اصول پردازش تصویر

Principles of Image Processing

مصطفی کمالی تبریزی

۱ و ۶ دی ۱۳۹۹

جلسه بیست هفتم و بیست هشتم

Homework 4

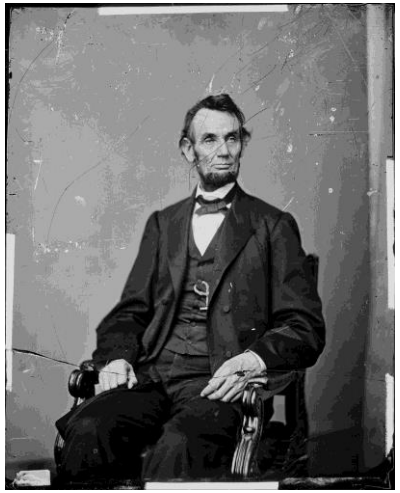
Q1: Texture Synthesis

Q2: Image Morphing

Blending and Compositing

Photo manipulation is as old as photography

classic Lincoln



John Calhoun

Slide credit: Alyosha Efros

…a political weapon

reprinted by New York Times (2004)

Photo: Owen Franken
1972 in Florida

Photo: Ken Light 1971
in New York

Slide: Alyosha Efros

Image Blending

How do I put an object from one image into another?

Slide: Derek Hoiem

Three methods

1. Cut and paste

2. Laplacian pyramid blending

3. Poisson blending

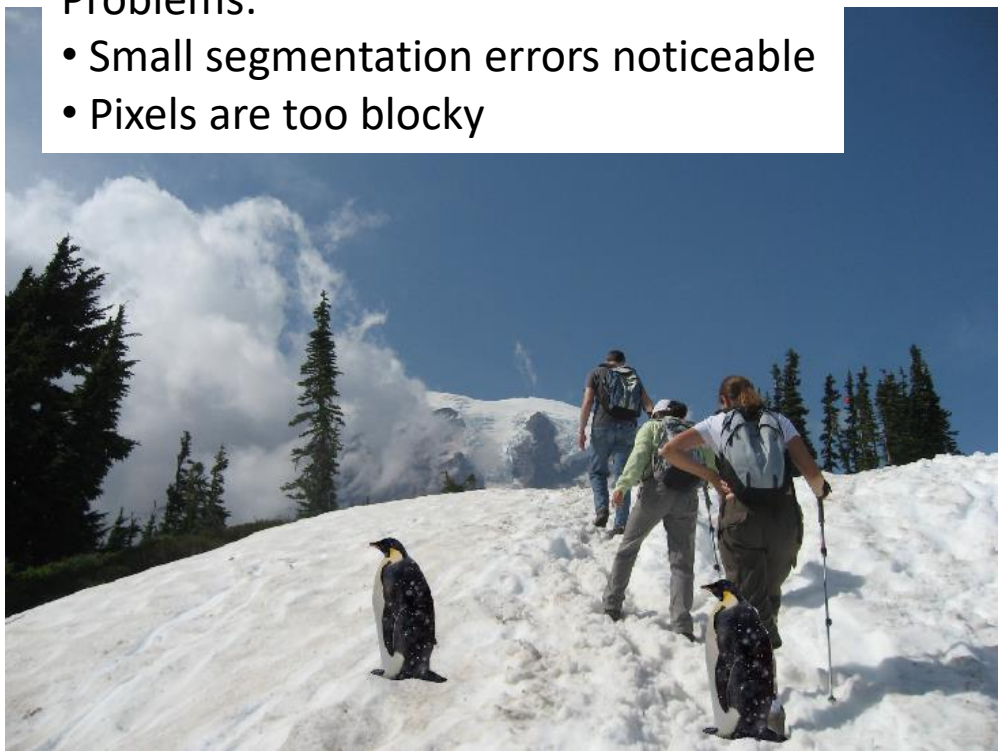
Slide: Derek Hoiem

Method 1: Cut and Paste

Slide: Derek Hoiem

Method 1: Cut and Paste

Problems:
- Small segmentation errors noticeable
- Pixels are too blocky



Slide: Derek Hoiem

Method 1: Cut and Paste

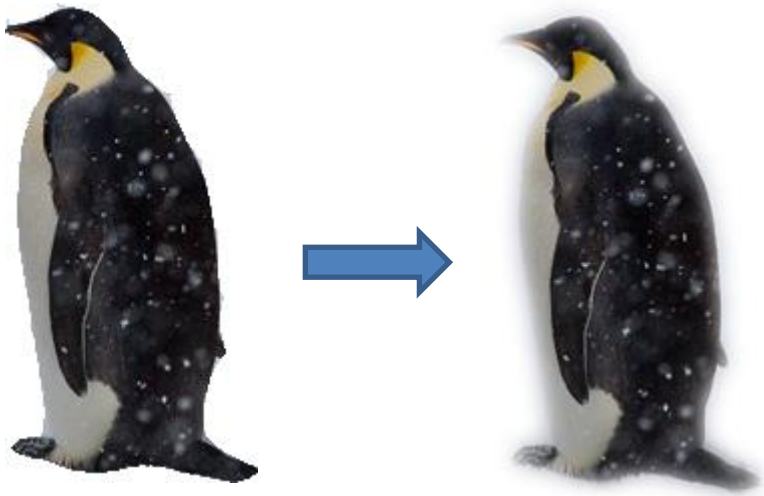
Problems:

- Small segmentation errors noticeable
- Pixels are too blocky
- Won’t work for semi-transparent materials

Slide: Derek Hoiem

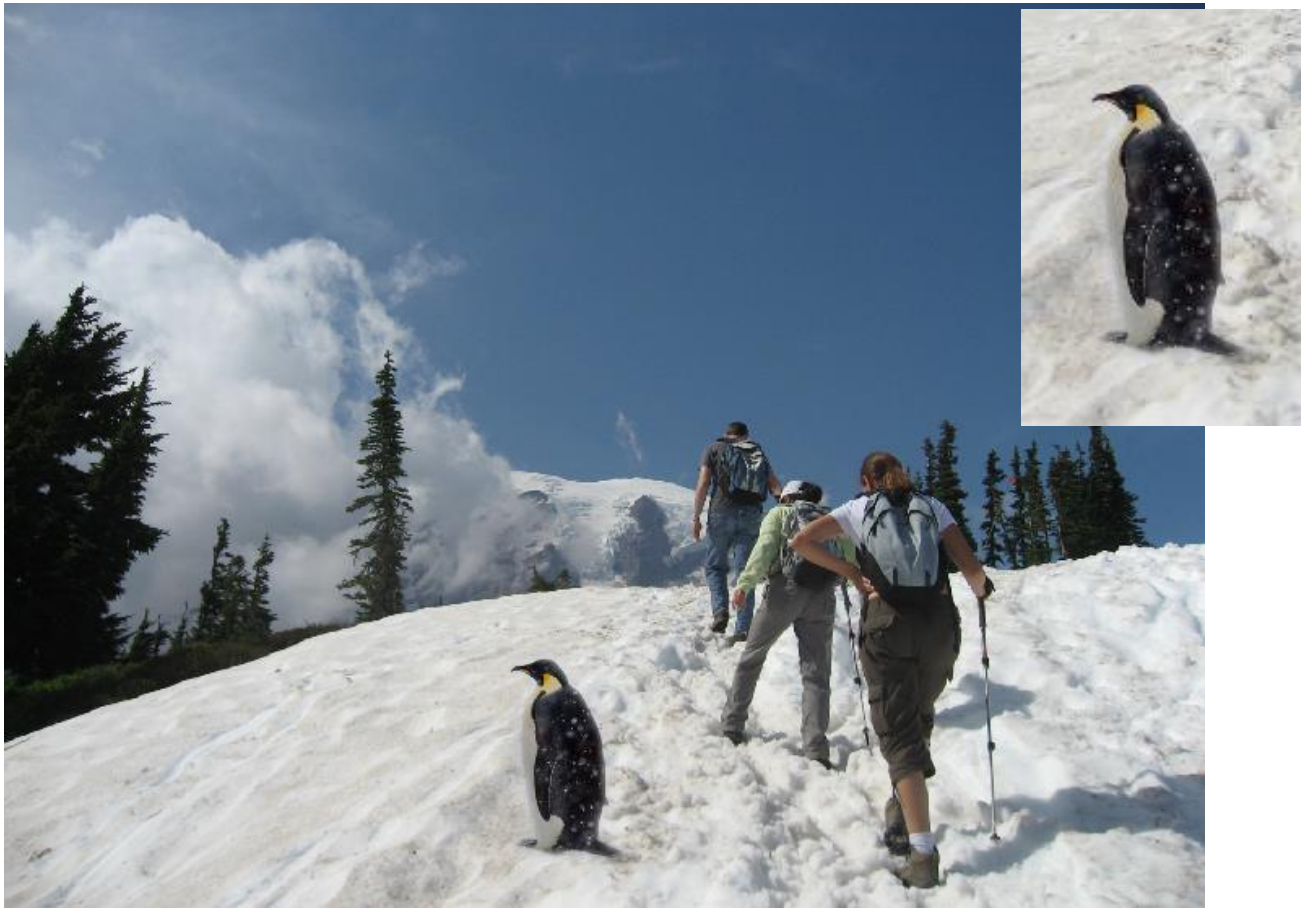
Feathering

Near object boundary pixel values come partly from foreground and partly from background



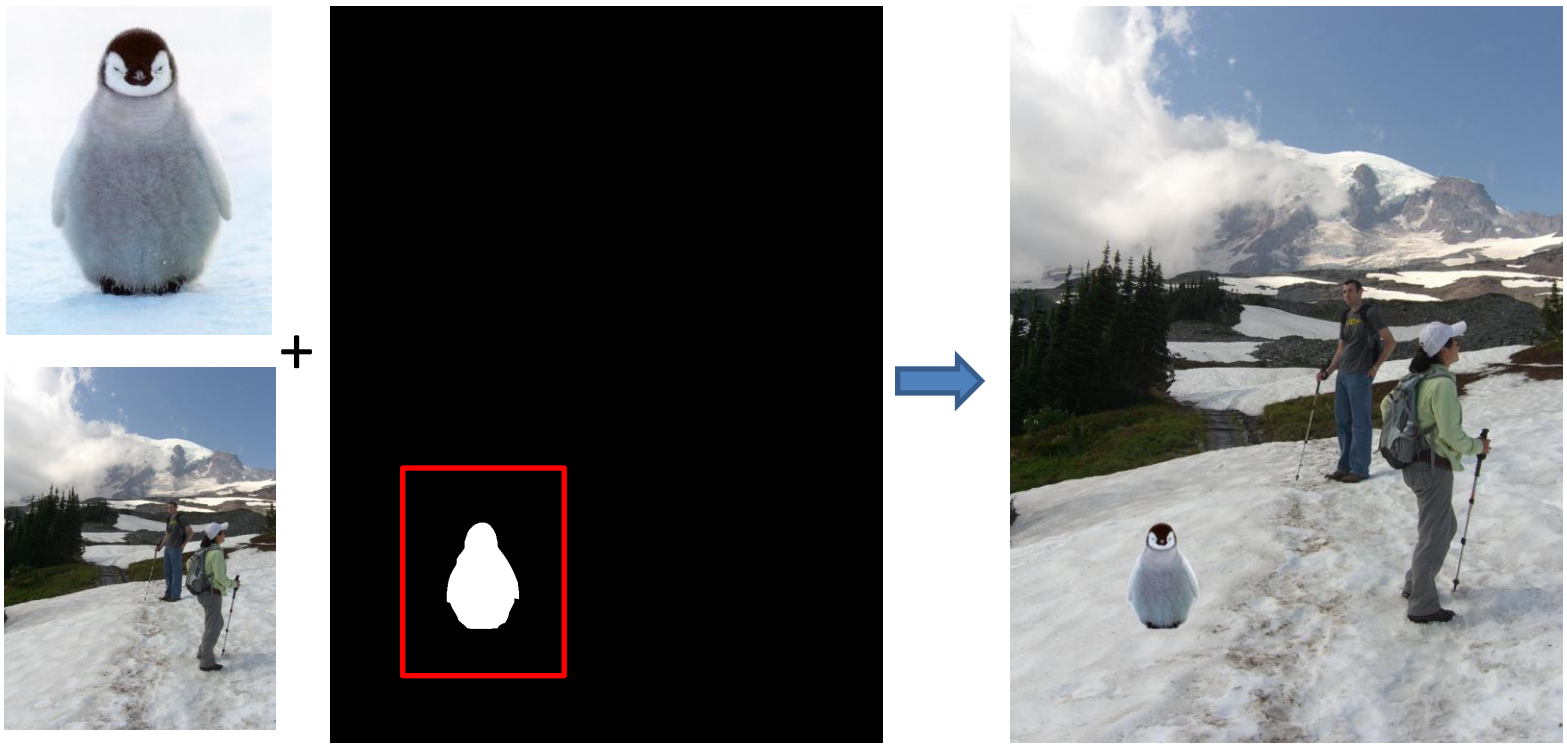
Slide: Derek Hoiem

Method 1: Cut and Paste (with feathering)



Slide: Derek Hoiem

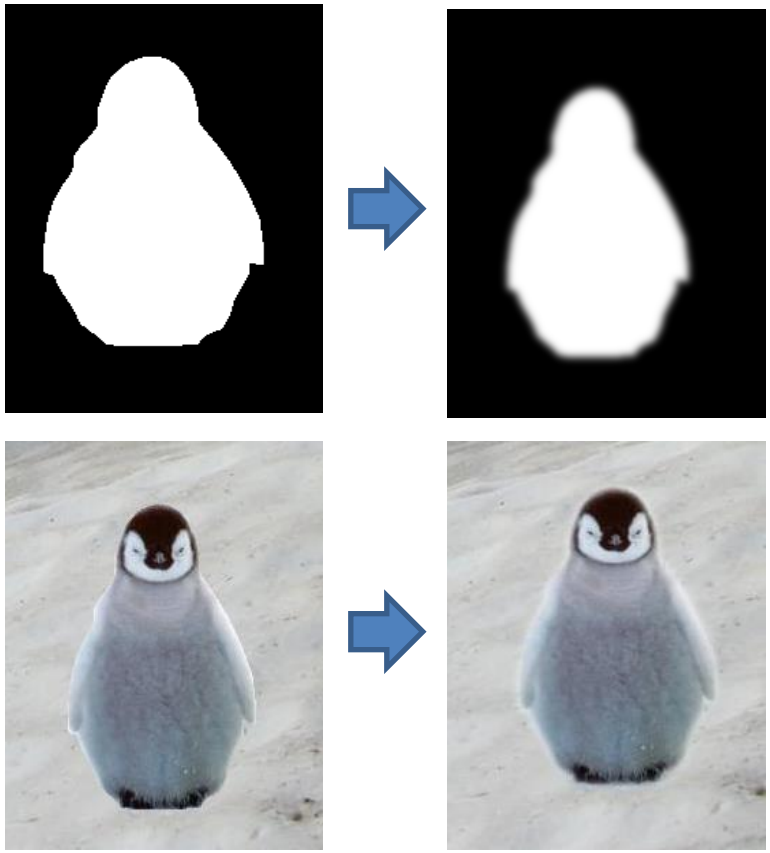
Alpha compositing



$$\text{Output} = (\text{Fourground} * \text{Mask}) + \big(\text{Background} * (1 - \text{Mask})\big)$$

