Advanced Graphics & Image Processing

Advanced image processing

Part 1/2 – edge stopping filters

Rafał Mantiuk

Computer Laboratory, University of Cambridge
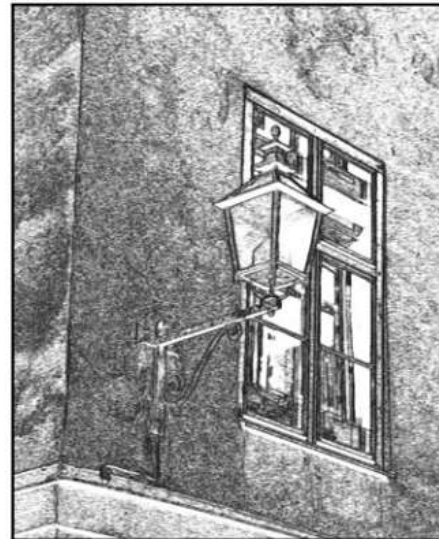
Edge stopping filters

Original

Edge-aware smoothing

Detail enhancement

Stylization

Recoloring
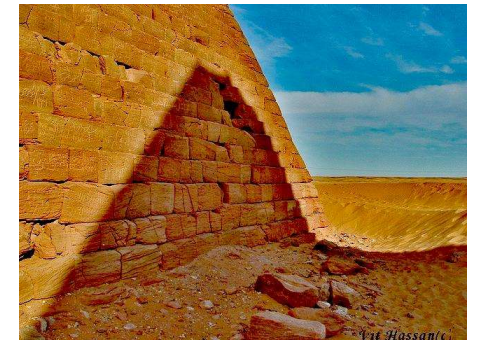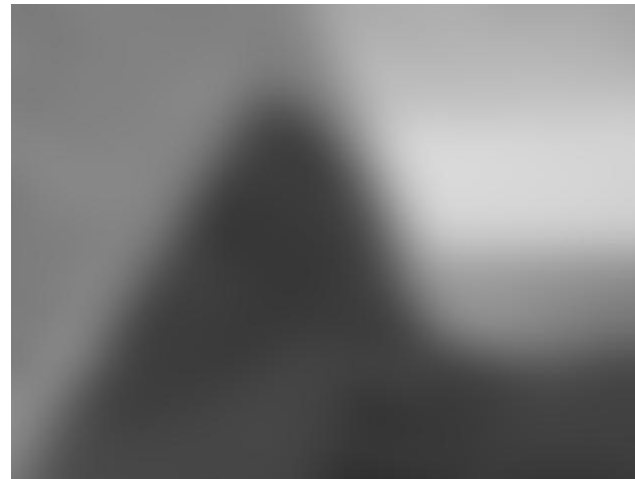
Pencil drawing

Depth-of-field

Examples from [Gastal & Oliveira 2011]

Nonlinear filters: Bilateral filter

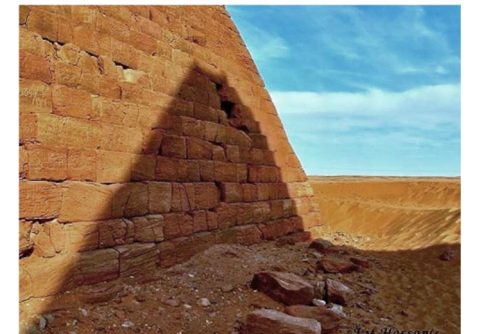- Goal: Smooth out an image without blurring edges
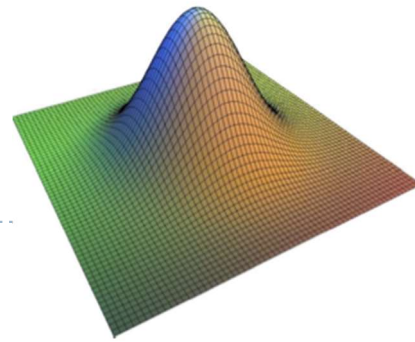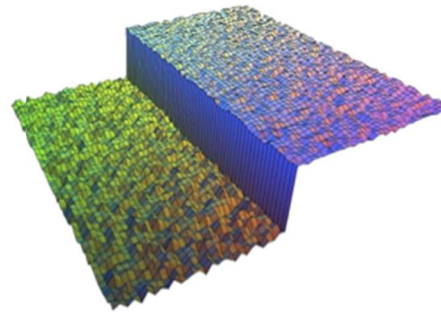
Gaussian filter
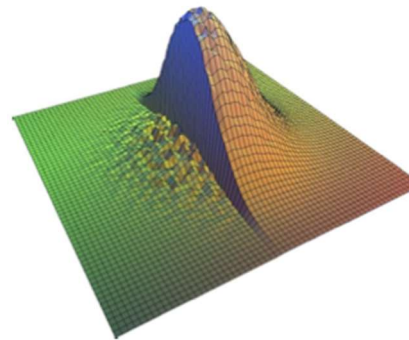
Unsharp masking

