Local Laplacian Filters

Mohammad Rami Koujan, Gourab Ghosh Roy, Raabid Hussain

Instructor: Dr. Alexander Belyaev

Novermber 22^{nd} 2016

Outline

Local Laplacian Filters (LLF)

What is LLF method?

- Edge-aware image processing based on Laplacian pyramids to achieve:
  1. Edge-preserving smoothing
  2. Detail enhancement
  3. Tone mapping
  4. Inverse tone mapping

Vibot

Gaussian and Laplacian Pyramids

Gaussian Pyramid

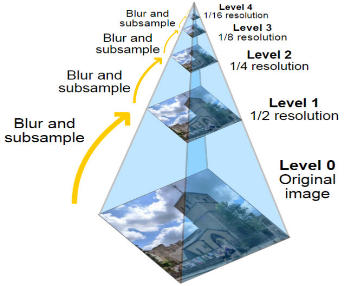


Figure 1: Visual representation of an image pyramid with 5 levels [a]

[a] Image courtesy of `https://commons.wikimedia.org/wiki/File:Image_pyramid.svg`

Gaussian and Laplacian Pyramids

Example

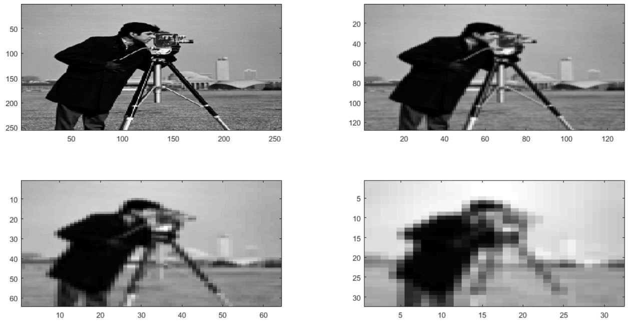


Figure 2: 4 levels of Gaussian smoothing of cameraman image

Gaussian and Laplacian Pyramids

Laplacian Pyramid

- $\{L_l\}$: representing details that distinguish successive levels of Gaussian pyramid
- $L_l{=}G_l{-}$upsample(G_{l+1})
- upsample(.): doubles the image size in each dimension
- $Ln{=}G_n$

Gaussian and Laplacian Pyramids

Laplacian Pyramid

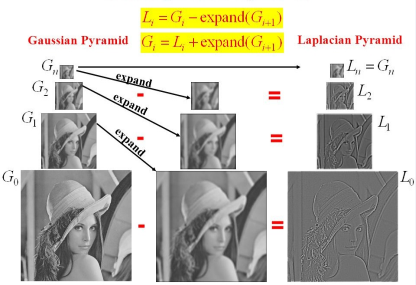


Figure 3: Visual representation of how to construct Laplacian pyramid [a]

[a] Image courtesy of https://commons.wikimedia.org/wiki/File:Image_pyramid.svg

Gaussian and Laplacian Pyramids

Example

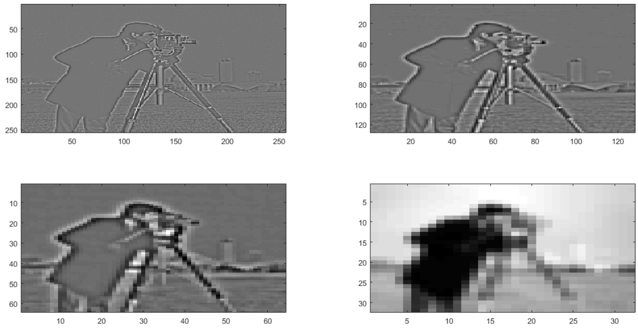


Figure 4: 4 levels of Laplacian pyramid of cameraman image

Edges in Laplacian Pyramids

1-D signal case

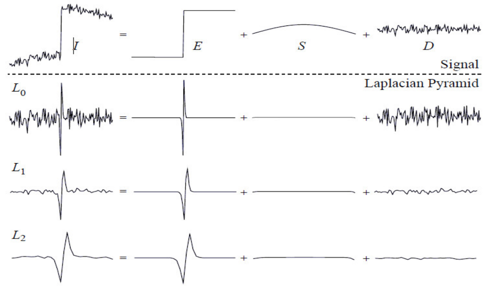


Figure 5: The 1D signal I is decomposed into three main components: a step edge E, a slowly varying signal S, and a high-frequency detail signal D. The Laplacian pyramid representation (three levels) is shown for the signal and its individual components