Slide: Derek Hoiem

Alpha compositing with feathering

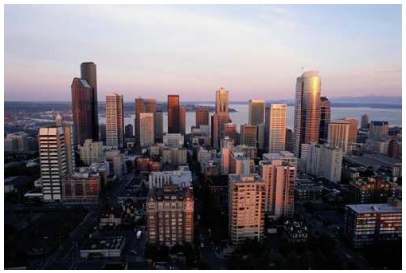


$$\text{Output} = (\text{Fourground} * \text{Mask}) + \big(\text{Background} * (1 - \text{Mask})\big)$$

Slide: Derek Hoiem

Another example (without feathering)

Mattes

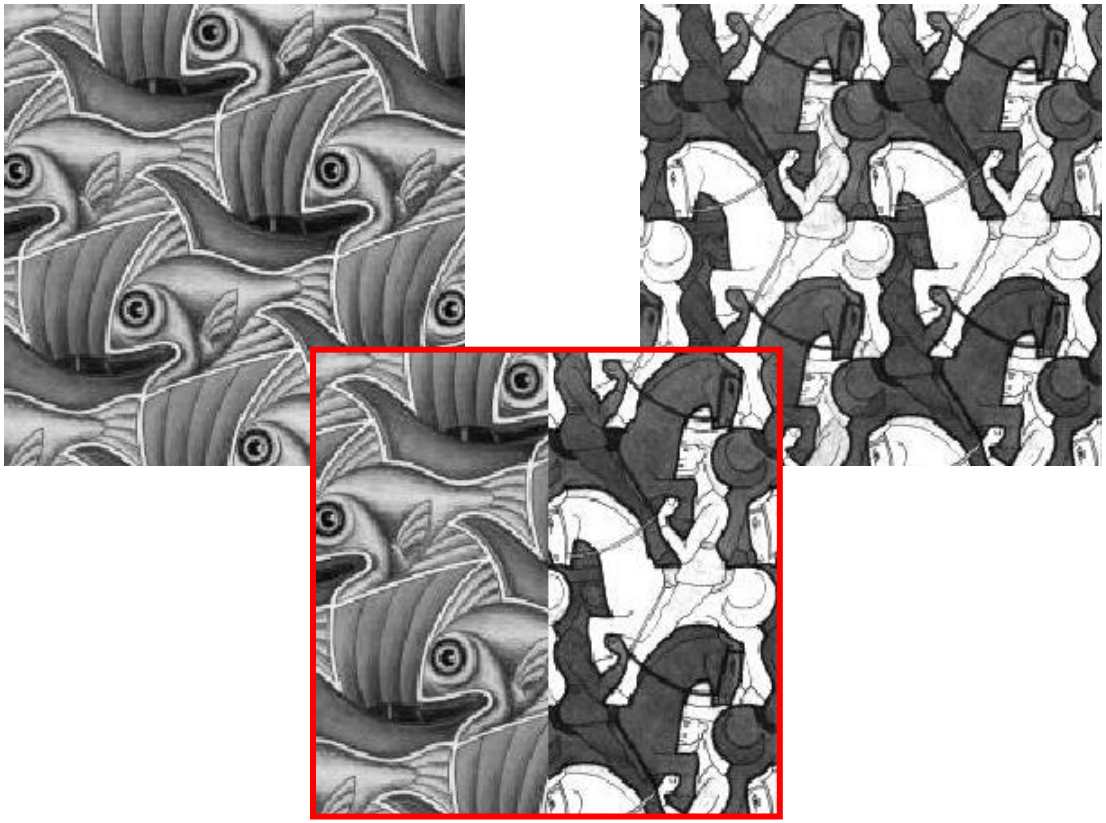


Composite

Composite by
David Dewey

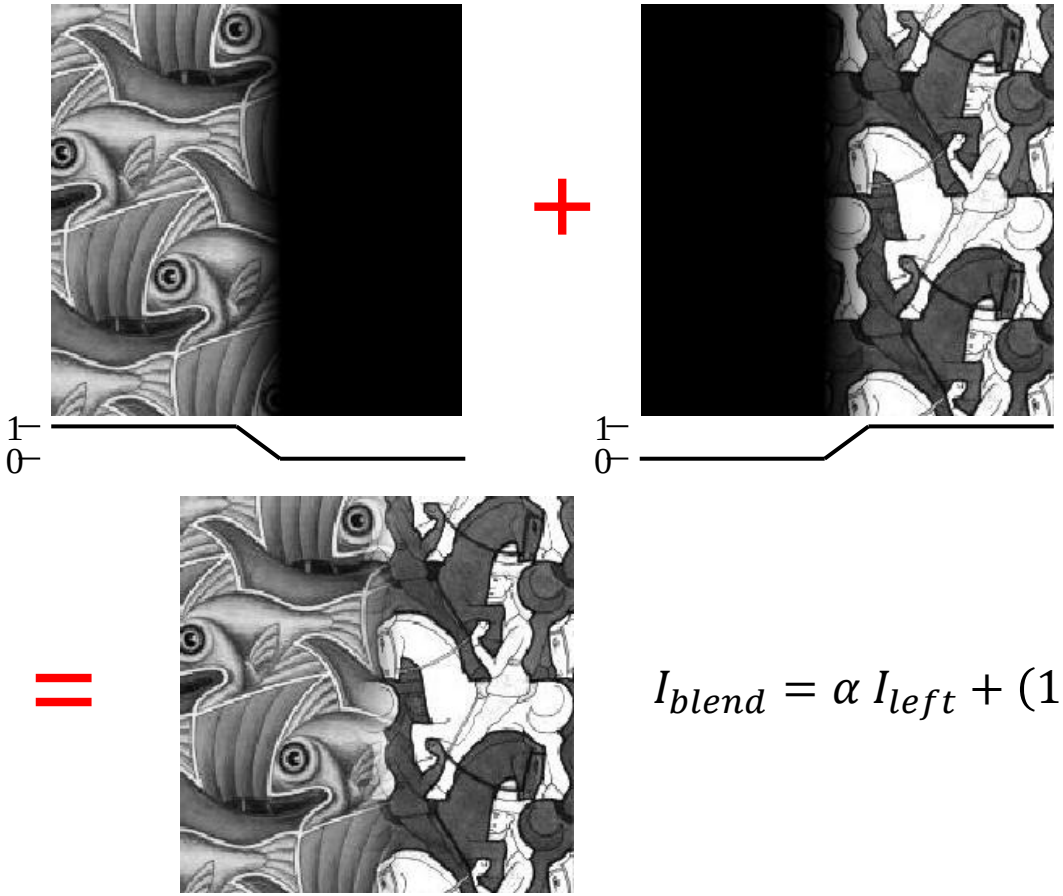
Slide: Derek Hoiem

Proper blending is key



Slide: Derek Hoiem

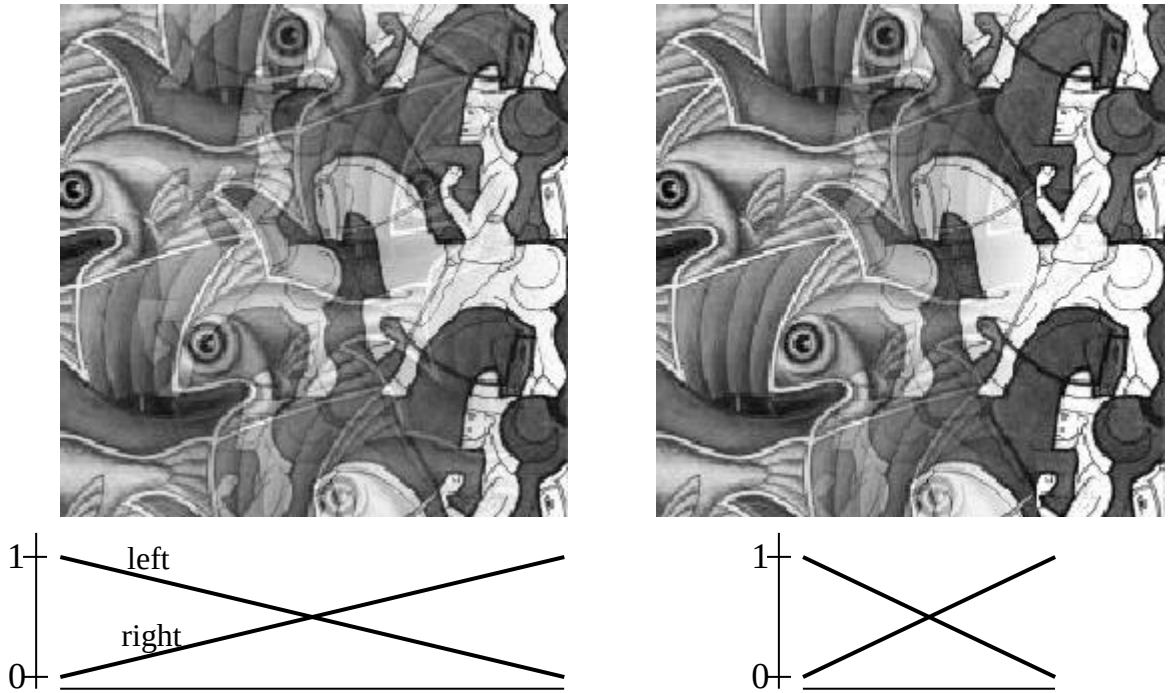
Alpha Blending / Feathering



$$I_{blend} = \alpha\, I_{left} + (1 - \alpha)\, I_{right}$$

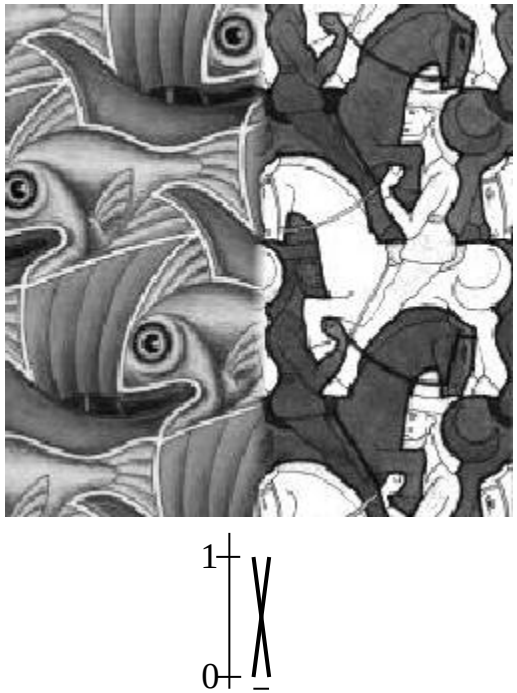
Slide: Derek Hoiem

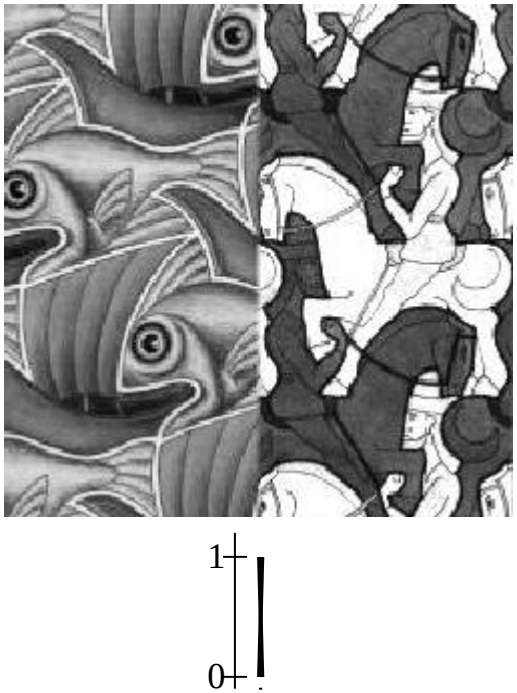
Effect of Window Size



Slide: Derek Hoiem

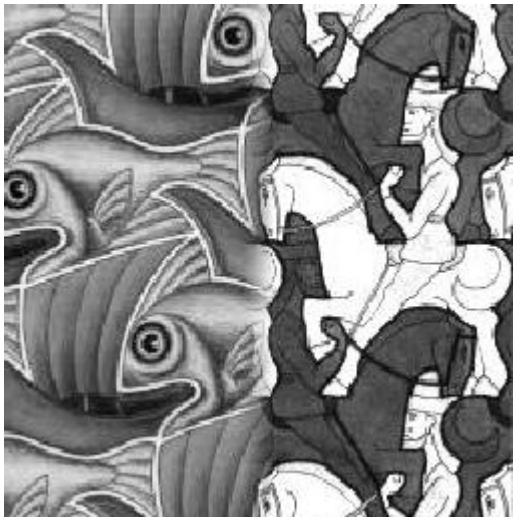
Effect of Window Size





Slide: Derek Hoiem

Good Window Size



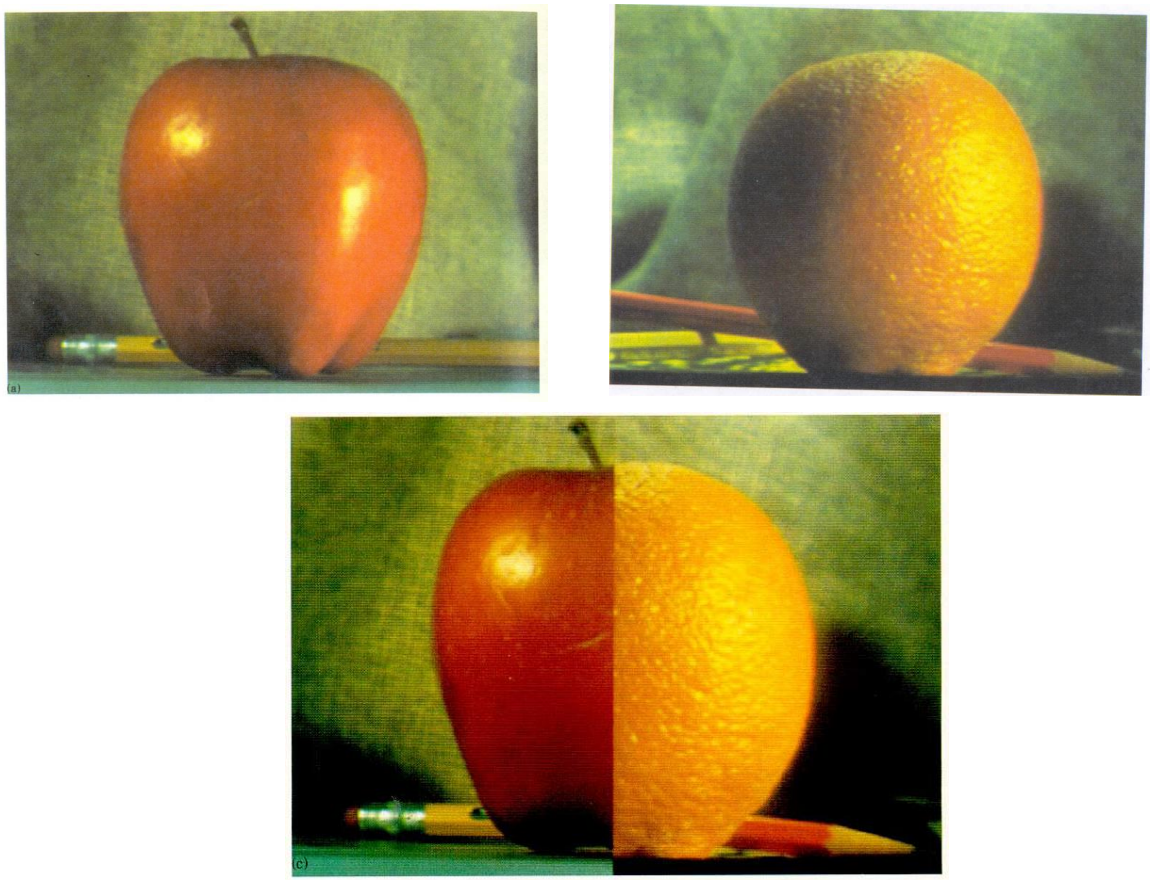