Bilateral filter

Bilateral filter

spatial kernel f

influence g in the intensity domain for the central pixel

weight $f \times g$ for the central pixel

output

input

"Kernel" changes from one pixel to another

Kernel for this pixel
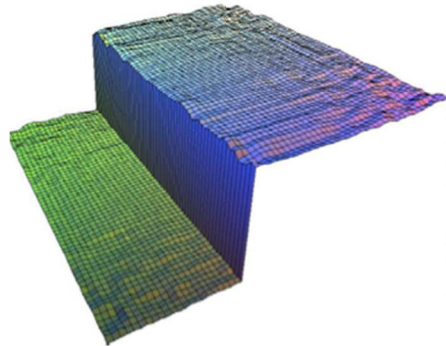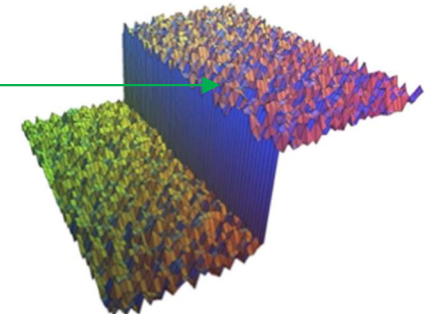
Bilateral filter

Input image

$$y(\boldsymbol{p}) = \frac{\sum_{\boldsymbol{q}\in\Omega} x(\boldsymbol{q})w(\boldsymbol{p},\boldsymbol{q})}{\sum_{\boldsymbol{q}\in\Omega} w(\boldsymbol{p},\boldsymbol{q})}$$

Pixel coordinates $\boldsymbol{p} = (i, j)$

Neighborhood of the pixel **p**

$$w(\boldsymbol{p},\boldsymbol{q}) = g_s(\boldsymbol{p}-\boldsymbol{q})g_r(x(\boldsymbol{p})-x(\boldsymbol{q}))$$

distance in the spatial position (x,y)

distance (difference) in pixel values

$$g_s(\boldsymbol{d}) = \exp\left(\frac{-\|\boldsymbol{d}\|_2}{2\sigma_s^2}\right) \qquad g_r(d) = \exp\left(\frac{-d^2}{2\sigma_r^2}\right)$$
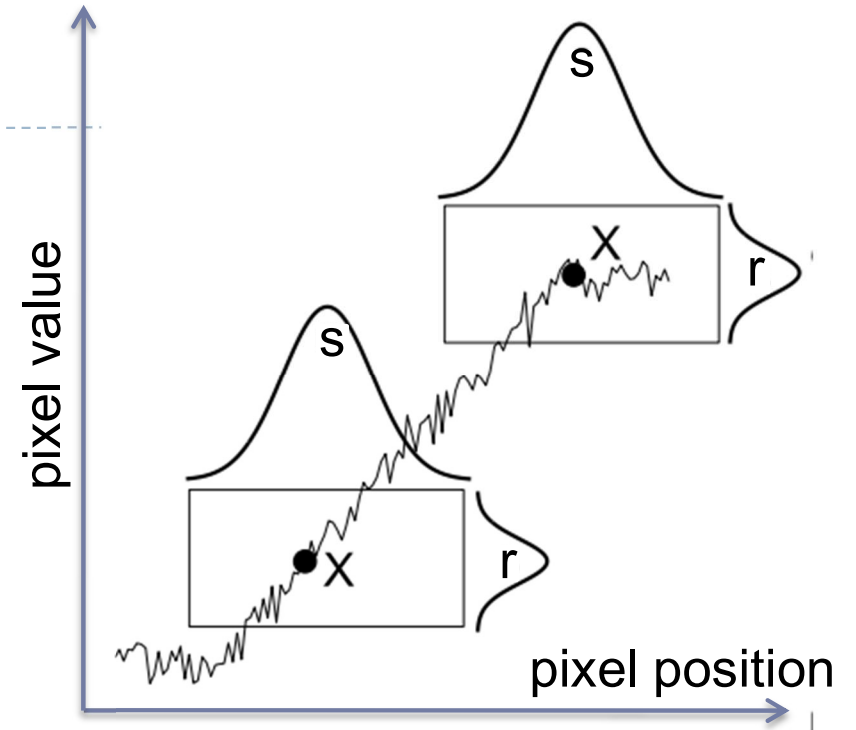
How to make the bilateral filter fast?

- A number of approximations have been proposed
  - Combination of linear filters [Durand & Dorsey 2002, Yang et al. 2009]
  - Bilateral grid [Chen et al. 2007]
  - Permutohedral lattice [Adams et al. 2010]
  - Domain transform [Gastal & Oliveira 2011]

Joint-bilateral filter (a.k.a guided/cross b.f.)

▸ The "range" term does not need to operate in the same domain as the filter output

  ▸ Example:

Stereo image pair

A simplified algorithm from [Mueller et al. 2010]

The "spatial" term operates on disparities

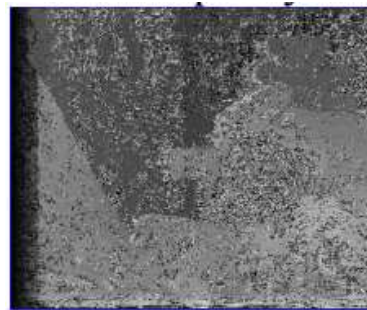The "range" term operates on the colour image
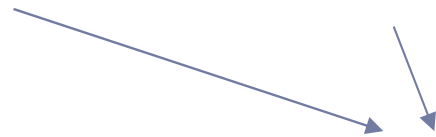
Estimated left-to-right disparity