Range Compression

Range Compression

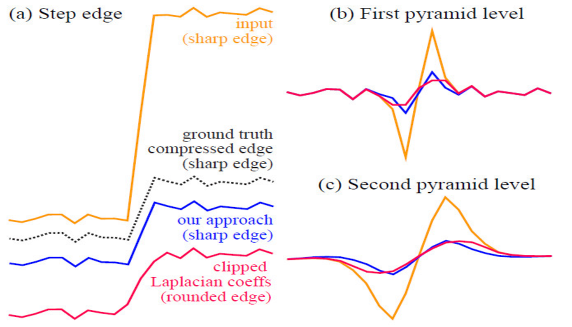


Figure 6: Range compression applied to a step edge with fine details

Local Processing

Incorporating Local Processing

- Construction of Laplacian pyramid for output signal
- For each pyramid coefficient (x_0, l_0), an intermediate signal $\tilde{I}$ generated from the local signal value g_0 at x_0
  - $\tilde{I} = min(max(I, g_0 - \sigma_r), g_0 + \sigma_r),\ g_0 = G_{l_0}(x_0)$
- Clipping of high and low values that are σ_r away from local value
- Laplacian pyramid of $\tilde{I}$ computed and value at (x_0, l_0) copied to output pyramid

Local Processing

Range Compression using local processing

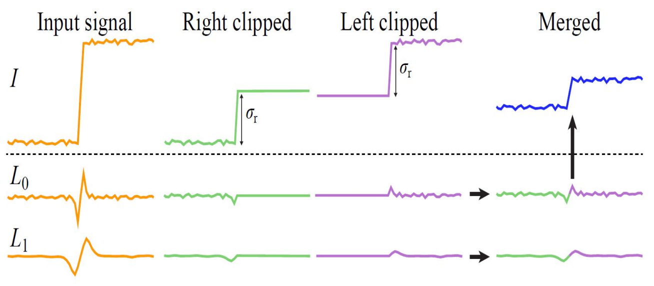


Figure 7: Range compression using thresholding and local processing

Algorithm

Local Laplacian Filtering

- Edge-aware image processing technique based on Laplacian pyramids
- Generalization of the 1-D range compression idea
- Local point-wise method

Algorithm

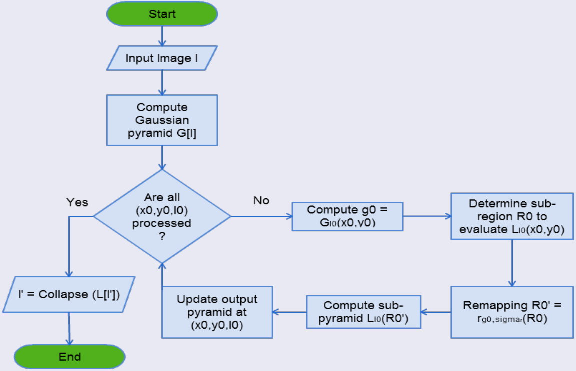


Figure 8: Algorithm Flowchart

Complexity

Algorithm Complexity

- Direct implementation $O(N^2)$
- Process only sub-pyramid needed to evaluate $L_{l_0}[\tilde{I}](x_0, y_0)$
- $K \times K$ sub-region in input image I, where $K = O(2^{l_0})$
- Number of levels $O(logN)$, number of coefficients per level $O(N/2^{l_0})$
- $O(NlogN)$ algorithm