“Optimal” Window: smooth but not ghosted

Smooth images: wider area

More details: smaller area

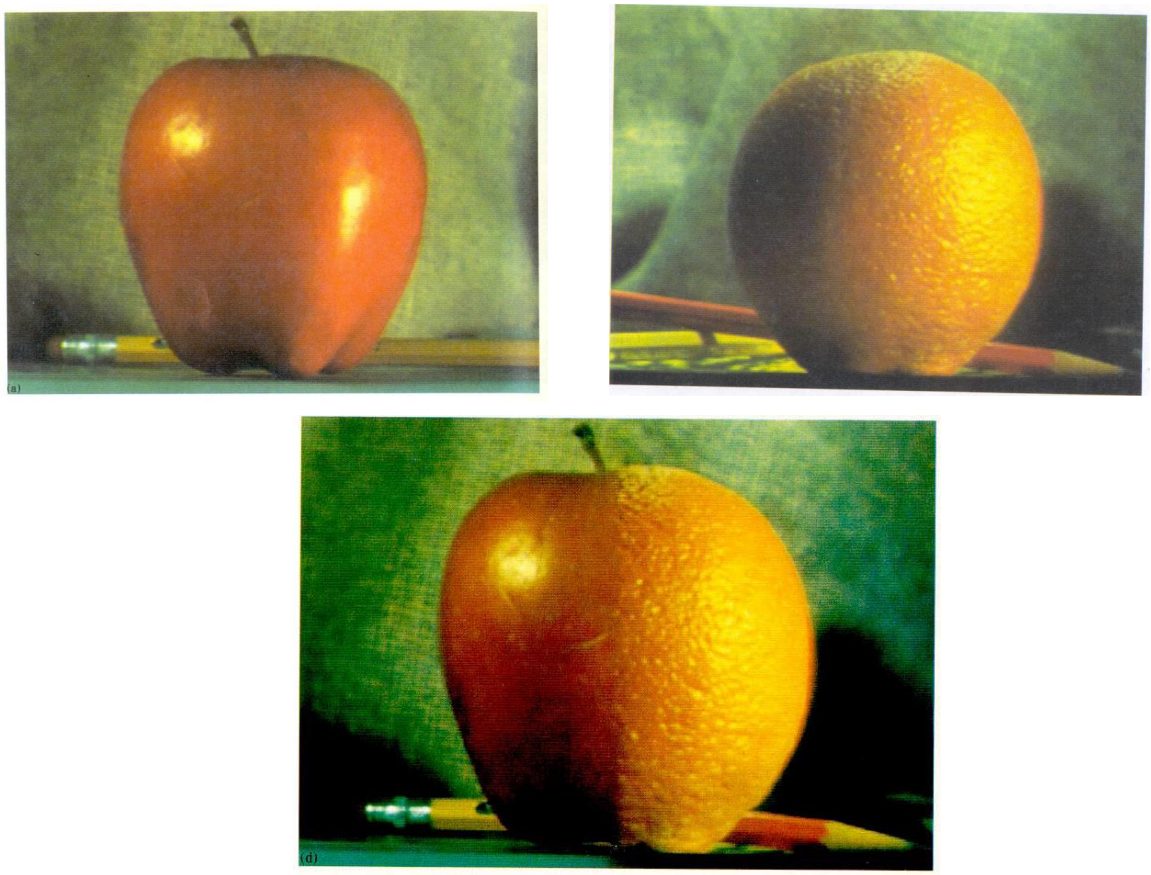
Slide: Derek Hoiem

How much should we blend?



Slide: Derek Hoiem

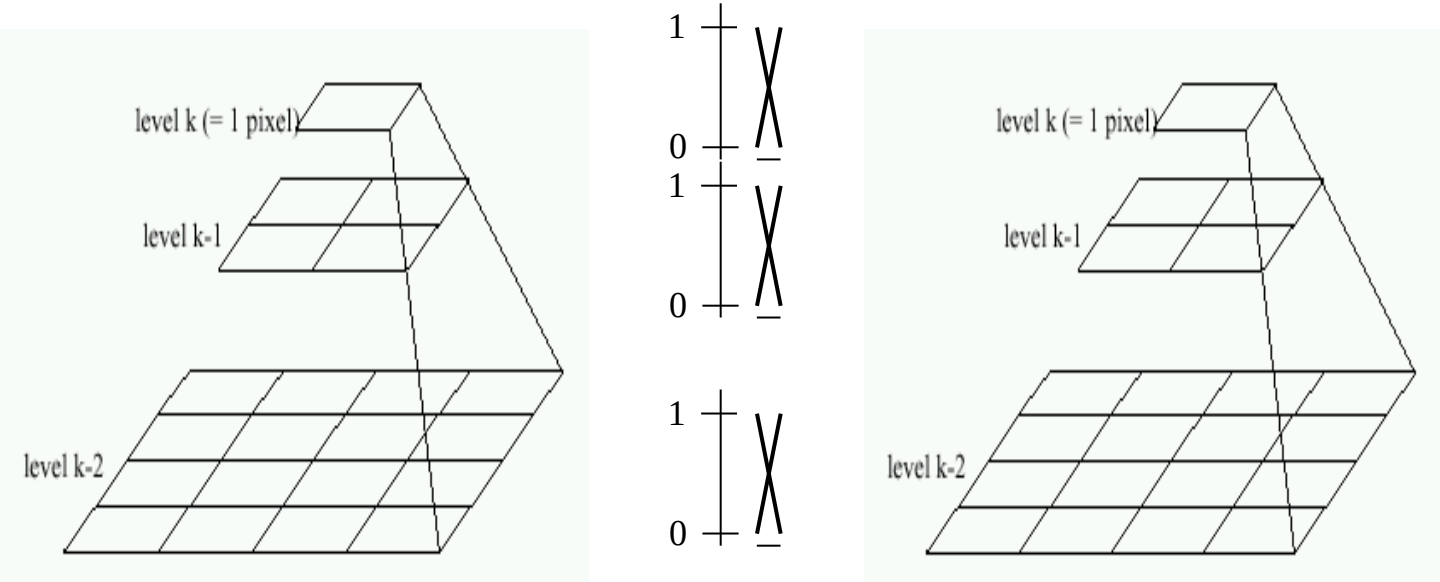
Method 2: Pyramid Blending



Slide: Derek Hoiem

Method 2: Pyramid Blending

- At low frequencies, blend slowly
- At high frequencies, blend quickly



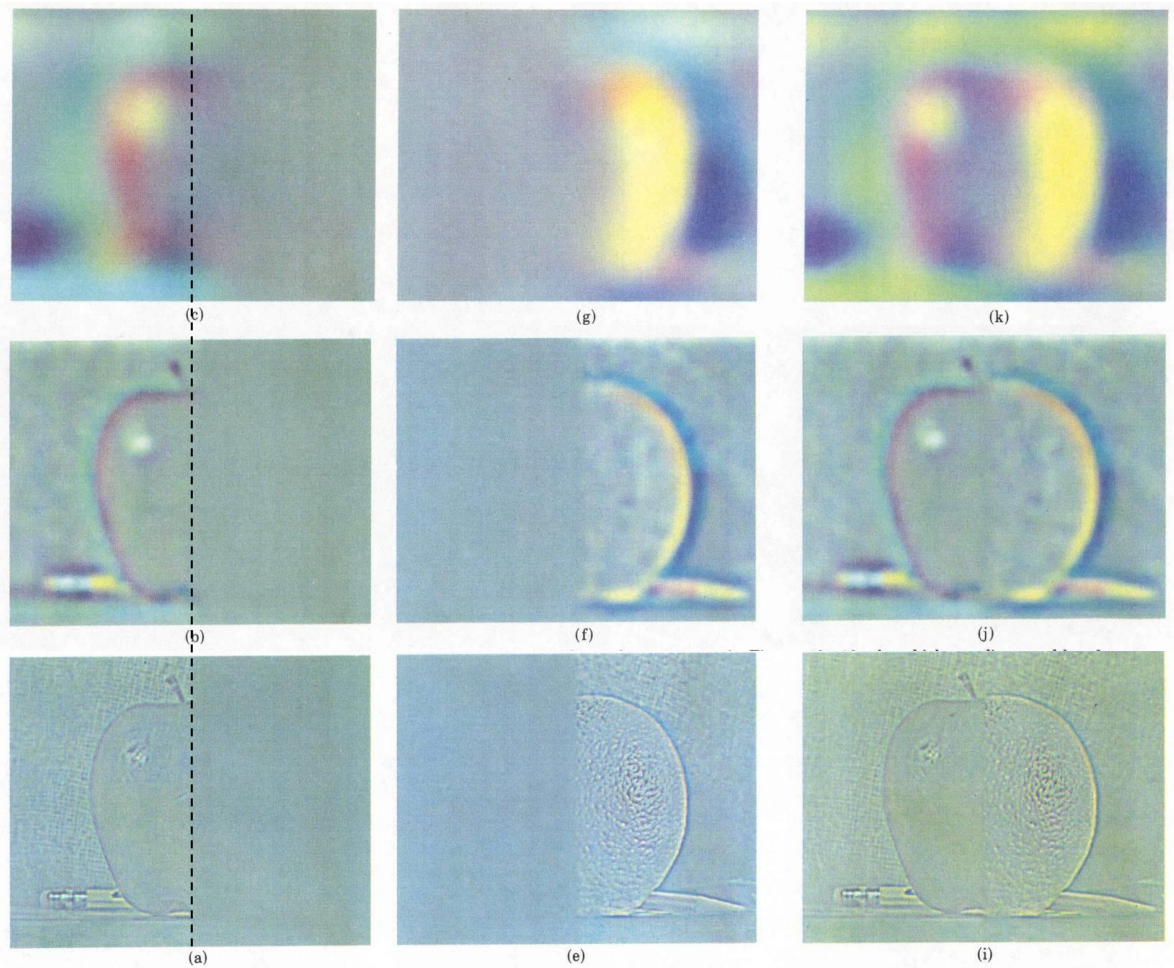
Left pyramid blend Right pyramid

Slide: Derek Hoiem

laplacian level 4

laplacian level 2

laplacian level 0



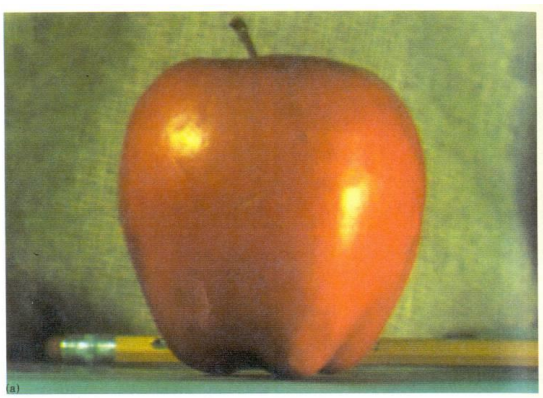
left pyramid | right pyramid | blended pyramid

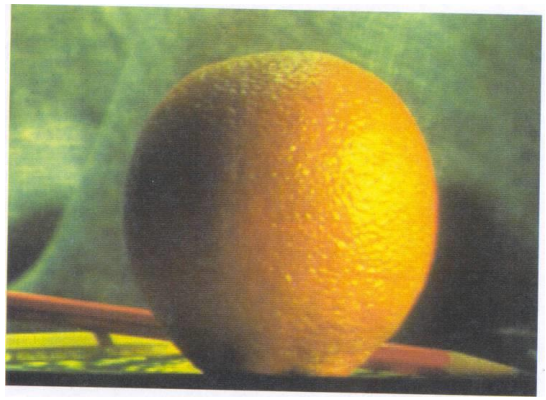
A Multiresolution Spline With Application to Image Mosaics

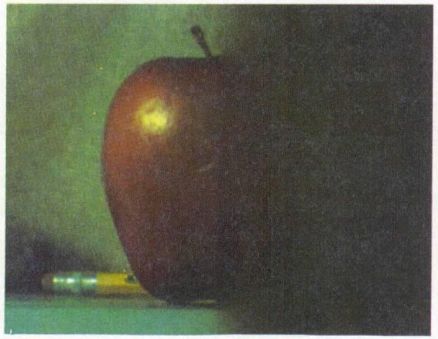
PETER J. BURT and EDWARD H. ADELSON
RCA David Sarnoff Research Center