Joint bilateral filter

Filtered disparity

Joint bilateral filter: Flash / no-flash

Flash   No-flash
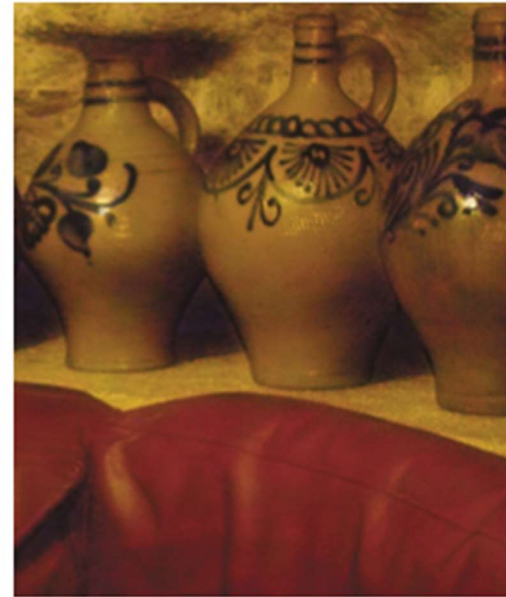
Detail transfer with denoising

- ▸ Preserve colour and illumination from the no-flash image
- ▸ Use flash image to remove noise and add details
- ▸ [Petshnigg et al. 2004]

A No-Flash Image		F Flash Image	
			F^{Lin} A^{Lin}
Bilateral Filter	Joint Bilateral Filter	Bilateral Filter	Shadow & Specularity Detection
		F^{Base}	
		$\div$	
A^{Base}	A^{NR}	F^{Detail}	Mask M
Denoising		Detail Transfer	Artifact detection

$$A^{Final} = (1 - M)A^{NR}F^{Detail} + MA^{Base}$$
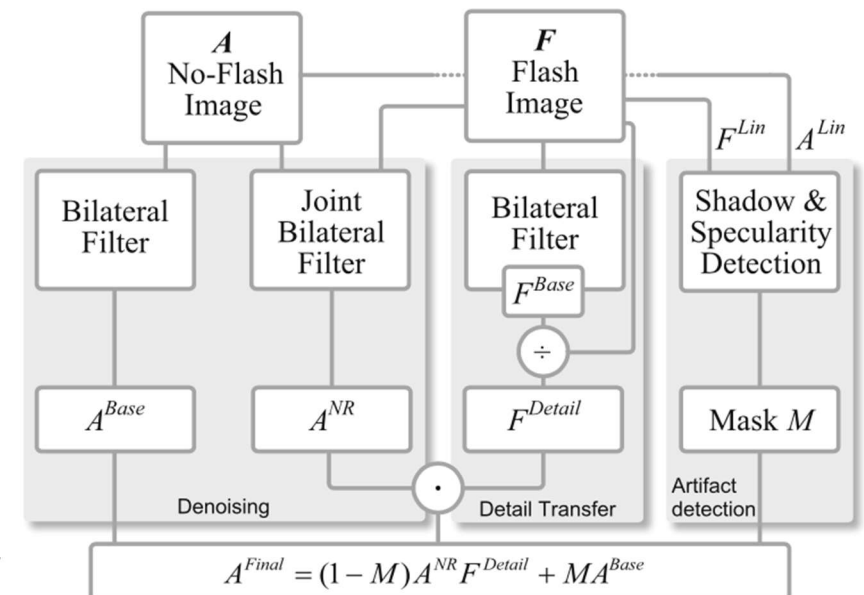
Example of edge preserving filtering

▶ Domain Transform for Edge-Aware Image and Video Processing

▶ Video:

  ▶ https://youtu.be/Ul1xh11QrTY?t=4m10s

  ▶ From: http://inf.ufrgs.br/~eslgastal/DomainTransform/

Advanced Graphics & Image Processing

Advanced image processing

Part 1/2 – processing by optimization

Rafał Mantiuk

Computer Laboratory, University of Cambridge

Optimization-based methods
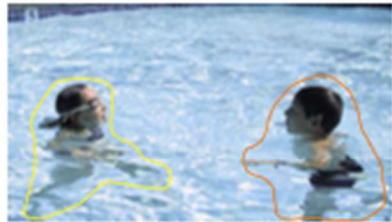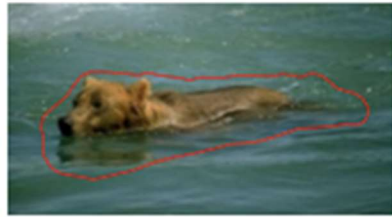
sources/destinations

cloning

seamless cloning

Poisson image editing [Perez et al. 2003]

Gradient Domain compositing

▸ Compositing [Wang et al. 2004]

images from [Drori at al. 2004]
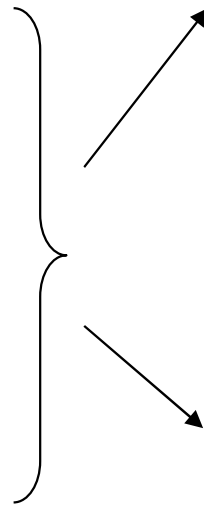
Gradient domain methods