Algorithm

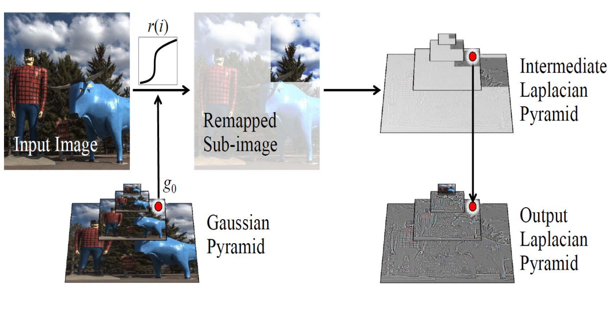


Figure 9: Local Laplacian Filtering Method

Remapping Function

Grayscale Images

- Pixels with values closer than σ_r to local value g_0 processed as details, otherwise processed as edges
- Remapping function $r(i)$
  1. $r(i) = r_d(i) \quad if\ |i - g_0| \leqslant \sigma_r$
  2. $r(i) = r_e(i) \quad otherwise$

Remapping Function

Grayscale Images

- For details,
  $r_d(i) = g_0 + sign(i - g_0)\sigma_r f_d(|i - g_0|/\sigma_r),$
  $f_d : [0, 1] \rightarrow [0, 1]$
- For edges,
  $r_e(i) = g_0 + sign(i - g_0)(f_e(|i - g_0| - \sigma_r) + \sigma_r),$
  $f_e : [0, \infty) \rightarrow \mathbb{R}$

Remapping Function

Remapping Function for Grayscale

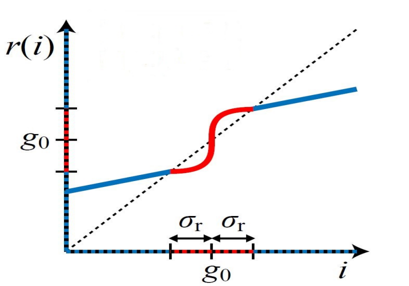


Figure 10: Remapping function

Remapping Function

Color Images

- Remapping functions extended to consider 3D vectors
  1. $r_d(\mathbf{i}) = \mathbf{g}_0 + unit(\mathbf{i} - \mathbf{g}_0)\sigma_r f_d(||\mathbf{i} - \mathbf{g}_0||/\sigma_r)$,
     $f_d : [0,1] \to [0,1]$
  2. $r_e(\mathbf{i}) = \mathbf{g}_0 + unit(\mathbf{i} - \mathbf{g}_0)(f_e(||\mathbf{i} - \mathbf{g}_0|| - \sigma_r) + \sigma_r)$,
     $f_e : [0,\infty) \to \mathbb{R}$

Execution Time

Processing Time

- 5 x 5 kernels exhibits 1 Megapixels per minute on a 2.26GHz processor

Faster Implementation

- Limit the depth of the intermediate pyramid
- Visually indistinguishable with PSNR 40dB
- Data-parallelism and Multi-threading

Vibot

Detail Manipulation

S-shaped Point-wise Detail Manipulation function

$$f_d(\Delta) = \Delta^\alpha$$

Identity Edge Modifying function

$$f_e(a) = a$$

Impact of σ_r

- Defines an Edge region

Detail Manipulation

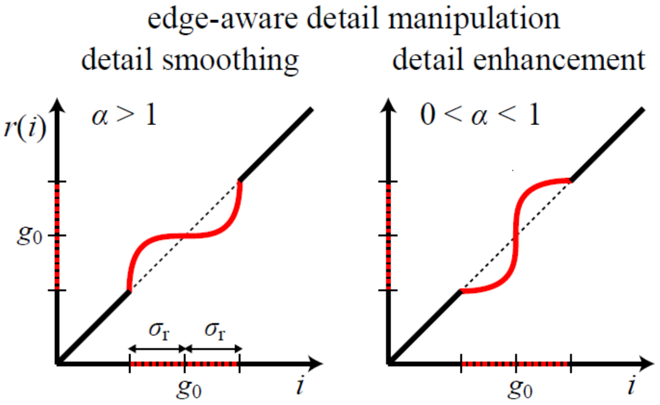


Figure 11: Remapping Function

Effect of α

input | reduced details ($\alpha = 4$) | increased details ($\alpha = 0.25$)