Slide: Derek Hoiem

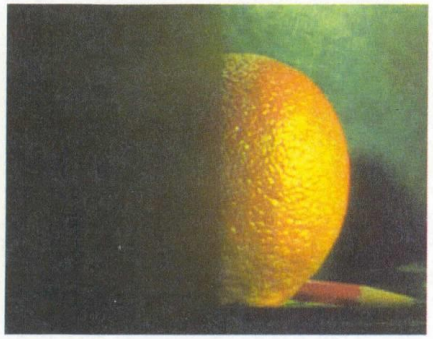
Method 2: Pyramid Blending



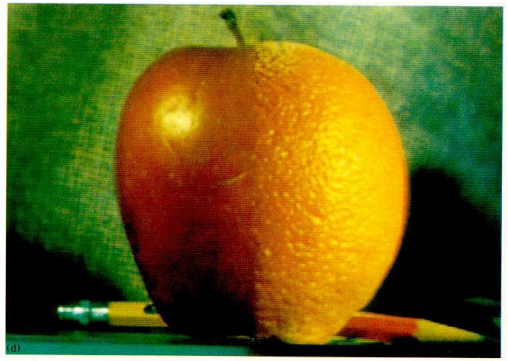




(d)



(h)



Burt and Adelson 1983

A Multiresolution Spline With Application to Image Mosaics

PETER J. BURT and EDWARD H. ADELSON
RCA David Sarnoff Research Center

Slide: Derek Hoiem

Laplacian Pyramid Blending

Implementation:

1. Build Laplacian pyramids for each image
2. Build a Gaussian pyramid of region mask
3. Blend each level of pyramid using region mask from the same level

$$L_{12}^{i} = L_{1}^{i} \cdot R^{i} + L_{2}^{i} \cdot (1 - R^{i})$$

Image 1 at level i of Laplacian pyramid

Pointwise multiply

Region mask at level i of Gaussian pyramid

4. Collapse the pyramid to get the final blended image

Burt and Adelson 1983

Slide: Derek Hoiem

Simplification: Two-band Blending

- Brown & Lowe, 2003
  - Only use two bands: high freq. and low freq.
  - Blends low freq. smoothly
  - Blend high freq. with no smoothing: use binary alpha

Slide: Derek Hoiem

2-band Blending

Low frequency

High frequency

Slide: Derek Hoiem

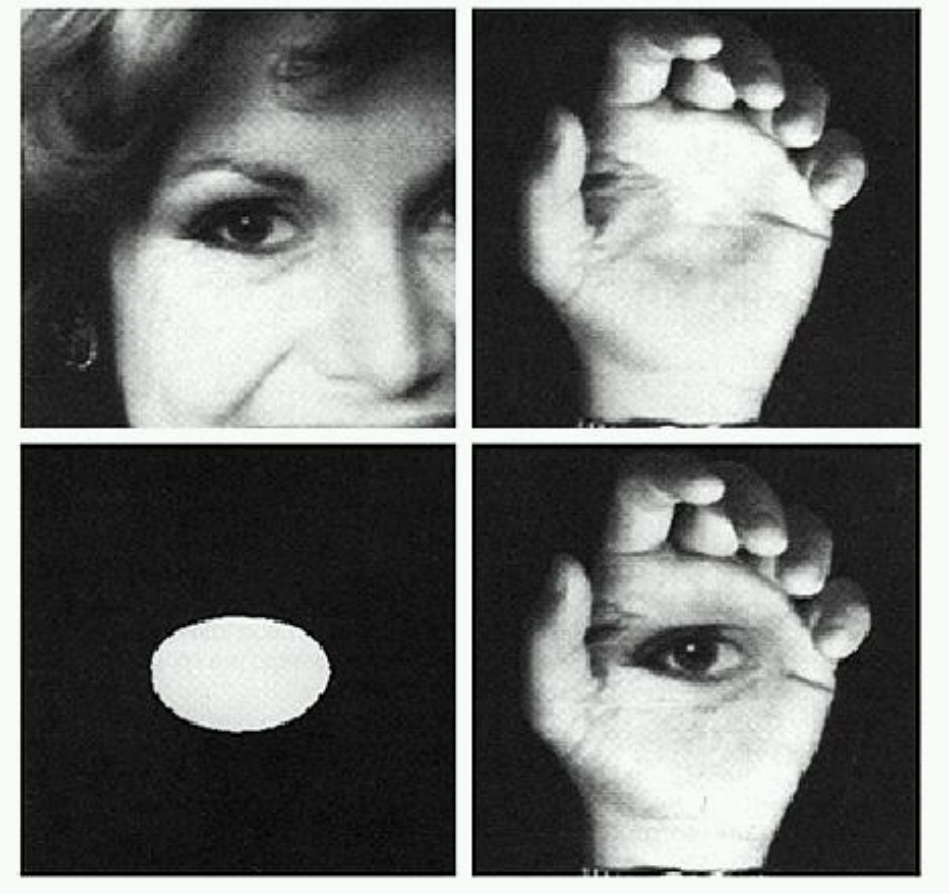
Linear Blending

Slide: Derek Hoiem

2-band Blending

Slide: Derek Hoiem

Blending Regions



Slide: Derek Hoiem

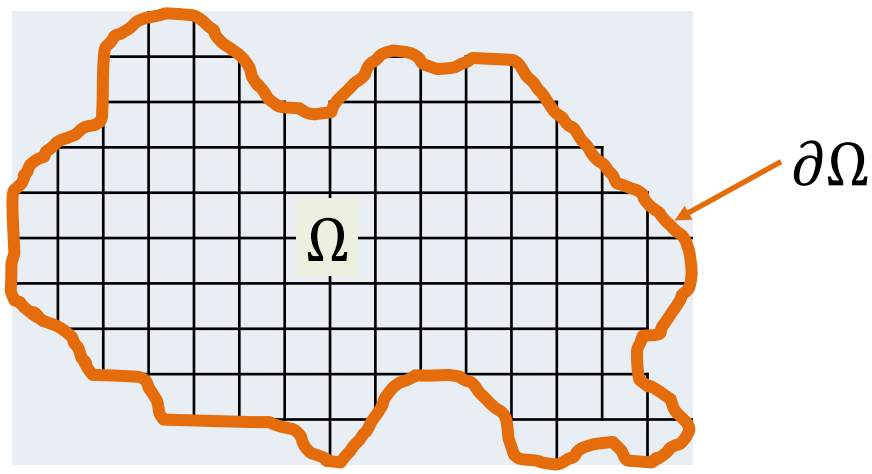
© Chris Cameron

Slide: Derek Hoiem

Gradient Domain Blending

2D Poisson Equation with Dirichlet Boundary Conditions

$$\begin{cases} \nabla^2 f = g \\ f|_{\partial\Omega} = h \end{cases} \longrightarrow \begin{cases} \nabla^2 f(x,y) = g(x,y) & (x,y) \in \Omega \\ f(x,y) = h(x,y) & (x,y) \in \partial\Omega \end{cases}$$



Numerical Solution, Finite Difference Method

$$\boxed{\begin{cases} \nabla^2 f(x,y) = g(x,y) & (x,y) \in \Omega \\ f(x,y) = h(x,y) & (x,y) \in \partial\Omega \end{cases}}$$

Laplacian $\longrightarrow$ $\boxed{\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}}$

Taylor expansion

$$f(x+\Delta x) = f(x) + \Delta x\, f'(x) + \frac{(\Delta x)^2}{2!} f''(x) + \frac{(\Delta x)^3}{3!} f^{(3)}(x) + \frac{(\Delta x)^4}{4!} f^{(4)}(x) + \cdots$$

$$\begin{cases} f(x_{i+1}, y_j) = f(x_i, y_j) + \frac{\partial f}{\partial x}(x_i, y_j) + \frac{1}{2!}\frac{\partial^2 f}{\partial x^2}(x_i, y_j) + \frac{1}{3!}\frac{\partial^3 f}{\partial x^3}(x_i, y_j) + \frac{1}{4!}\frac{\partial^4 f}{\partial x^4}(x_i, y_j) + \cdots \\ f(x_{i-1}, y_j) = f(x_i, y_j) - \frac{\partial f}{\partial x}(x_i, y_j) + \frac{1}{2!}\frac{\partial^2 f}{\partial x^2}(x_i, y_j) - \frac{1}{3!}\frac{\partial^3 f}{\partial x^3}(x_i, y_j) + \frac{1}{4!}\frac{\partial^4 f}{\partial x^4}(x_i, y_j) + \cdots \\ f(x_i, y_{j+1}) = f(x_i, y_j) + \frac{\partial f}{\partial y}(x_i, y_j) + \frac{1}{2!}\frac{\partial^2 f}{\partial y^2}(x_i, y_j) + \frac{1}{3!}\frac{\partial^3 f}{\partial y^3}(x_i, y_j) + \frac{1}{4!}\frac{\partial^4 f}{\partial y^4}(x_i, y_j) + \cdots \\ f(x_i, y_{j-1}) = f(x_i, y_j) - \frac{\partial f}{\partial y}(x_i, y_j) + \frac{1}{2!}\frac{\partial^2 f}{\partial y^2}(x_i, y_j) - \frac{1}{3!}\frac{\partial^3 f}{\partial y^3}(x_i, y_j) + \frac{1}{4!}\frac{\partial^4 f}{\partial y^4}(x_i, y_j) + \cdots \end{cases}$$

$$f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1} - 4\, f_{i,j} = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + error$$

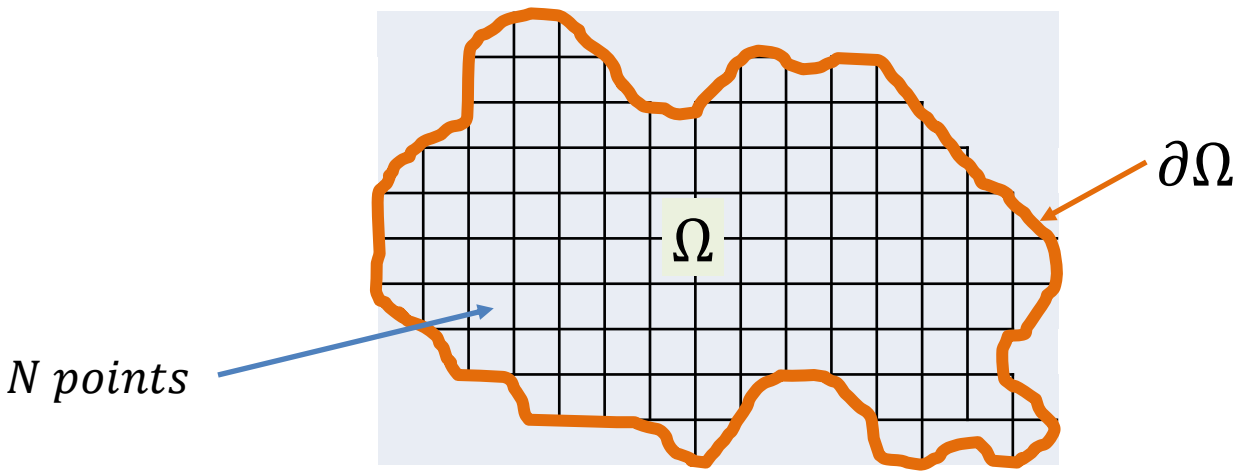
Numerical Solution, Finite Difference Method

$$\begin{cases} \nabla^2 f(x,y) = g(x,y) & (x,y) \in \Omega \\ f(x,y) = h(x,y) & (x,y) \in \partial\Omega \end{cases}$$

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1} - 4\, f_{i,j} = g_{i,j}$$

$$\boxed{\begin{array}{c} N\ unknowns, N\ equations \\ A\, f = b \\ system\ of\ linear\ equations \end{array}}$$



Motivation for gradient-domain filtering?

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Motivation for gradient-domain filtering?

- Important in human perception

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Motivation for gradient-domain filtering?

- Important in human perception
  - human visual system more sensitive to local contrast than absolute pixel-value

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

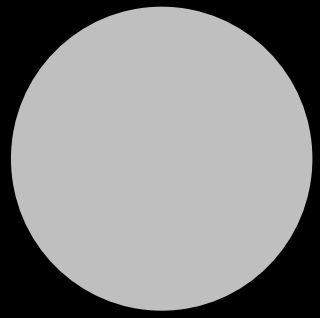
Motivation for gradient-domain filtering?

- Important in human perception
  - human visual system more sensitive to local contrast than absolute pixel-value
  - a demonstration

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

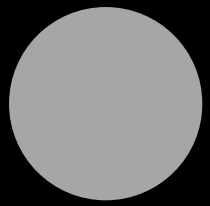


Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

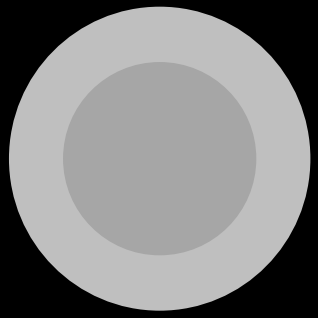
What were the grayscale values?

What were the grayscale values?
Using these values,
can you tell which was brighter?

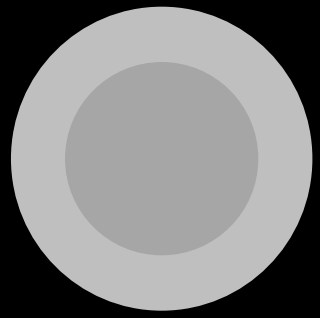
Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering



Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

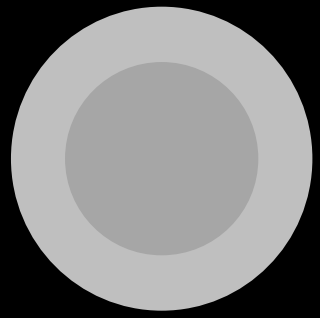


Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering



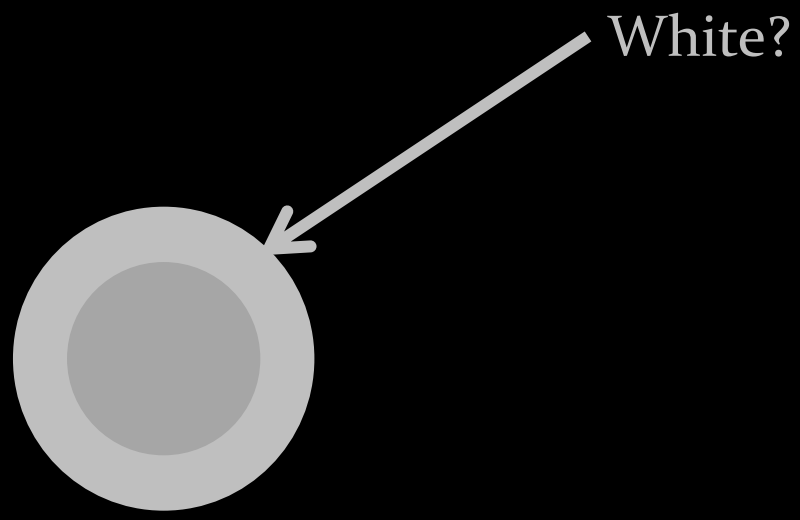
Which one is brighter?