▸ Operate on pixel gradients instead of pixel values
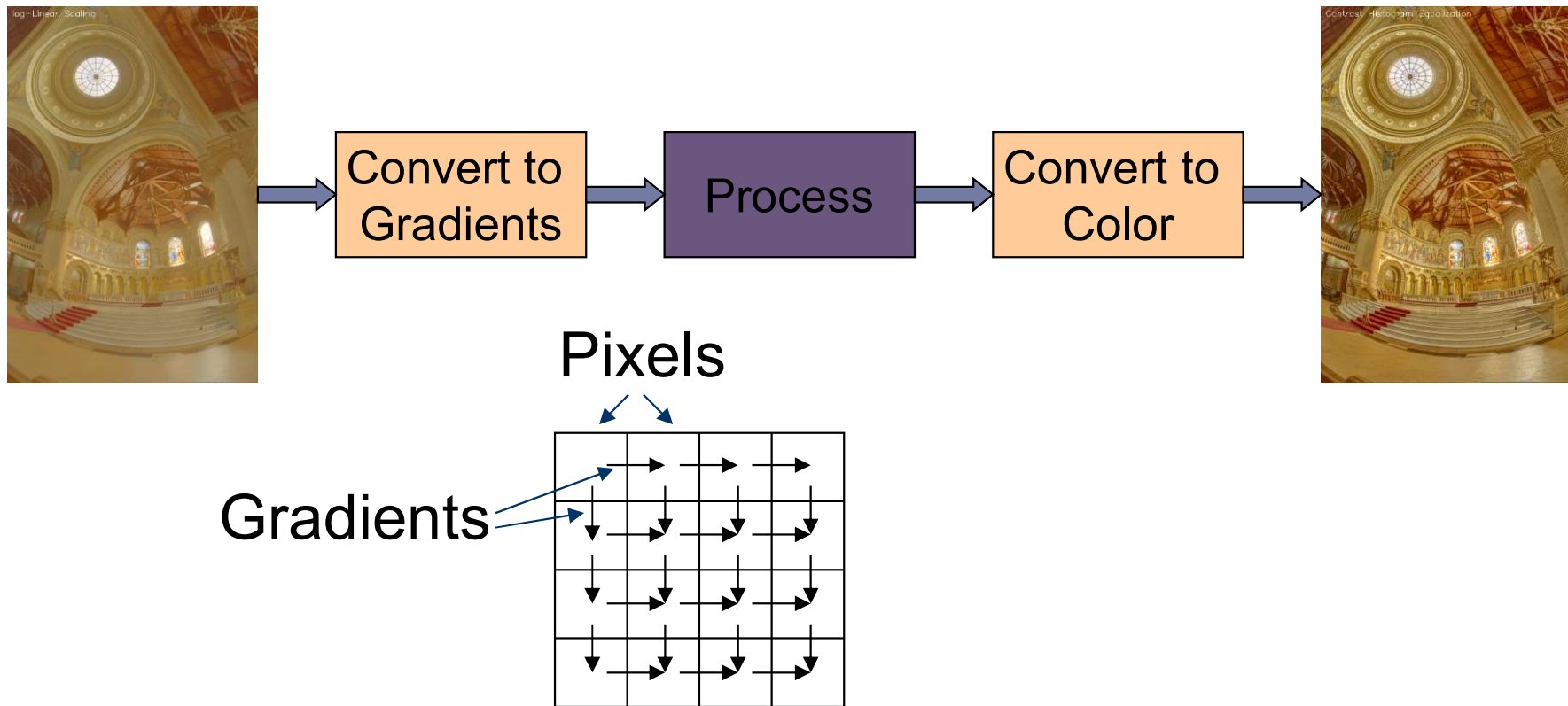
Forward Transformation

▸ **Forward Transformation**

  ▸ Compute gradients as differences between a pixel and its two neighboors

  $$\nabla I_{x,y} = \begin{bmatrix} I_{x+1,y} - I_{x,y} \\ I_{x,y+1} - I_{x,y} \end{bmatrix}$$
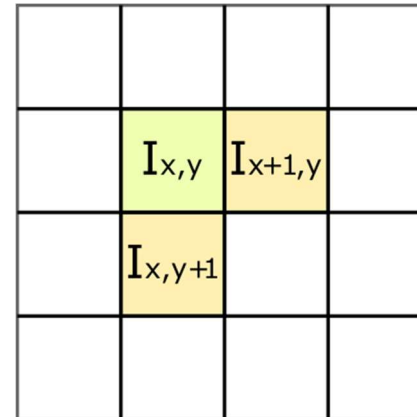
  ▸ Result: 2D gradient map (2 x more values than the number of pixels)

Processing gradient field

▸ Typically, gradient magnitudes are modified while gradient direction (angle) remains the same

$$G_{x,y} = \nabla I_{x,y} \cdot \frac{f\left(||\nabla I_{x,y}||\right)}{||\nabla I_{x,y}||}$$

Gradient editing function

▸ Examples of gradient editing functions:

