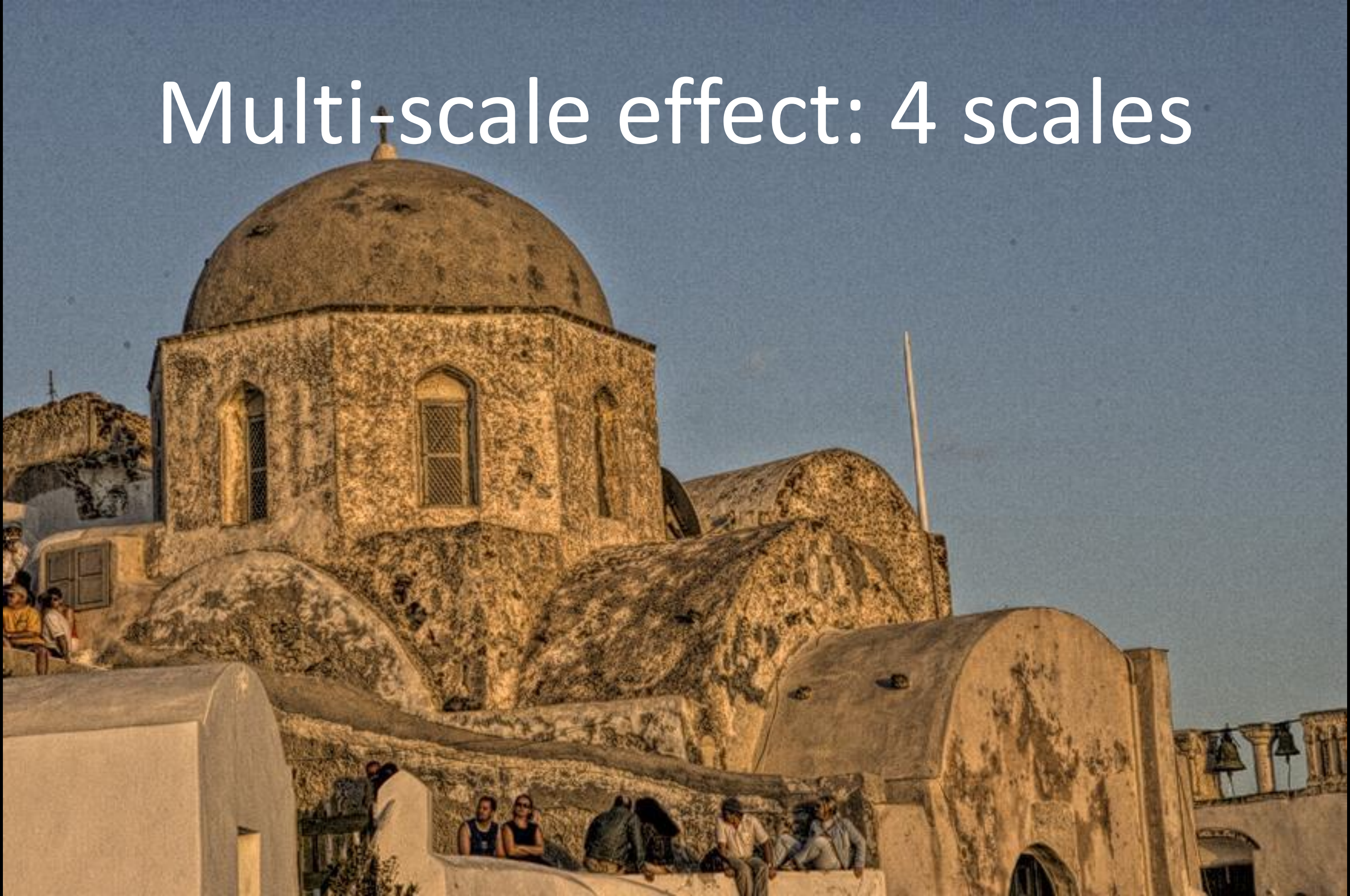
Remapping function

Multi-scale effect: input

Multi-scale effect: 1 scale

Multi-scale effect: 2 scales

Multi-scale effect: 4 scales



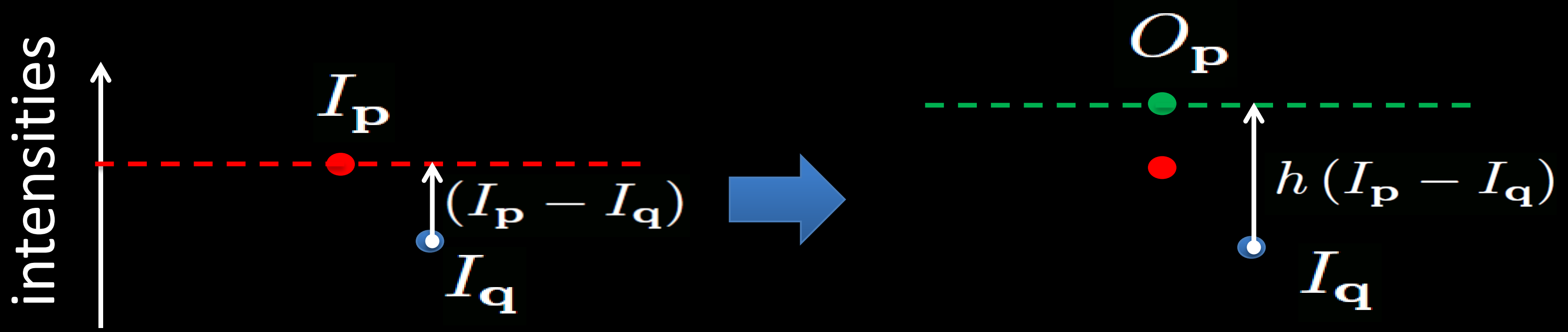
Multi-scale effect: 8 scales

3. Style transfer

Local statistics manipulation

Interpret the remapping function as a remapping of pixel differences

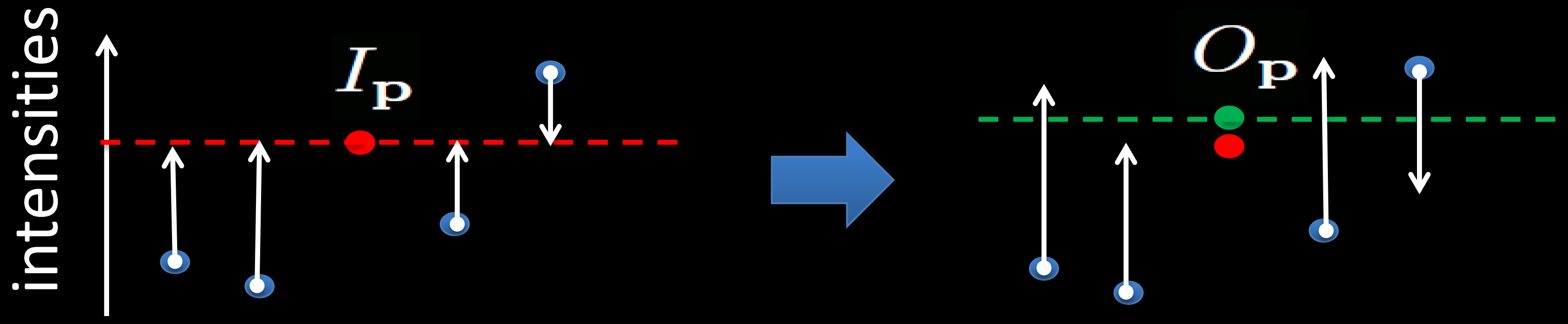
Single-neighbor case