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

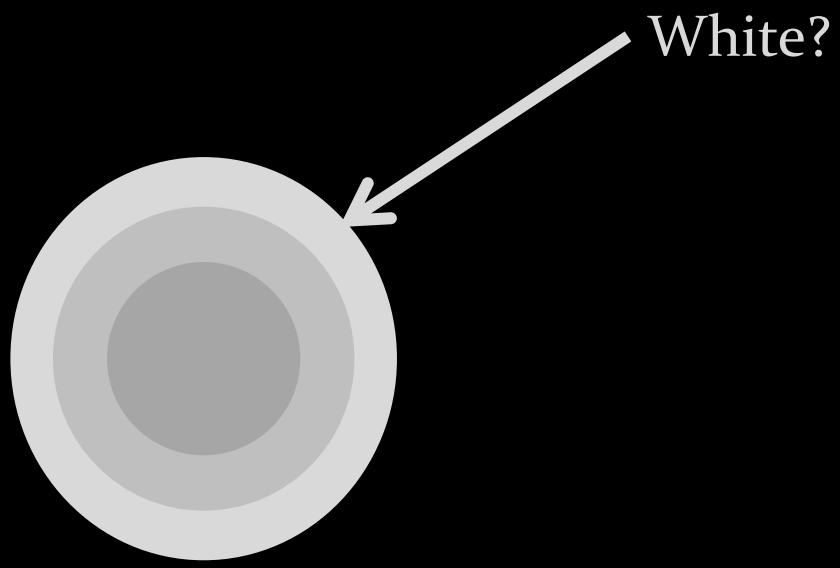


Which one is brighter?
Which one is white?

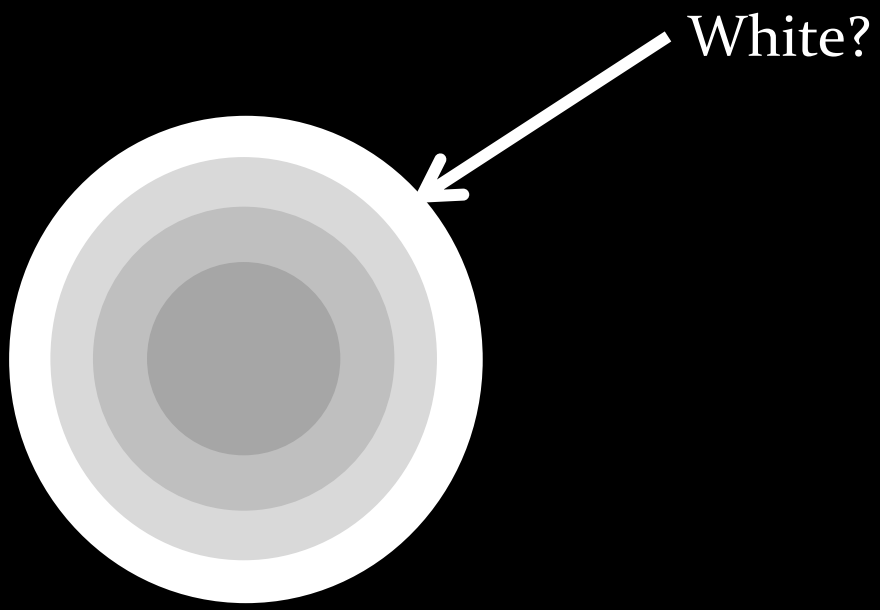
Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering



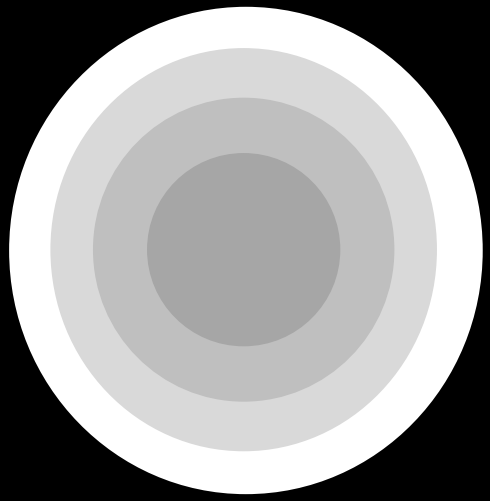
Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering



Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering



Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering



You think that's air
you're breathing?

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Take home message:
Gradients are really important
in image perception

That 's All Folks

Gradient manipulation

Idea:

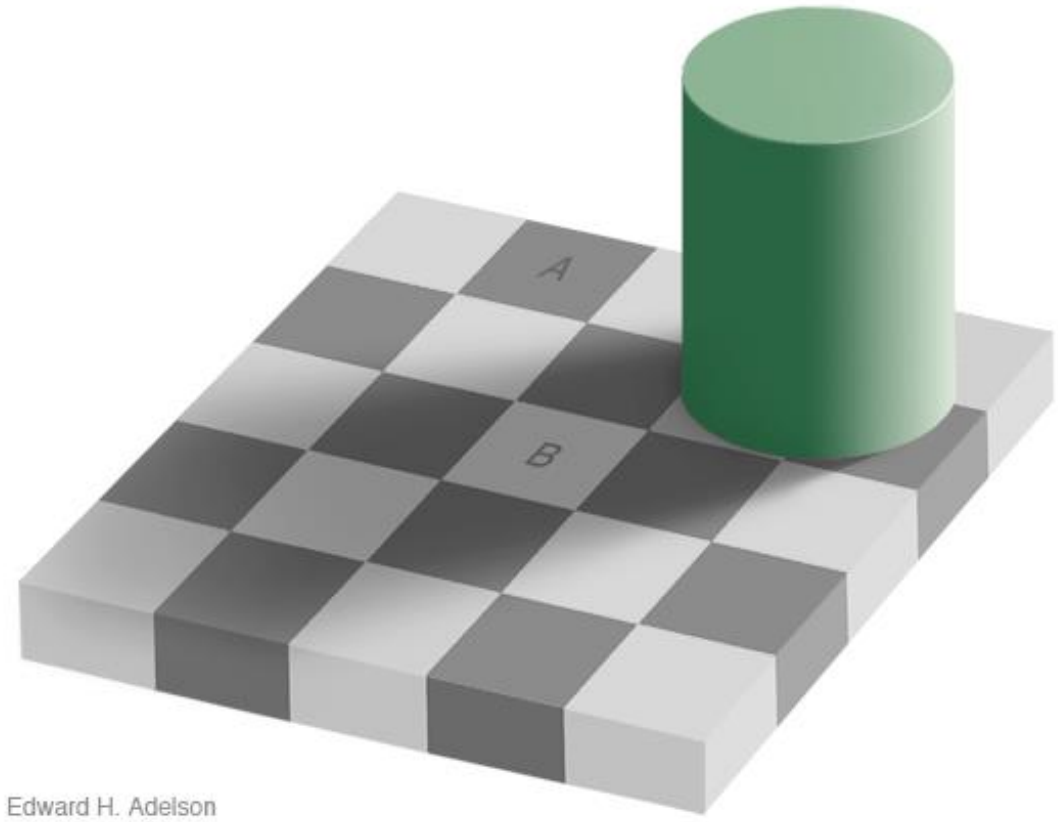
- Human visual system is very sensitive to gradient
- Gradient encodes edges and local contrast quite well

- Do your editing in the gradient domain
- Reconstruct image from gradient

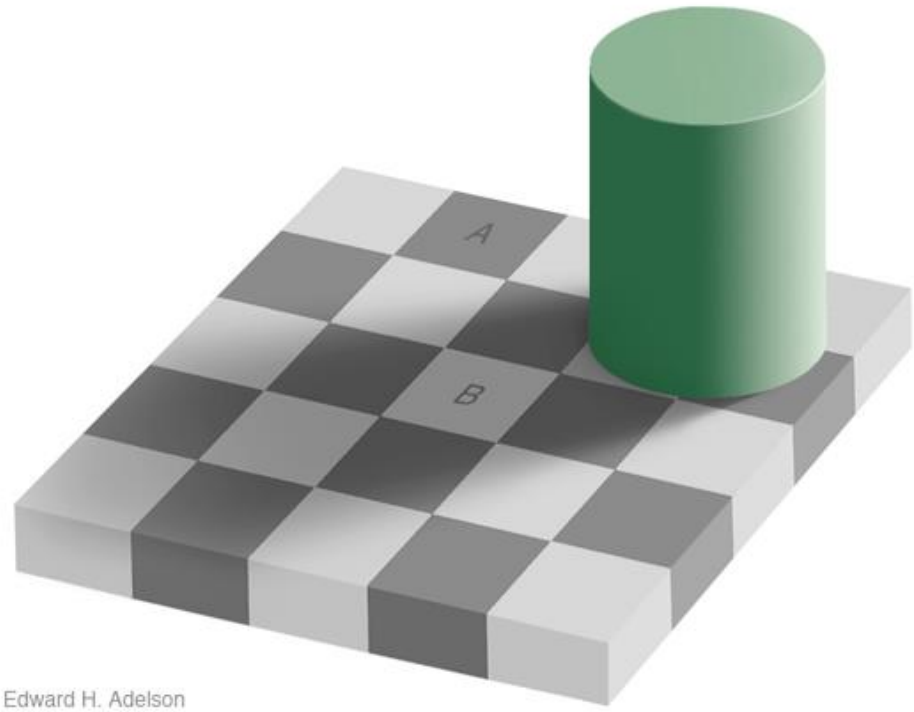
- Various instances of this idea, I'll mostly follow Perez et al. Siggraph 2003
  http://research.microsoft.com/vision/cambridge/papers/perez_siggraph03.pdf

Slide credit: F. Durand

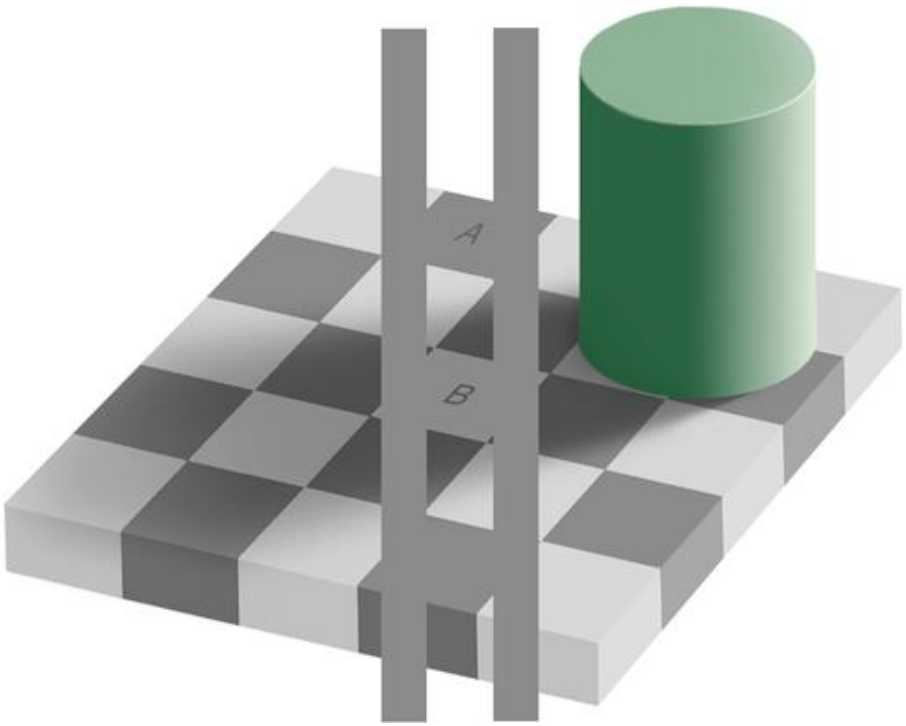
Which is perceptually meaningful



Consider the pixel difference between A and B; is it large or small?



Edward H. Adelson



Idea: we'll manipulate gradient field of image rather than image

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Related idea: Poisson Blending

A good blend should preserve gradients of source region without changing the background

Perez et al. 2003

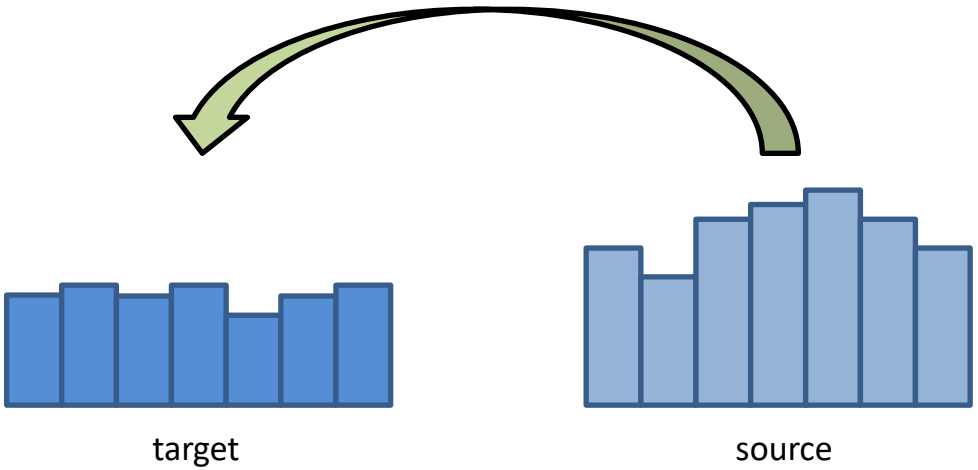
Slide: Derek Hoiem

Related idea: Poisson Blending

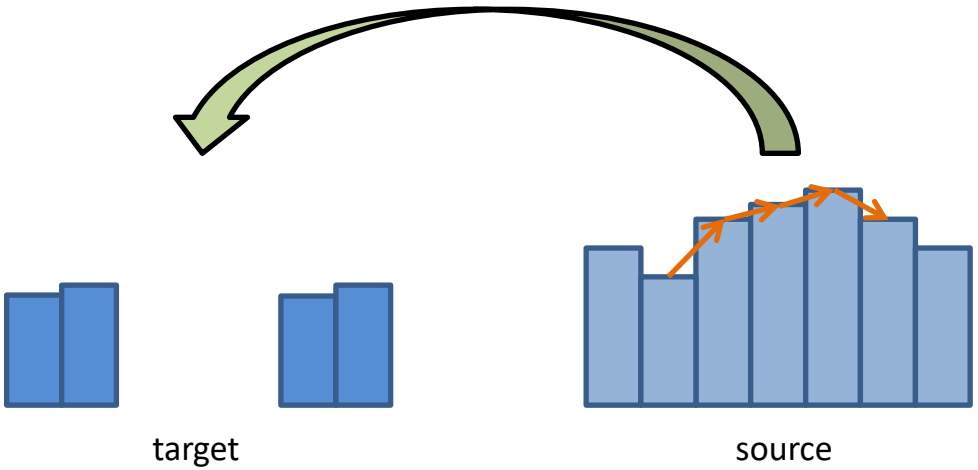
A good blend should preserve gradients of source region without changing the background

Perez et al. 2003

Slide: Derek Hoiem



Slide: James Hays, James Tompkins



It is impossible to faithfully preserve the gradients

Slide: James Hays, James Tompkins

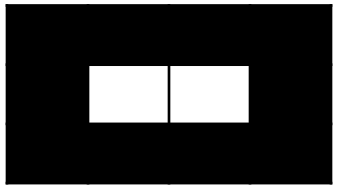
Simple 2d example

.2	.5	.2	.2
.7	.7	.7	.7
.9	.9	.8	.9

target, t

.8	.6	.6	.6
.6	.6	.2	.6
.6	.8	.6	.6

source, s



mask

?	?	?	?
?	?	?	?
?	?	?	?

output, x

What properties do we want x to have?