Hard thresholding — Edited magnitude vs. Original magnitude

Soft thresholding — Edited magnitude vs. Original magnitude

Smooth attenuation — Edited magnitude vs. Original magnitude
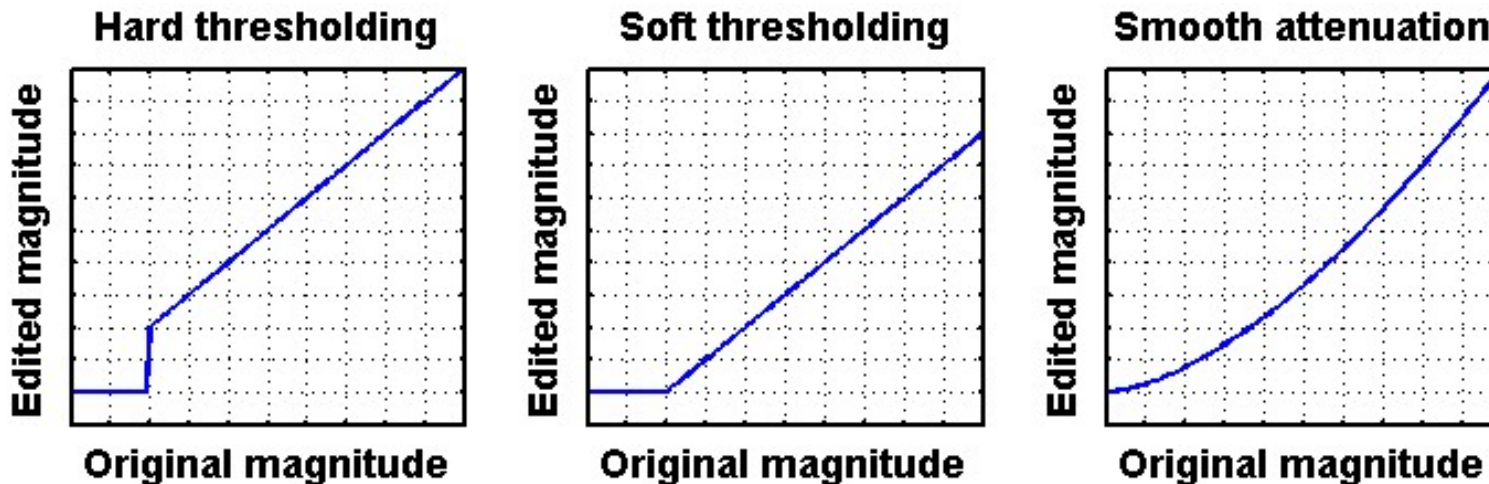
Inverse transform: the difficult part

▸ There is no strightforward transformation from gradients to luminance
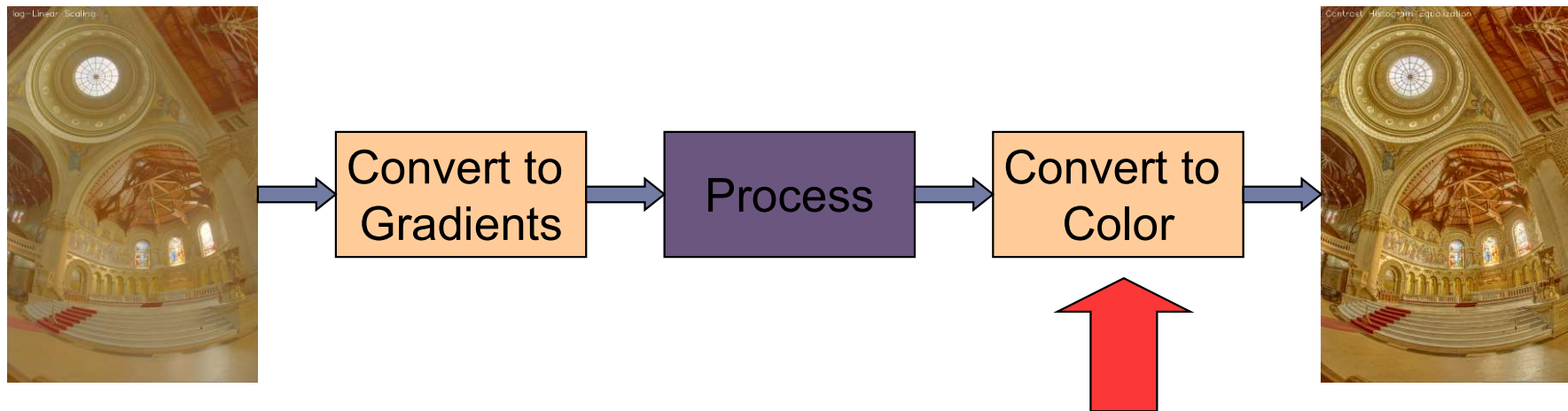
▪ Instead, a minimization problem is solved:

$$\arg \min_{I} \sum_{x,y} \left[\left(I_{x+1,y} - I_{x,y} - G^{(x)}_{x,y} \right)^2 + \left(I_{x,y+1} - I_{x,y} - G^{(y)}_{x,y} \right)^2 \right]$$

Image Pixels

Desired gradients

Inverse transformation

▸ **Convert modified gradients to pixel values**

  ▸ Not trivial!

