Image Blending and Compositing

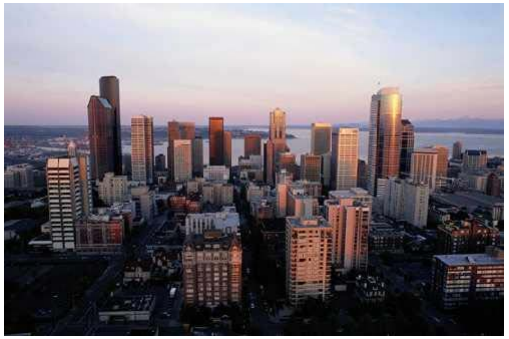
CS194: Image Manipulation & Computational Photography

Alexei Efros, UC Berkeley, Fall 2017

Image Compositing

Compositing Procedure

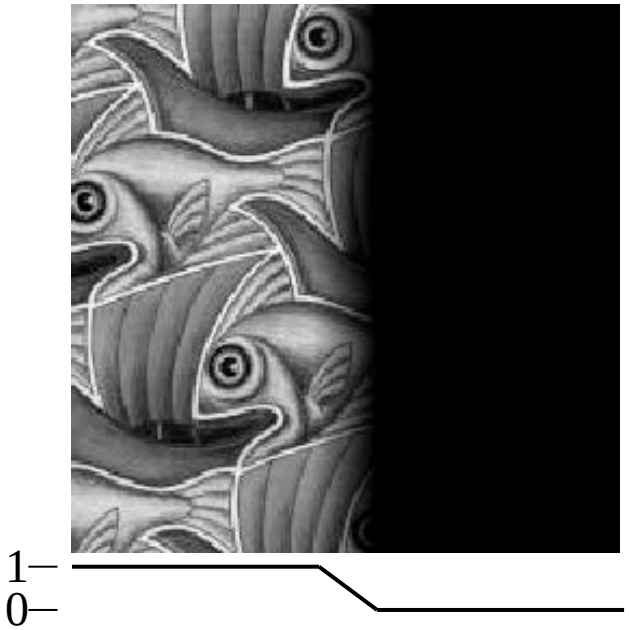
1. Extract Sprites (e.g using *Intelligent Scissors* in Photoshop)



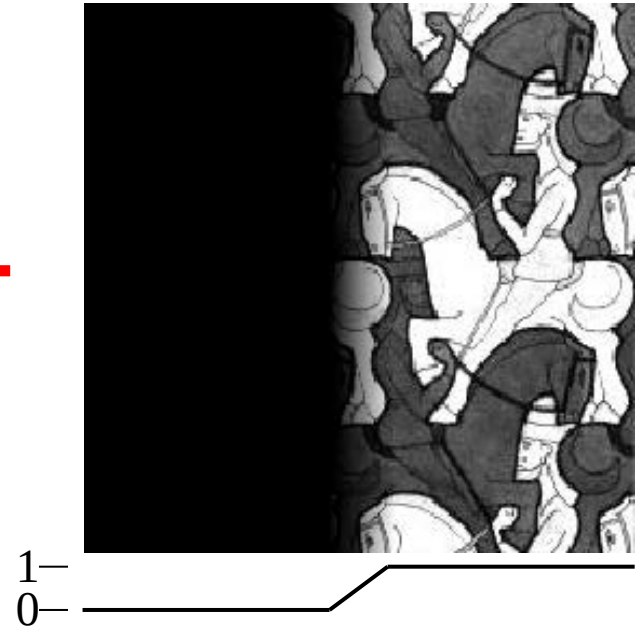
2. Blend them into the composite (in the right order)