Slide: James Hays, James Tompkins

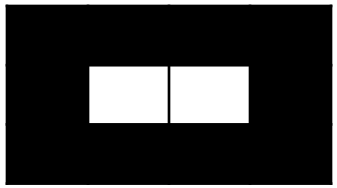
Simple 2d example

.2	.5	.2	.2
.7	.7	.7	.7
.9	.9	.8	.9

target, t

.8	.6	.6	.6
.6	.6	.2	.6
.6	.8	.6	.6

source, s



mask

?	?	?	?
?	?	?	?
?	?	?	?

output, x

(1) For unmasked pixels, $x_i = t_i$
(2) For masked pixels, we want the gradients at x_i to match the gradients at s_i

But how do we define the gradient? Instead of constraining one or many gradients, in this example we will use the Laplacian.

0	-1	0
-1	4	-1
0	-1	0

Slide: James Hays, James Tompkins

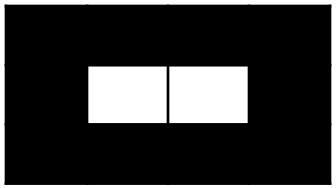
Simple 2d example

.2	.5	.2	.2
.7	.7	.7	.7
.9	.9	.8	.9

target, t

.8	.6	.6	.6
.6	.6	.2	.6
.6	.8	.6	.6

source, s



mask

0	-1	0
-1	4	-1
0	-1	0

Laplacian

?	?	?	?
?	?	?	?
?	?	?	?

output, x

1	4	7	10
2	5	8	11
3	6	9	12

Pixel indexing

$$
\begin{aligned}
x_1 &= t_1 \\
x_2 &= t_2 \\
x_3 &= t_3 \\
x_4 &= t_4 \\
4x_5 - x_4 - x_2 - x_6 - x_8 &= 4s_5 - s_4 - s_2 - s_6 - s_8 \\
x_6 &= t_6 \\
x_7 &= t_7 \\
4x_8 - x_7 - x_5 - x_9 - x_{11} &= 4s_8 - s_7 - s_5 - s_9 - s_{11} \\
x_9 &= t_9 \\
x_{10} &= t_{10} \\
x_{11} &= t_{11} \\
x_{12} &= t_{12}
\end{aligned}
$$

Slide: James Hays, James Tompkins

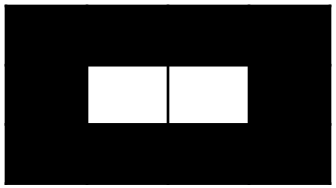
Simple 2d example

.2	.5	.2	.2
.7	.7	.7	.7
.9	.9	.8	.9

target, t

.8	.6	.6	.6
.6	.6	.2	.6
.6	.8	.6	.6

source, s



mask

0	-1	0
-1	4	-1
0	-1	0

Laplacian

?	?	?	?
?	?	?	?
?	?	?	?

output, x

1	4	7	10
2	5	8	11
3	6	9	12

Pixel indexing

$$
\begin{aligned}
x_1 &= 0.2 \\
x_2 &= 0.7 \\
x_3 &= 0.9 \\
x_4 &= 0.5 \\
4x_5 - x_4 - x_2 - x_6 - x_8 &= 0.2 \\
x_6 &= 0.9 \\
x_7 &= 0.2 \\
4x_8 - x_7 - x_5 - x_9 - x_{11} &= -1.6 \\
x_9 &= 0.8 \\
x_{10} &= 0.2 \\
x_{11} &= 0.7 \\
x_{12} &= 0.9
\end{aligned}
$$

Slide: James Hays, James Tompkins

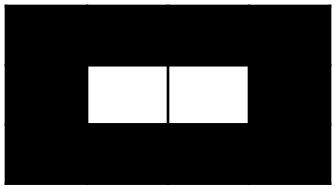
Simple 2d example

.2	.5	.2	.2
.7	.7	.7	.7
.9	.9	.8	.9

target, t

.8	.6	.6	.6
.6	.6	.2	.6
.6	.8	.6	.6

source, s



mask

0	-1	0
-1	4	-1
0	-1	0

Laplacian

?	?	?	?
?	?	?	?
?	?	?	?

output, x

1	4	7	10
2	5	8	11
3	6	9	12

Pixel indexing

$$x = A\backslash \mathrm{b}$$

1											
	1										
		1									
			1								
	-1		-1	4	-1		-1				
					1						
						1					
				-1		-1	4	-1		-1	
								1			
									1		
										1	
											1

A

*

?
?
?
?
?
?
?
?
?
?
?
?

x

=

.2
.7
.9
.5
.2
.9
.2
-1.6
.8
.2
.7
.9

b

Slide: James Hays, James Tompkins

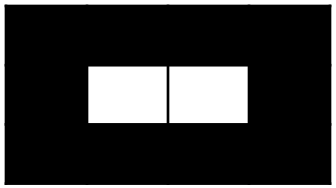
Simple 2d example

.2	.5	.2	.2
.7	.7	.7	.7
.9	.9	.8	.9

target, t

.8	.6	.6	.6
.6	.6	.2	.6
.6	.8	.6	.6

source, s



mask

0	-1	0
-1	4	-1
0	-1	0

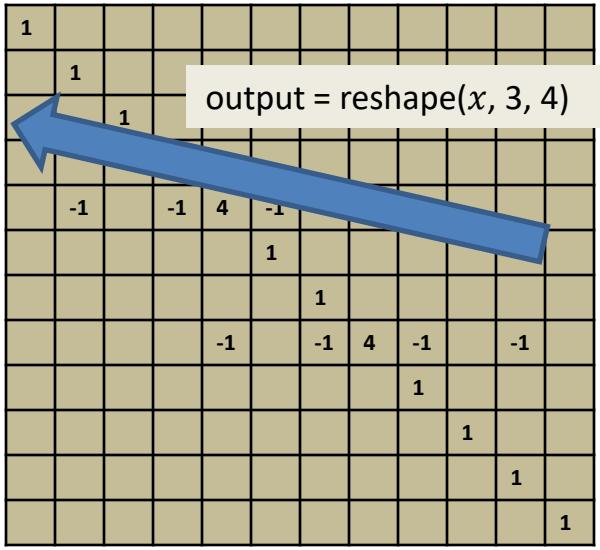
Laplacian

.2	.5	.2	.2
.7	.62	.18	.7
.9	.9	.8	.9

output, x

1	4	7	10
2	5	8	11
3	6	9	12

Pixel indexing



A

$*$

.2
.7
.9
.5
.62
.9
.2
.18
.8
.2
.7
.9

x

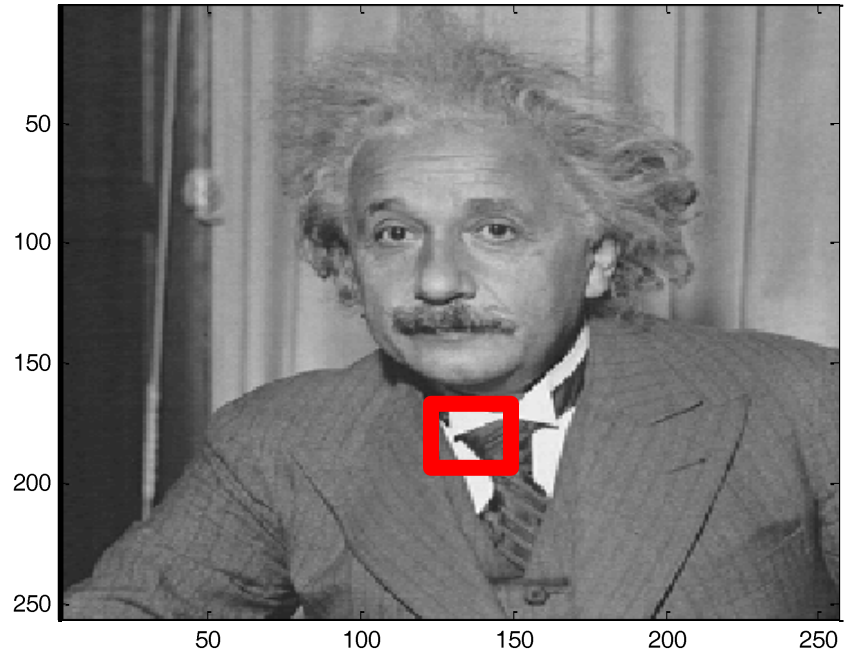
$=$

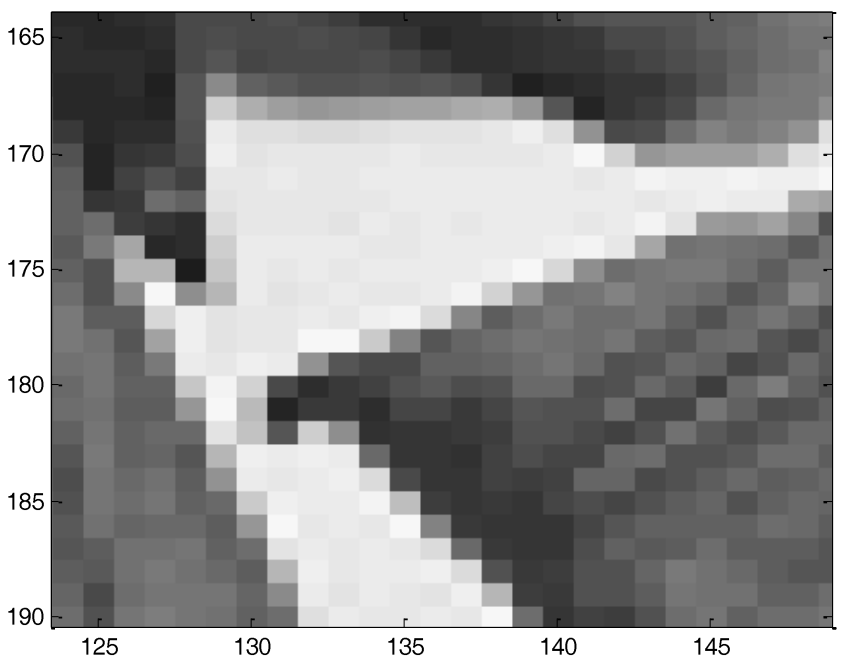
.2
.7
.9
.5
.2
.9
.2
-1.6
.8
.2
.7
.9

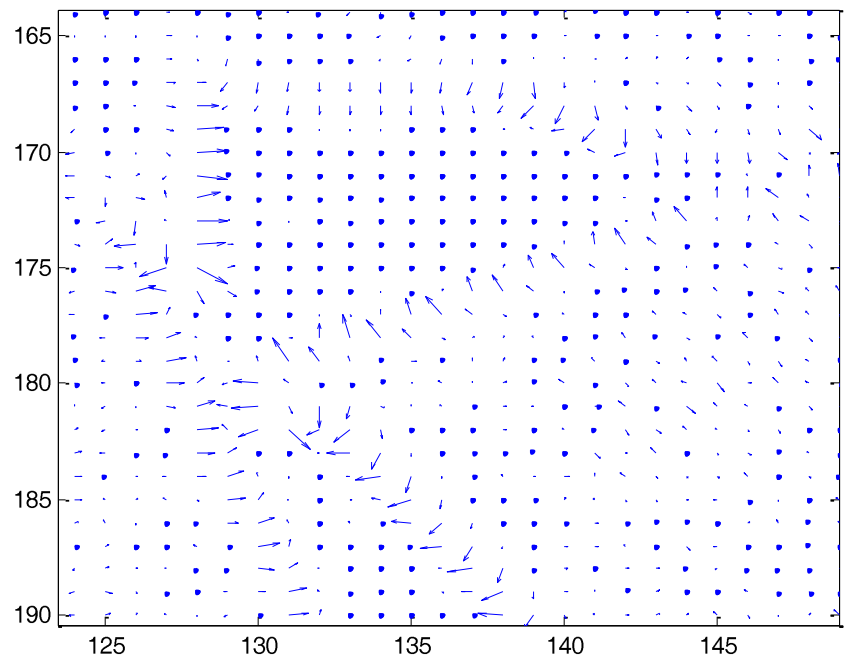
b

Slide: James Hays, James Tompkins

Gradient domain view of image

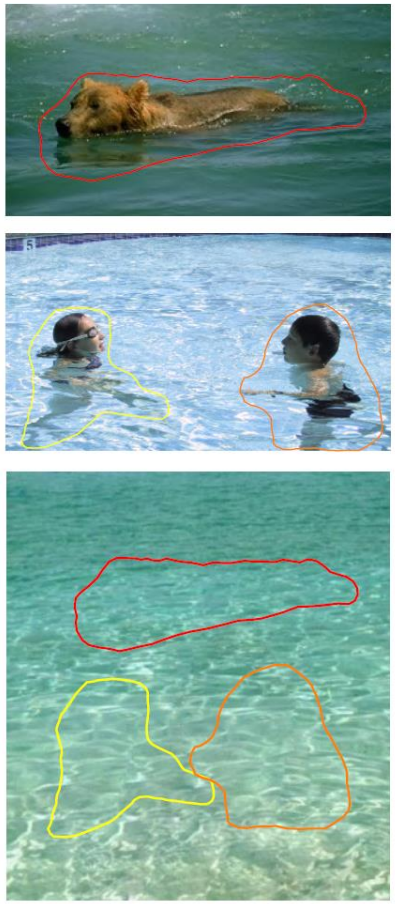






Slide: Rob Fergus

Gradient domain cloning



sources/destinations

cloning

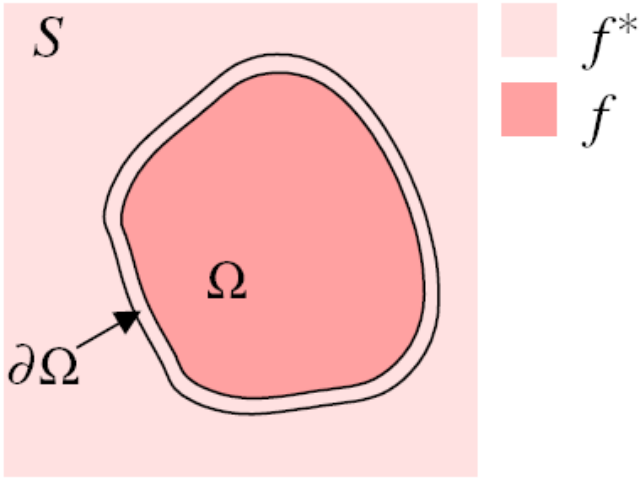
seamless cloning

Slide: Rob Fergus

Membrane interpolation

- Laplace equation (a.k.a. membrane equation)

$$\min_f \iint_\Omega |\nabla f|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$