  ▸ Most gradient fields are inconsistent - do not produce valid images

  ▸ If no accurate solution is available, take the best possible solution

  ▸ Analogy: system of springs
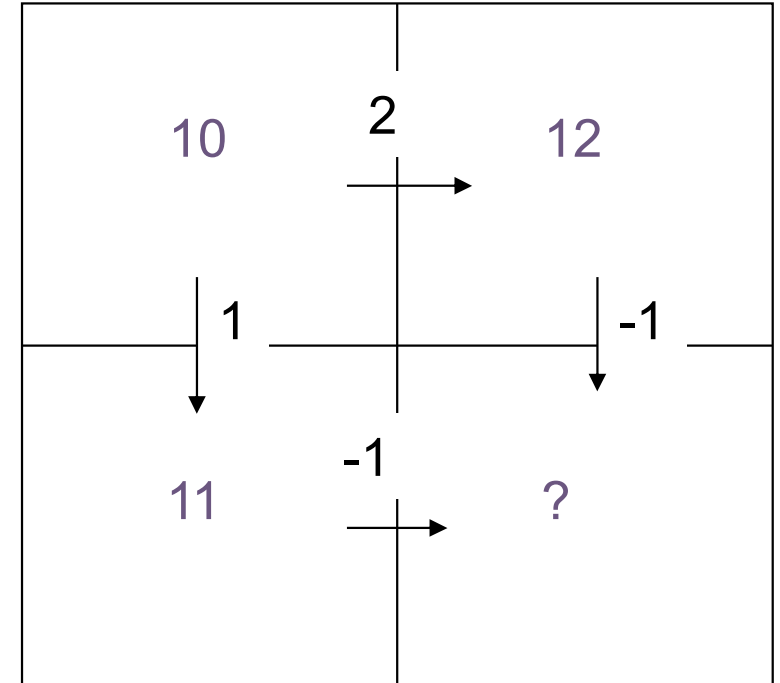
Gradient field reconstruction: derivation

▸ **The minimization problem is given by:**

$$\arg\min_{I} \sum_{x,y} \left[\left(I_{x+1,y} - I_{x,y} - G_{x,y}^{(x)} \right)^2 + \left(I_{x,y+1} - I_{x,y} - G_{x,y}^{(y)} \right)^2 \right]$$

▸ **After equating derivatives over pixel values to 0 we get:**

  ▸ Derivation done in the lecture

$$I_{x-1,y} + I_{x+1,y} + I_{x,y-1} + I_{x,y+1} - 4I_{x,y} = G_{x,y}^{(x)} - G_{x-1,y}^{(x)} + G_{x,y}^{(y)} - G_{x,y-1}^{(y)}$$

  ▸ In matrix notation:

Laplace operator (NxN matrix)

$$\nabla^2 I = \mathrm{div}G$$

Divergence of a vector field (Nx1 vector)

$$\nabla^2 I_{x,y} = I_{x-1,y} + I_{x+1,y} + I_{x,y-1} + I_{x,y+1} - 4I_{x,y}$$

Image as a column vector

$$\begin{bmatrix} I_{1,1} \\ I_{2,1} \\ \dots \\ I_{N,M} \end{bmatrix}$$

$$\mathrm{div}G_{x,y} = G_{x,y}^{(x)} - G_{x-1,y}^{(x)} + G_{x,y}^{(y)} - G_{x,y-1}^{(y)}$$
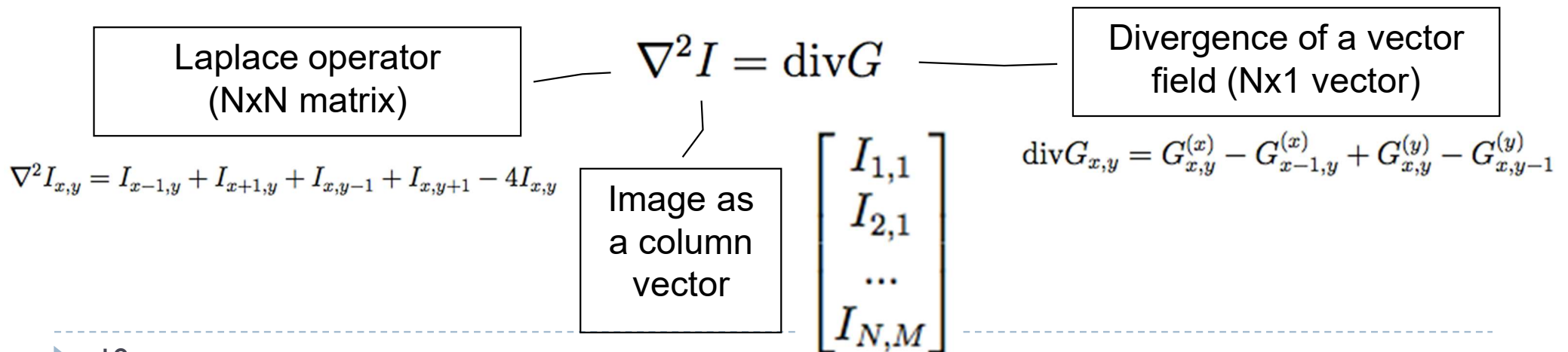
Laplace operator for 3x3 image

$$\nabla^2 = \begin{bmatrix} -2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -3 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -2 \end{bmatrix}$$

Solving sparse linear systems

▸ Just use "\" operator in Matlab / Octave:

  ▸ x = A \ b;

▸ Great "cookbook":

  ▸ TEUKOLSKY, S.A., FLANNERY, B.P., PRESS, W.H., AND VETTERLING, W.T. 1992. *Numerical recipes in C.* Cambridge University Press, Cambridge.