Composite by
David Dewey

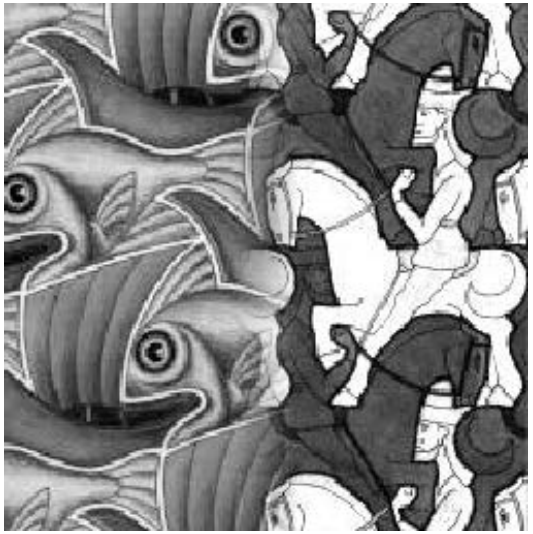
Alpha Blending / Feathering



+

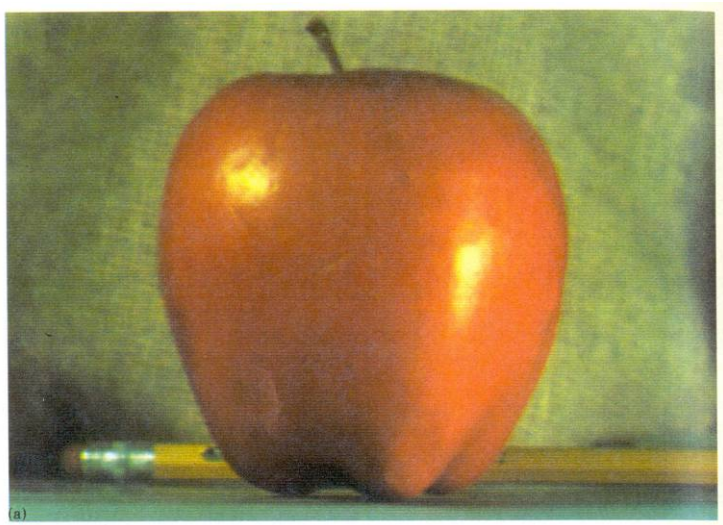


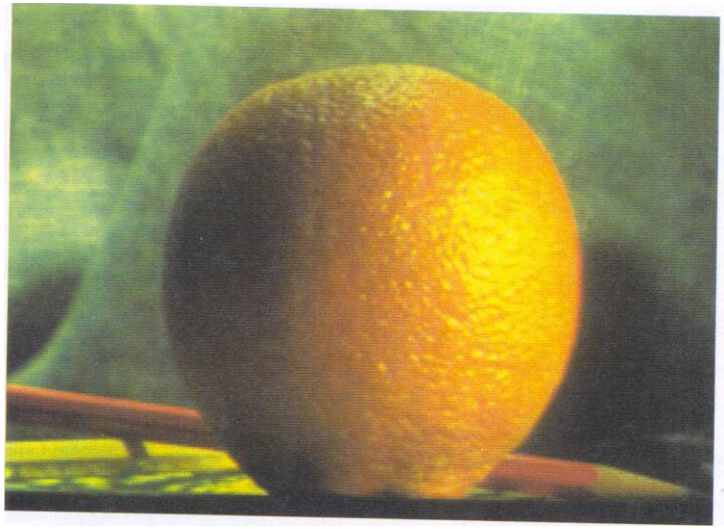
=

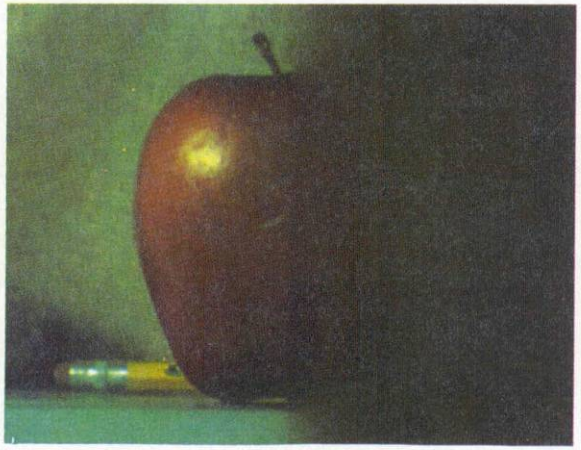


$$I_{blend} = \alpha I_{left} + (1-\alpha) I_{right}$$

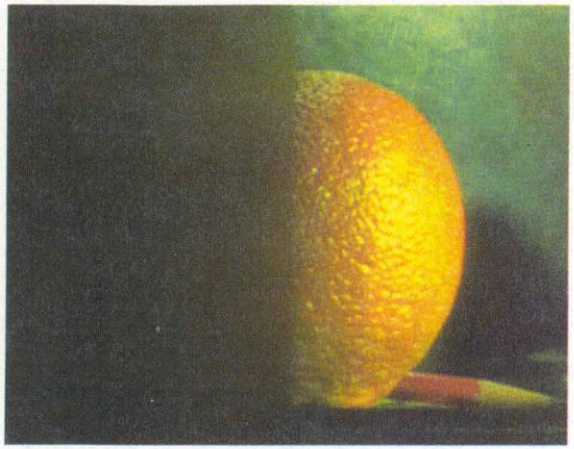
Pyramid Blending



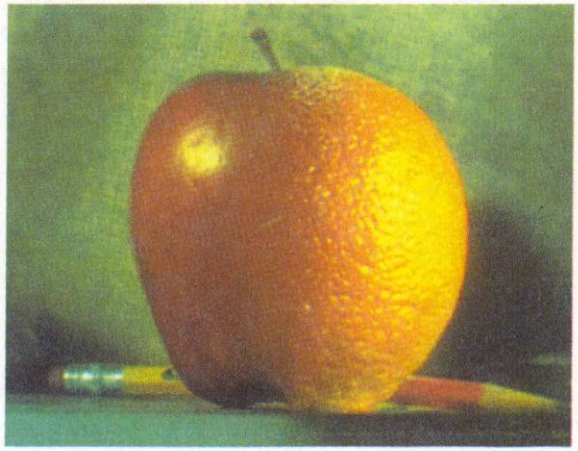




(d)



(h)



(l)

Gradient Domain vs. Frequency Domain

In Pyramid Blending, we decomposed our images into several frequency bands, and transferred them separately