Local statistics manipulation

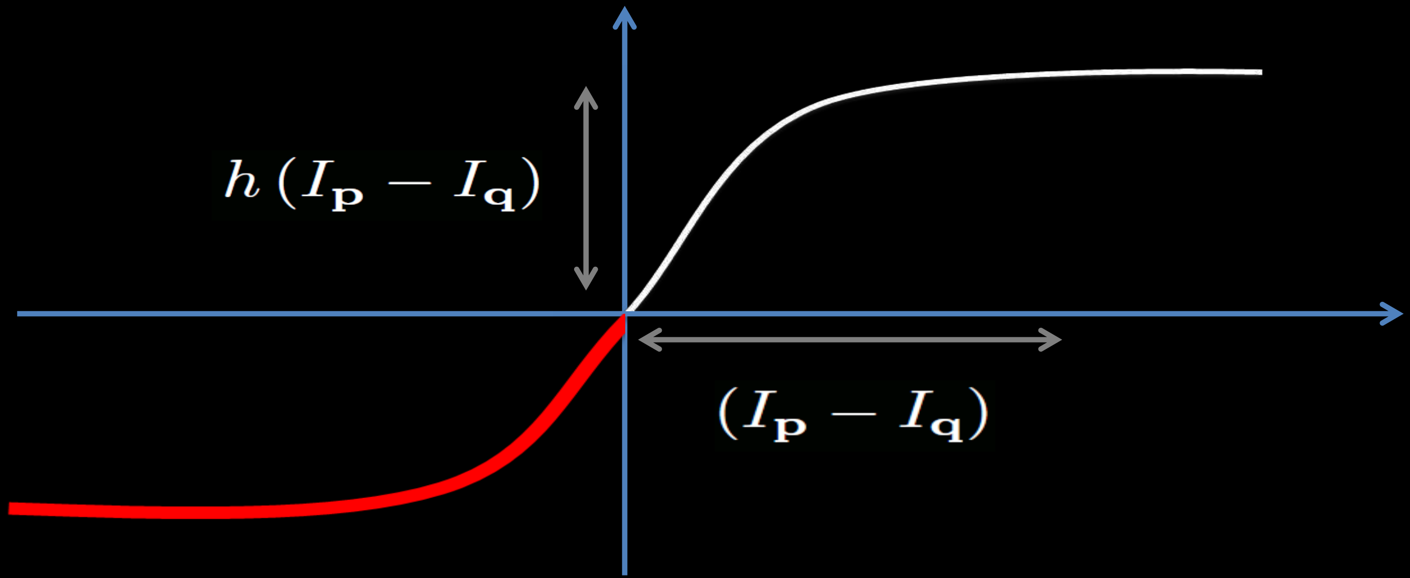
Many neighboors case

Can be interpreted as averaging target differences



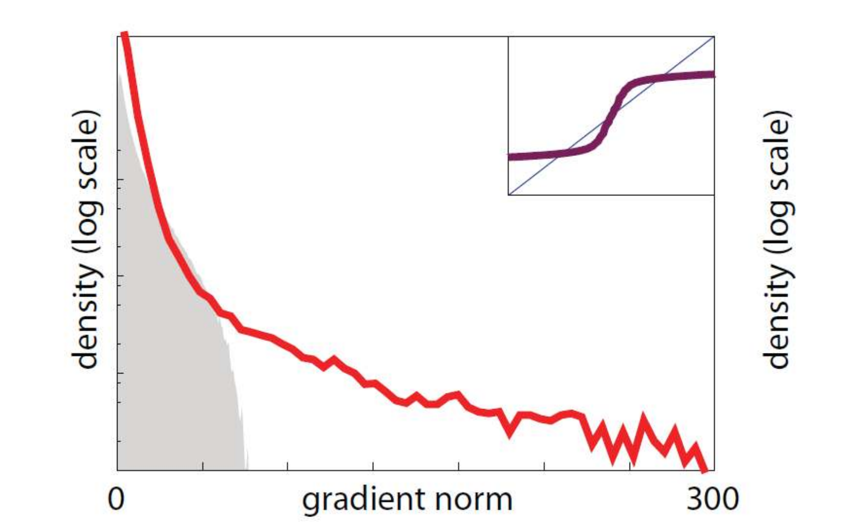
Local statistics manipulation

- *h* controls how the gradients are remapped

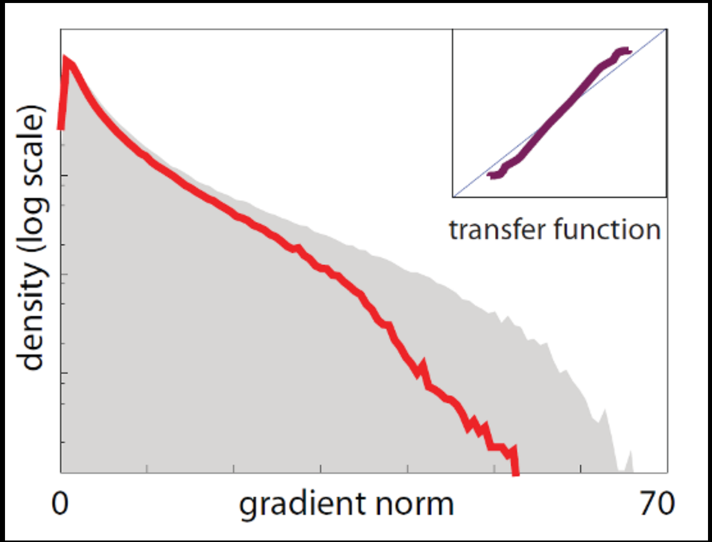


➢ Use histogram transfer function to define *h*

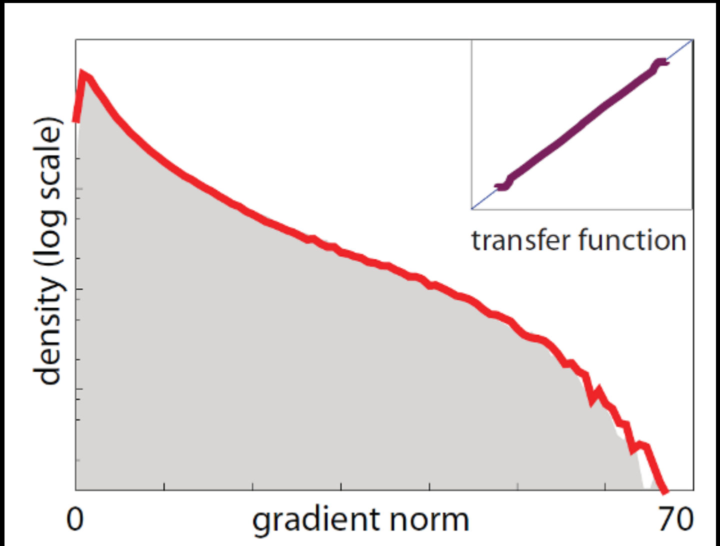
Example:



Example: iteration 1



Example: iteration 2




Also in the paper

- Link with PDEs / Anisotropic diffusion
- Introduction of Un-normalized Bilateral Filter
  - Discussion of effect on edges
- More results and comparisons
  - Quantitative evaluations of transfer

Conclusion

- 20x to 50x speed-up
  - in Lightroom and Photoshop
- Relationship with BF and PDE
- Gradient histogram transfer
  - Photographic style transfer

Matlab **code** and more results:

http://www.di.ens.fr/~aubry/llf.html

We would like to thank…

- **Mark Fairchild** for his HDR survey
- The **anonymous reviewers** for their constructive comments
- **Adobe** for its gifts to Jan Kautz, Sam Hasinoff and Frédo Durand

Conclusion

- Relationship with BF and PDE
- 20x to 50x speed-up
  - in Lightroom and Photoshop
- Gradient histogram transfer
  - Photographic style transfer

Matlab **code** and more results:

http://www.di.ens.fr/~aubry/llf.html