▸ Some general methods

  ▸ Cosine-transform – fast but cannot work with weights (next slides) and may suffer from floating point precision errors

  ▸ Multi-grid – fast, difficult to implement, not very flexible

  ▸ Conjugate gradient / bi-conjugate gradient – general, memory efficient, iterative but fast converging

  ▸ Cholesky decomposition – effective when working on sparse matrices

Pinching artefacts

- A common problem of gradient-based methods is that they may result in "pinching" artefacts (left image)

- Such artefacts can be avoided by introducing weights to the optimization problem

Weighted gradients

▸ The new objective function is:

$$\arg\min_{I} \sum_{x,y} \left[w_{x,y}^{(x)} \left(I_{x+1,y} - I_{x,y} - G_{x,y}^{(x)} \right)^2 + w_{x,y}^{(y)} \left(I_{x,y+1} - I_{x,y} - G_{x,y}^{(y)} \right)^2 \right]$$

▸ so that higher weights are assigned to low gradient magnitudes (in the original image).

$$w_{x,y}^{(x)} = w_{x,y}^{(y)} = \frac{1}{||\nabla I_{x,y}^{(o)}|| + \epsilon}$$

▸ The linear system can be derived again

  ▸ but this is a lot of work and is error-prone

Weighted gradients - matrix notation (1)

- The objective function:

$$\arg\min_{I} \sum_{x,y} \left[w_{x,y}^{(x)} \left(I_{x+1,y} - I_{x,y} - G_{x,y}^{(x)} \right)^2 + w_{x,y}^{(y)} \left(I_{x,y+1} - I_{x,y} - G_{x,y}^{(y)} \right)^2 \right]$$

- In the matrix notation (without weights for now):

$$\arg\min_{I} \left\| \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} I - \begin{bmatrix} G^{(x)} \\ G^{(y)} \end{bmatrix} \right\|^2$$

- Gradient operators (for 3x3 pixel image):

$$\nabla_x = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\nabla_y = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Weighted gradients - matrix notation (2)

▸ The objective function again:

$$\arg\min_I \left\| \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} I - \begin{bmatrix} G^{(x)} \\ G^{(y)} \end{bmatrix} \right\|^2$$

▸ Such over-determined least-square problem can be solved using pseudo-inverse:

$$\begin{bmatrix} \nabla'_x & \nabla'_y \end{bmatrix} \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} I = \begin{bmatrix} \nabla'_x & \nabla'_y \end{bmatrix} \begin{bmatrix} G^{(x)} \\ G^{(y)} \end{bmatrix}$$

▸ Or simply:

$$\left(\nabla'_x \nabla_x + \nabla'_y \nabla_y \right) I = \nabla'_x G^{(x)} + \nabla'_y G^{(y)}$$

▸ With weights:

$$\left(\nabla'_x W \nabla_x + \nabla'_y W \nabla_y \right) I = \nabla'_x W G^{(x)} + \nabla'_y W G^{(y)}$$
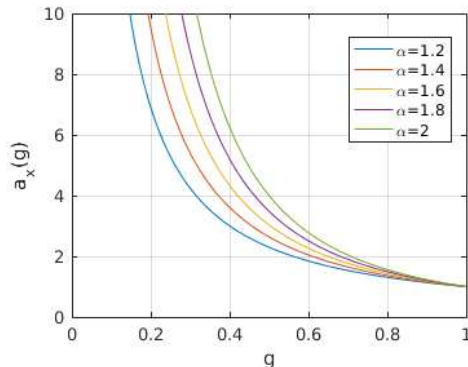
WLS filter: Edge stopping filter by optimization

▸ ## Weighted-least-squares optimization

> Make reconstructed image u possibly close to input g

> Smooth out the image by making partial derivatives close to 0

$$\underset{u}{\operatorname{argmin}} \sum_p \left((u_p - g_p)^2 + \lambda \left(a_{x,p}(g) \left(\frac{\partial u}{\partial x} \right)_p^2 + a_{y,p}(g) \left(\frac{\partial u}{\partial y} \right)_p^2 \right) \right)$$
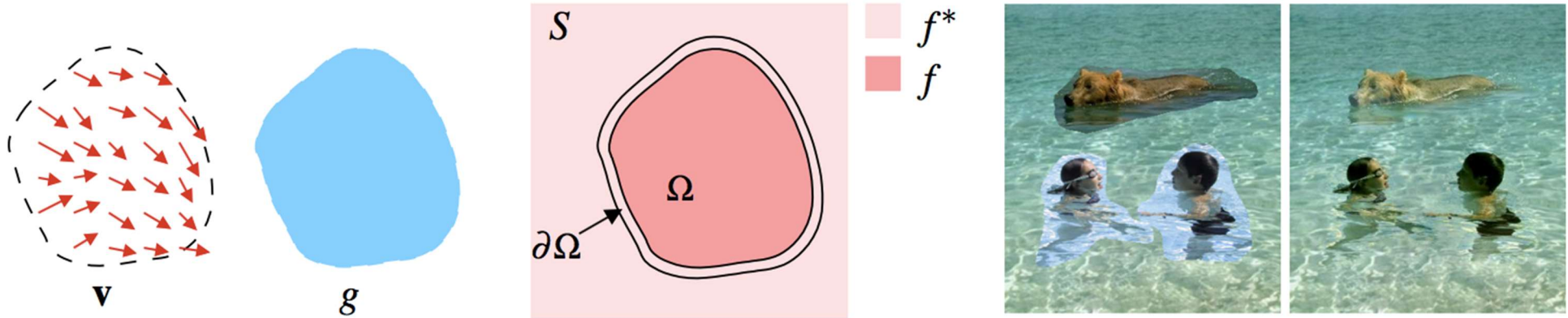
> Spatially varying smoothing – less smoothing near the edges

$$a_{x,p}(g) = \frac{1}{\left| \frac{\partial u}{\partial x}(g) \right|^\alpha + \epsilon}$$

▸ [Farbman, Z., Fattal, R., Lischinski, D., & Szeliski, R. (2008). Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM SIGGRAPH 2008*, 1–10.]