Slide credit: F. Durand

Membrane interpolation

- Laplace equation (a.k.a. membrane equation)

$$\min_f \iint_\Omega |\nabla f|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

- Mathematicians will tell you there is Associated Euler-Lagrange equation:

$$\Delta f = 0 \text{ over } \Omega \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

  – Where the Laplacian Δ is similar to -1 2 -1 in 1D
- Kind of the idea that we want a minimum, so we kind of derive and get a simpler equation

Slide credit: F. Durand

Seamless Poisson cloning

- Given vector field v (pasted gradient), find the value of f in unknown region that optimize:
- Previously, v was null

$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

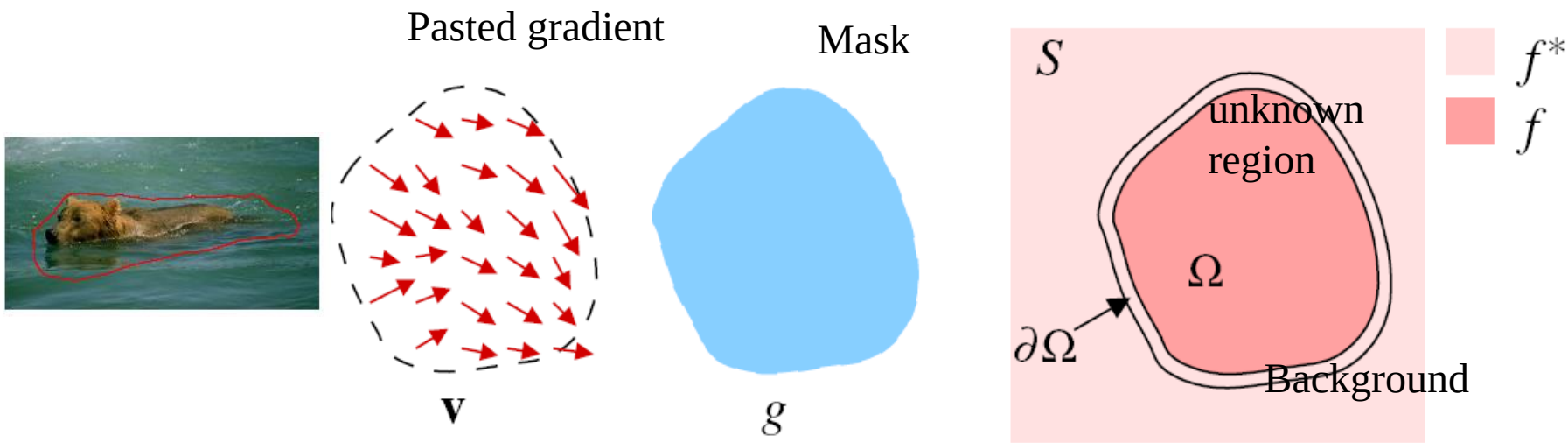


Figure 1: **Guided interpolation notations**. Unknown function f interpolates in domain Ω the destination function f^*, under guidance of vector field $\mathbf{v}$, which might be or not the gradient field of a source function g.

Slide credit: F. Durand

What if v is not null: 2D

- Variational minimization (integral of a functional) with boundary condition

$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \quad with \quad f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

- Euler-Lagrange equation:

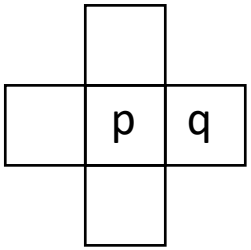
$$\Delta f = \operatorname{div} \mathbf{v} \quad \text{over } \Omega, \text{with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

$$\text{where } \operatorname{div} \mathbf{v} = \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} \text{ is the divergence of } \mathbf{v} = (u, v)$$

- (Compared to Laplace, we have replaced $\Delta = 0$ by $\Delta = \operatorname{div}$)

Slide credit: F. Durand

Discrete Poisson Solver



$$\text{for all } p \in \Omega \quad |N_p| f_p - \sum_{q \in N_p \cap \Omega} f_q = \underbrace{\sum_{q \in N_p \cap \partial\Omega} f_q^*}_{\text{Only for boundary pixels}} + \sum_{q \in N_p} v_{pq}$$

Big yet sparse linear system

Known:

$$v_{pq} = gradient(p) - gradient(q)$$

f^* = constrained border pixels

N_p = list of neighbors of pixel p

Unknown:

f = reconstructed pixels

Write equation as $Af = b$

- A is a large space matrix mostly of 0s and ± 1s
- b is a vector of the righthandside of the above equation

Matlab can solve (for small images) in a single command: $f = A\backslash \mathrm{b}$

Slide credit: F. Durand

Other results

sources | destinations | cloning | seamless cloning

Perez et al. 2003

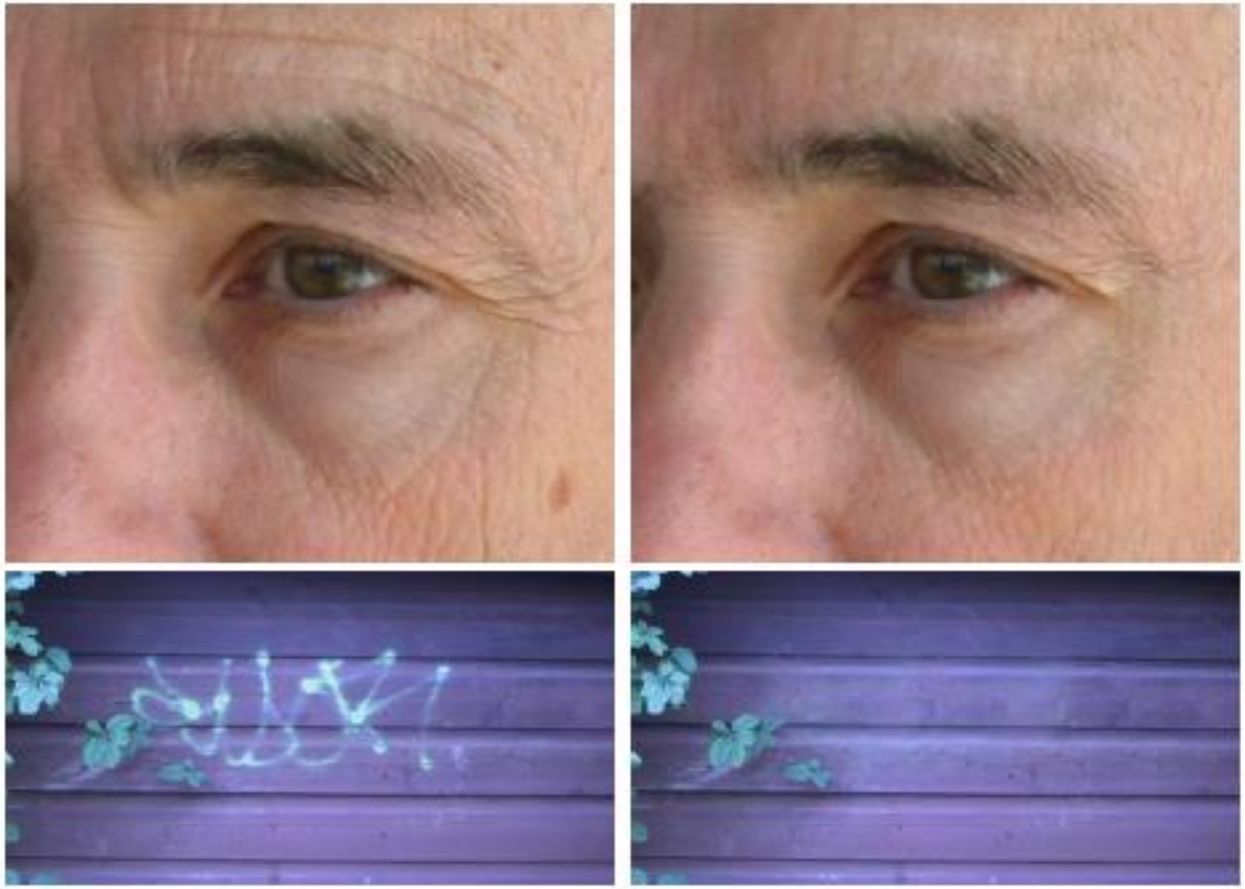
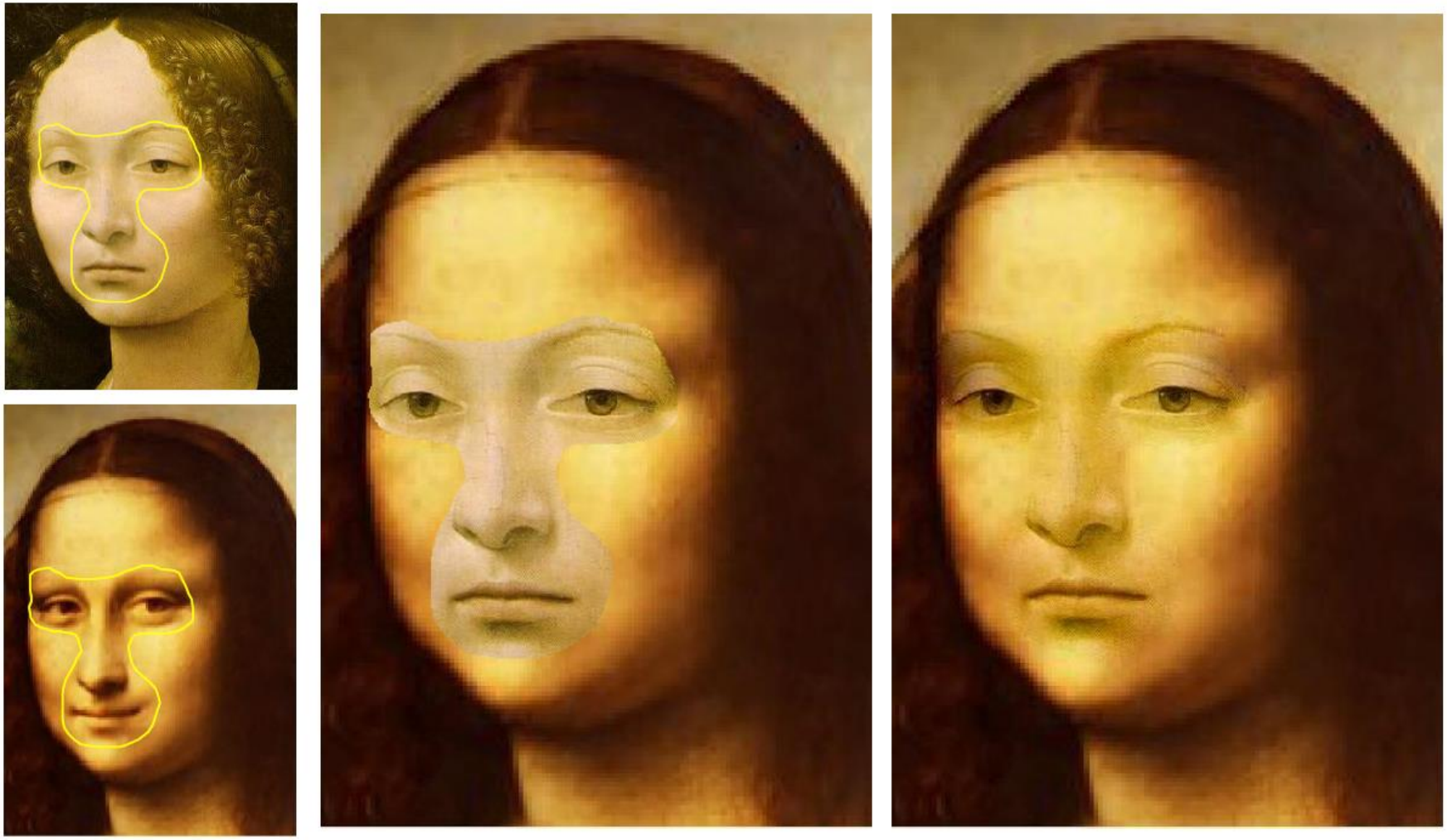


Figure 2: **Concealment**. By importing seamlessly a piece of the background, complete objects, parts of objects, and undesirable artifacts can easily be hidden. In both examples, multiple strokes (not shown) were used.