Figure 12: Effect of α ($\sigma_r = 0.3$)

Effect of σ_r

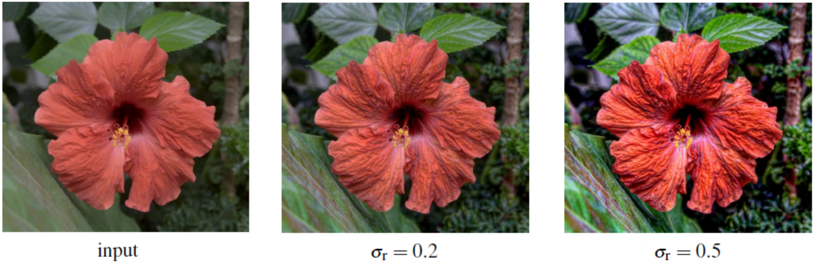


Figure 13: Effect of σ_r ($\alpha = 0.25$)

Detail Manipulation

On selected Subsets of Pyramids

- Applied on selected scale levels
- Interpolation for the intermediate levels

Noise Amplification

- Noise and Artifacts may be introduced
- Limit the smallest Δ amplified:

$$f_d(\Delta) = \tau\Delta^\alpha + (1\tau)\Delta$$

where:

$$\tau = \begin{cases} 0 & \text{if } \Delta < 1\% \text{ of maximum Intensity} \\ 1 & \text{if } \Delta > 2\% \text{ of maximum Intensity} \\ (0,1) & \text{otherwise} \end{cases}$$

Tone Manipulation

Edge Modifying function

$$f_e(a) = \beta a$$

Intensity Image

$$I_i = \frac{1}{61}(20I_r + 40I_g + I_b)$$

Colour Ratios

$$(\rho_r, \rho_g, \rho_b) = \frac{1}{I_i}(I_r, I_g, I_b)$$

Parameters

$$\alpha \leq 1, \qquad \beta < 1$$

Tone Manipulation

Algorithm

- Filter is applied on log of intensities $(log(I_i))$
- Normalize the result
- Scale according to a user-defined parameter (100:1)
- Multiply by color ratios
- Gamma correction $(\exp^{\frac{1}{2.2}})$

σ_r and β control the balance between the local and global contrast

Tone Manipulation

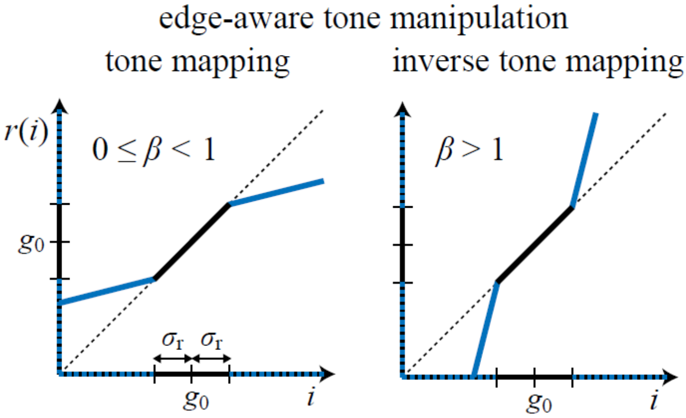
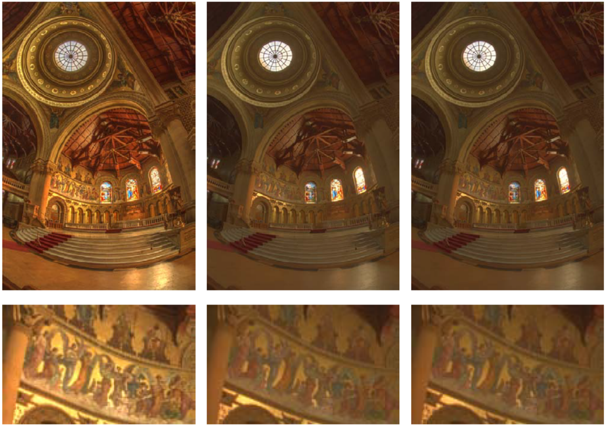