Poisson image editing

$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2$$

subject to:

$$f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

▸ Reconstruct unknown values *f* given a source guidance gradient field *v* and the boundary conditions $f|_{\partial\Omega} = f^*|_{\partial\Omega}$

▸ [Pérez, P., Michel Gangnet, & Blake, A. (2003). Poisson Image Editing. *ACM Transactions on Graphics*, 3(22), 313–318. https://doi.org/10.1145/882262.882269]

Colour 2 Gray

▶ **Transform colour images to gray scale**

▶ **Preserve colour saliency**

  ▶ When gradient in luminance close to 0

  ▶ Replace it with gradient in chrominance

  ▶ Reconstruct an image from gradients

▶ Gooch, A. A., Olsen, S. C., Tumblin, J., & Gooch, B. (2005). Color2Gray. *ACM Transactions on Graphics*, 24(3), 634. https://doi.org/10.1145/1073204.1073241
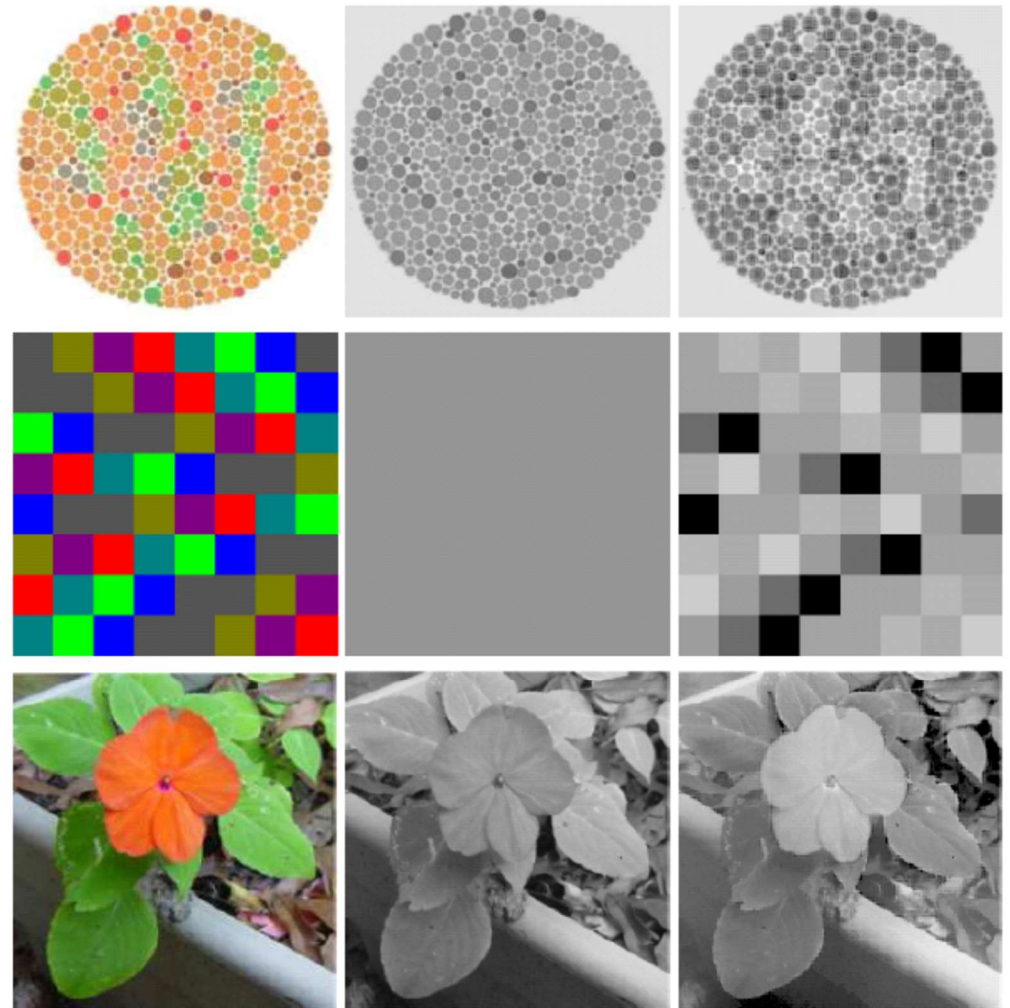
Gradient Domain: applications

- More applications:
  - Lightness perception (Retinex) [Horn 1974]
  - Matting [Sun et al. 2004]
  - Color to gray mapping [Gooch et al. 2005]
  - Video Editing [Perez at al. 2003, Agarwala et al. 2004]
  - Photoshop's Healing Brush [Georgiev 2005]

References

▸ F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 257–266, Jul. 2002.

▸ E. S. L. Gastal and M. M. Oliveira, "Domain transform for edge-aware image and video processing," *ACM Trans. Graph.*, vol. 30, no. 4, p. 1, Jul. 2011.

▸ Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. *ACM Trans. Graph.* 22, 3 (July 2003), 313-318. DOI: http://dx.doi.org/10.1145/882262.882269

▸ Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski. 2008. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans. Graph.* 27, 3, Article 67 (August 2008), 10 pages. DOI: https://doi.org/10.1145/1360612.1360666