Perez et al. 2003

Result (eye candy)

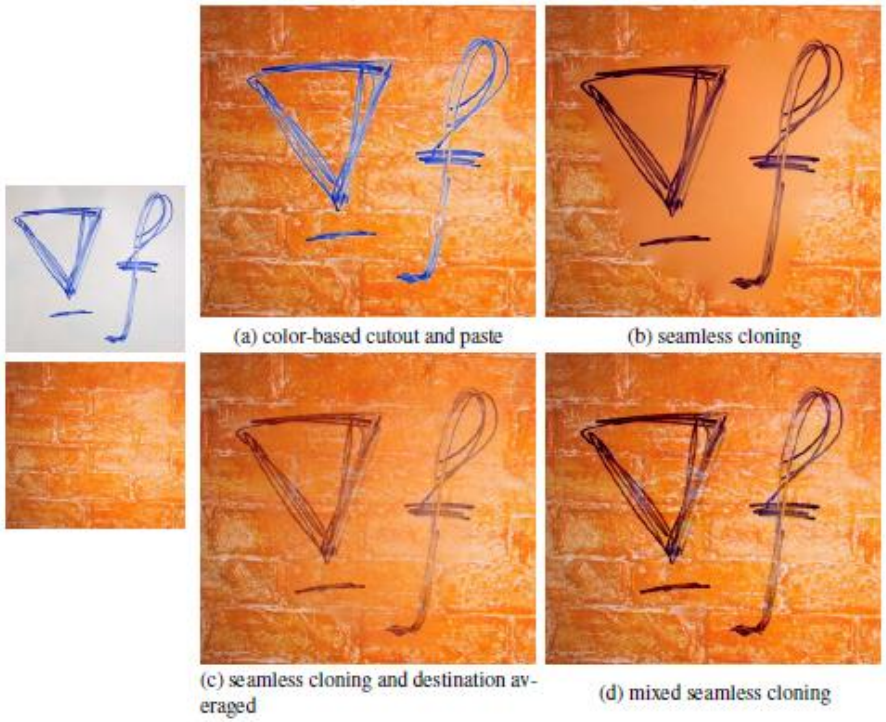


source/destination cloning seamless cloning

Perez et al. 2003

Blending with Mixed Gradients

- Use foreground or background gradient with larger magnitude as the guiding gradient



$$\text{for all } \mathbf{x} \in \Omega, \mathbf{v}(\mathbf{x}) = \begin{cases} \nabla f^*(\mathbf{x}) & \text{if } |\nabla f^*(\mathbf{x})| > |\nabla g(\mathbf{x})| \\ \nabla g(\mathbf{x}) & \text{otherwise} \end{cases}$$

Perez et al. 2003

Manipulate the gradient

-

source/destination seamless cloning mixed seamless cloning

Figure 8: **Inserting one object close to another**. With seamless cloning, an object in the destination image touching the selected region Ω bleeds into it. Bleeding is inhibited by using mixed gradients as the guidance field.

Perez et al. 2003

GradientShop

- Optimization framework

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

GradientShop

- Optimization framework
  - Input unfiltered image – ***u***

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

GradientShop

- Optimization framework
  - Input unfiltered image – ***u***
  - Output filtered image – ***f***

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

GradientShop

- Optimization framework
  - Input unfiltered image – $\boldsymbol{u}$
  - Output filtered image – $\boldsymbol{f}$
  - Specify desired pixel-differences – $(\boldsymbol{g}^x, \boldsymbol{g}^y)$

Energy function

$$\min_{f} \quad (\boldsymbol{f}_x - \boldsymbol{g}^x)^2 + (\boldsymbol{f}_y - \boldsymbol{g}^y)^2$$

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

GradientShop

- Optimization framework
  - Input unfiltered image – $\boldsymbol{u}$
  - Output filtered image – $\boldsymbol{f}$
  - Specify desired pixel-differences – $(\boldsymbol{g}^x, \boldsymbol{g}^y)$
  - Specify desired pixel-values – $\boldsymbol{d}$

Energy function

$$\min_{f} \quad (\boldsymbol{f}_x - \boldsymbol{g}^x)^2 + (\boldsymbol{f}_y - \boldsymbol{g}^y)^2 + (\boldsymbol{f} - \boldsymbol{d})^2$$

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

GradientShop

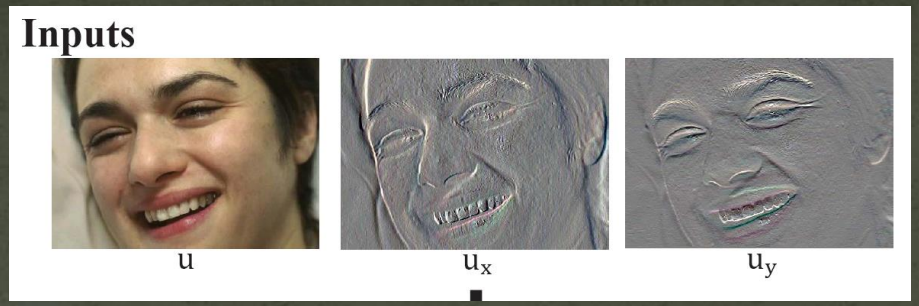
- Optimization framework
  - Input unfiltered image – $\boldsymbol{u}$
  - Output filtered image – $\boldsymbol{f}$
  - Specify desired pixel-differences – $(\boldsymbol{g}^x, \boldsymbol{g}^y)$
  - Specify desired pixel-values – $\boldsymbol{d}$
  - Specify constraints weights – $(\boldsymbol{w}^x, \boldsymbol{w}^y, \boldsymbol{w}^d)$

Energy function

$$\min_{f} \quad \boldsymbol{w}^x(\boldsymbol{f}_x - \boldsymbol{g}^x)^2 + \boldsymbol{w}^y(\boldsymbol{f}_y - \boldsymbol{g}^y)^2 + \boldsymbol{w}^d(\boldsymbol{f} - \boldsymbol{d})^2$$

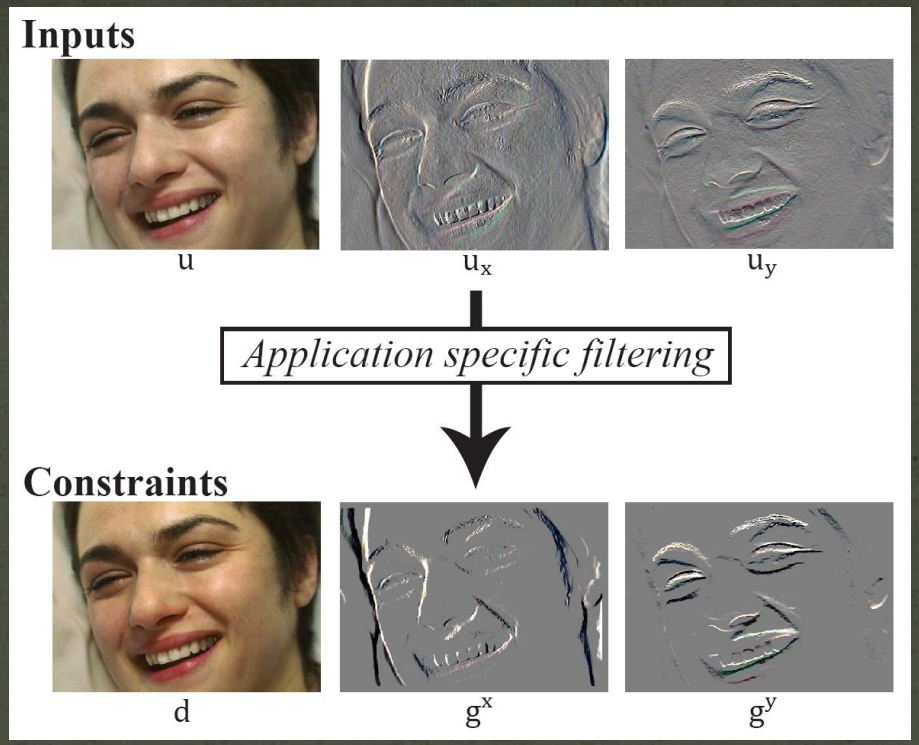
Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

GradientShop



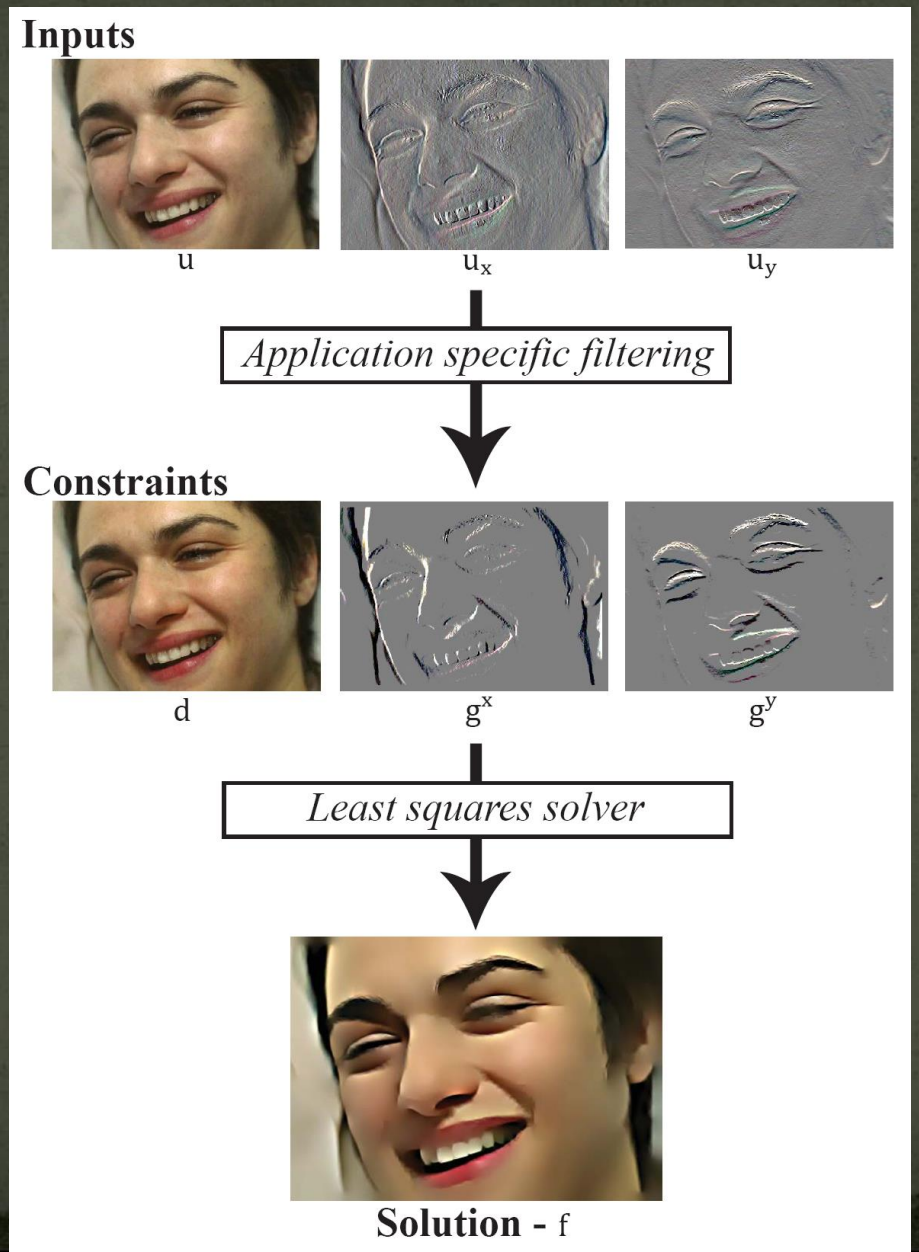
Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

GradientShop



Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

GradientShop



Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Applications

Sharpening

u

f

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Sharpening

Energy function

$$\min_{f} \quad w^x(f_x - g^x)^2 + w^y(f_y - g^y)^2 + w^d(f - d)^2$$

u

f

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Sharpening

Energy function

$$\min_{f} \quad w^x(f_x - g^x)^2 + w^y(f_y - g^y)^2 + w^d(f - d)^2$$

u

f

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Sharpening

Energy function

$$\min_{f} \quad w^x(f_x - g^x)^2 + w^y(f_y - g^y)^2 + w^d(f - d)^2$$

u

f

- Definition:
  - $g^x = c_1 * u_x$; $g^y = c_1 * u_y$

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Sharpening

Energy function

$$\min_{f} \quad w^x(f_x - g^x)^2 + w^y(f_y - g^y)^2 + w^d(f - d)^2$$

u

f

- Definition:
  - $g^x = c_1 * u_x$; $g^y = c_1 * u_y$
  - $d = u$

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Sharpening

u

sharpen

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Sharpening

u

sharpen

Problems:
- sharpens noise
- sharpens clutter

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Sharpening

u

sharpen

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Smart sharpening

smart sharpen

sharpen

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Smart sharpening

smart sharpen

- Definition:

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Smart sharpening

smart sharpen

- Definition:
  - **run edge detector**

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Smart sharpening

smart sharpen

e

- Definition:
  - **run edge detector**

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Smart sharpening

smart sharpen

e

- Definition:
  - **run edge detector**
  - $g^x = (1 + e(p)) * u_x$
  - $g^y = (1 + e(p)) * u_y$

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Smart sharpening

smart sharpen

e

- Definition:
  - **run edge detector**
  - $g^x = (1 + e(p)) * u_x$
  - $g^y = (1 + e(p)) * u_y$
  - $d = u$

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Smart sharpening

smart sharpen

e

- Definition:
  - **run edge detector**
  - $g^x = (1 + e(p)) * u_x$
  - $g^y = (1 + e(p)) * u_y$
  - $d = u$
  - $w^x(p) = 1$
    $w^y(p) = 1$

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Smart sharpening

smart sharpen (uniform weights)

e

- Definition:
  - **run edge detector**
  - $g^x = (1 + e(p)) * u_x$
  - $g^y = (1 + e(p)) * u_y$
  - $d = u$
  - $w^x(p) = 1$
    $w^y(p) = 1$

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Smart sharpening

u

e

- Definition:
  - **run edge detector**
  - $g^x = (1 + e(p)) * u_x$
  - $g^y = (1 + e(p)) * u_y$
  - $d = u$
  - $w^x(p) = 1$
    $w^y(p) = 1$

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Smart sharpening

smart sharpen
(uniform weights)

e

- Definition:
  - **run edge detector**
  - $g^x = (1 + e(p)) * u_x$
  - $g^y = (1 + e(p)) * u_y$
  - $d = u$
  - $w^x(p) = 1$
    $w^y(p) = 1$

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

Smart sharpening

smart sharpen (robust weights)

e

- Definition:
  - **run edge detector**
  - $g^x = (1 + e(p)) * u_x$
  - $g^y = (1 + e(p)) * u_y$
  - $d = u$
  - **robust weighting:**

$$w^x(p) = \frac{1}{(1 + |u_x - g^x|^b)}$$

Slide: GradientShop: A Gradient-Domain Optimization Framework for Image & Video Filtering

References

- Blending
  Szeliski, Section 3.5.5 & 9.3.4