Figure 14: Remapping Function

Effect of σ_r and β



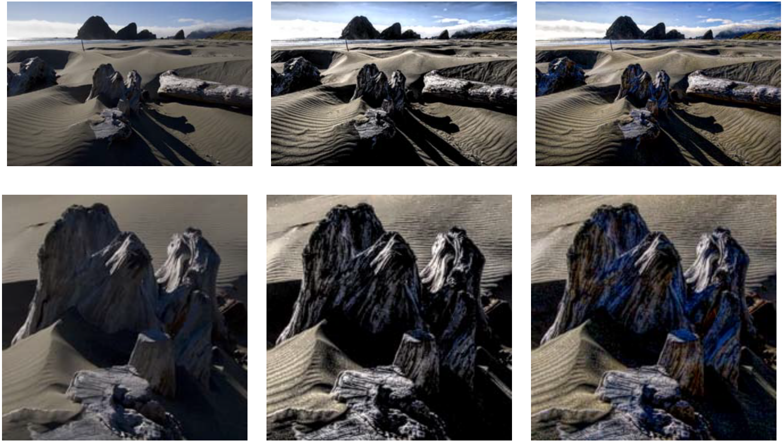
(a) $\beta = 0$
$\sigma_r = \log(2.5)$

(b) $\beta = 0$
$\sigma_r = \log(30)$

(c) $\beta = 0.75$
$\sigma_r = \log(2.5)$

Figure 15: Effect of σ_r and β

Luminance vs RGB Filtering



(d) Input (e) Luminance (f) RGB

Figure 16: Luminance vs RGB images ($\alpha = 0.25$, $\beta = 1$, $\sigma_r = 0.4$)

Combined Manipulation

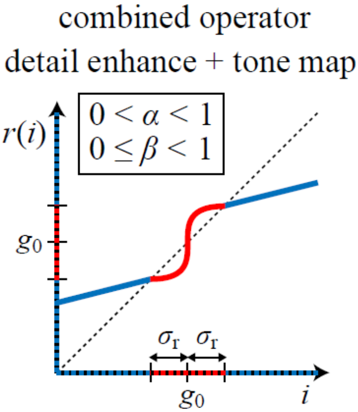


Figure 17: Remapping Function

Ignorance of Scene Semantics

(a) no detail increase ($\alpha = 1$) (b) detail increased ($\alpha = 0.25$)

Figure 18: Artifacts introduced due to poor selection of parameters

Comparison with Bilateral Filter

(a) Output Image (b) Bilateral Image (c) Laplacian Image

Figure 19: Comparison between uncorrelated bilateral filter and local laplacian filter

Conclusion

Concluding Remarks

- Edge-aware technique based on Laplacian Pyramids
- Extension to a wide-range of filters
- Good Resistance to Artifacts
- Consistent with a fair range of parameter settings
- Computationally inefficient

References

- [1] S. Paris, S. Hasinoff, J. Kautz. 2011. Local Laplacian Filters: Edge-aware Image Processing with a Laplacian Pyramid. ACM Trans. Graph. 30 (4), 68.
- [2] P. J. Burt, E. H. Adelson. 1983. The Laplacian pyramid as a compact image code. IEEE Transactions on Communication 31, 4, 532-540.
- [3] BAE, S., PARIS, S., AND DURAND, F. 2006. Two-scale tone management for photographic look. ACM Transactions on Graphics (Proc. SIGGRAPH) 25, 3, 637-645.
- [4] CHEN, J., PARIS, S., AND DURAND, F. 2007. Real-time edgeaware image processing with the bilateral grid. ACM Transactions on Graphics (Proc. SIGGRAPH) 26, 3.
- [5] SUBR, K., SOLER, C., AND DURAND, F. 2009. Edge-preserving multiscale image decomposition based on local extrema. ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 28, 5.

THANK YOU