Fast Local Laplacian Filters: Theory and Applications

Mathieu Aubry (INRIA, ENPC), Sylvain Paris (Adobe), Sam Hasinoff (Google), Jan Kautz (UCL), and Frédo Durand (MIT)
Unsharp Mask, not edge-aware 😞
Edge-aware image processing

- Bilateral Filter [Tomasi and Manduchi 1998]
- Guided Image Filtering [He et al. 2010]
- $L_0$ Gradient Minimization [Xu et al. 2011]
- 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
Background on Laplacian Pyramids

• Difference between adjacent Gaussian levels
Background on Local Laplacian Filters

Input image

Output Laplacian pyramid
Background on Local Laplacian Filters

Input image

Output Laplacian pyramid

Level 0

Level 1

Level 2
Background on Local Laplacian Filters

Local contrast manipulation

Input image → Locally processed image

Output Laplacian pyramid

Level 0 → Level 1 → Level 2
Background on Local Laplacian Filters

Input
average

Output intensity

Input intensity

Output
Background on Local Laplacian Filters

- Input image
- Locally processed image
- Level 0 Laplacian pyramid
- Level 1 Laplacian pyramid
- Level 2 Laplacian pyramid

Output Laplacian pyramid
Background on Local Laplacian Filters

- Input image
- Locally processed image
- Partial pyramid

Output Laplacian pyramid
Background on Local Laplacian Filters

- Input image
- Locally processed image
- Partial pyramid
- Laplacian pyramid

Output
Laplacian pyramid

Levels:
- Level 0
- Level 1
- Level 2

Copy
Background on Local Laplacian Filters

Input image → locally processed image → partial pyramid

Output Laplacian pyramid

Levels:
- Level 0
- Level 1
- Level 2

Copy mechanism for pyramid construction.
Background on Local Laplacian Filters

input image → locally processed image → partial pyramid

level 0

level 1

level 2

copy
Smoothing
Enhancement
1. Speed up
**One-level Local Laplacian Filter**

$$i \rightarrow i - d(i - g)$$

**STEP 1: INTENSITY REMAPPING**

**STEP 2: PYRAMID**

$$I \rightarrow I - G_\sigma * I$$
One-level Local Laplacian Filter

\[ i \rightarrow i - d(i - g) \quad \text{and} \quad I \rightarrow I - G_\sigma * I \]

\[ O_p = I_p + \sum_q G_\sigma(q - p) d(I_q - I_p) \]

- **Output image**
- **Input image**
- **Local sum**
- **Gaussian spatial weight**
- **Influence from intensity difference**
Why is it slow?

For each pixel

\[ i \to i - d(i - g) \]

For each neighborhood

\[ I \to I - G_\sigma \ast I \]

\[ O_p = I_p + \sum_q G_\sigma (q - p) d(I_q - I_p) \]

For each neighborhood \( X \) pixels

Computed \#neighborhood \( X \) #pixels
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 $d$ $K \times \#\text{pixels}$.
In practice

Remapped images

Laplacian pyramids

\[ g = 0.3, 0.5, 0.7 \]
### Performance

<table>
<thead>
<tr>
<th></th>
<th>[Paris 2011]</th>
<th>Our method</th>
<th>Speed up</th>
</tr>
</thead>
<tbody>
<tr>
<td>1Mpixel CPU</td>
<td>15 s</td>
<td>350 ms</td>
<td>50x</td>
</tr>
<tr>
<td>4Mpixel GPU</td>
<td>1 s</td>
<td>49 ms</td>
<td>20x</td>
</tr>
</tbody>
</table>

Suitable for interactive editing

- implemented in Lightroom/Photoshop
Ground truth enhancement
Our method with 20 values
Our method with 10 values
Our method with 5 values
2. Relation to Bilateral Filter
Bilateral Filter

\[
BF_p = \frac{1}{W_p} \sum_q G_{\sigma_s}(q - p) G_{\sigma_r}(I_q - I_p) I_q
\]

One-level Local Laplacian Filter

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

Spatial weight from pyramid

Weighted intensities

Remapping function
Interpretation

Bilateral Filter

\[ BF_p = \frac{1}{W_p} \sum_q G_{\sigma_s}(q - p) G_{\sigma_r}(I_q - I_p) I_q \]

One-level Local Laplacian Filter

\[ O_p = I_p + \sum_q G_{\sigma}(q - p) G_{\sigma_r}(I_q - I_p)(I_q - I_p) \]

Spatial weight

Weighted intensities

Spatial weight from pyramid

Remapping function
Bilateral Filter

\[ BF_p = \frac{1}{W_p} \sum_{q} G_{\sigma_s}(q - p) G_{\sigma_r}(I_q - I_p) I_q \]

One-level Local Laplacian Filter

\[ O_p = I_p + \sum_{q} G_{\sigma}(q - p) G_{\sigma_r}(I_q - I_p)(I_q - I_p) \]
Rewriting the bilateral filter

\[ BF_p = \frac{1}{W_p} \sum_q G_{\sigma_s}(q - p)G_{\sigma_r}(I_q - I_p)I_q \]

\[ BF_p = I_p + \frac{1}{W_p} \sum_q G_{\sigma_s}(q - p)G_{\sigma_r}(I_q - I_p)(I_q - I_p) \]
Bilateral Filter

One-level Local Laplacian Filter

\[ O_p = I_p + \sum_{q} G_\sigma(q - p) G_{\sigma_r}(I_q - I_p)(I_q - I_p) \]

Interpretation

Original image

Spatial weight from pyramid

Weighted intensities

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 neighbors case

Can be interpreted as averaging target differences
Local statistics manipulation

- $h$ controls how the gradients are remapped

$\Rightarrow$ 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