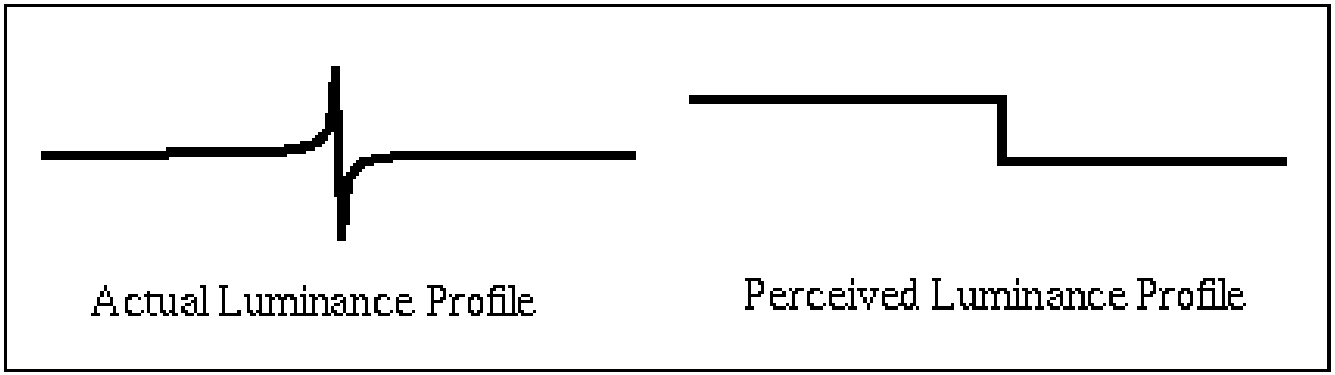
- But boundaries appear across multiple bands

But what about representation based on derivatives (gradients) of the image?:

- Represents local change (across all frequences)
- No need for low-res image
  - captures everything (up to a constant)
- Blending/Editing in Gradient Domain:
  - Differentiate
  - Copy / Blend / edit / whatever
  - Reintegrate

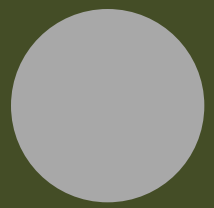
Gradients vs. Pixels

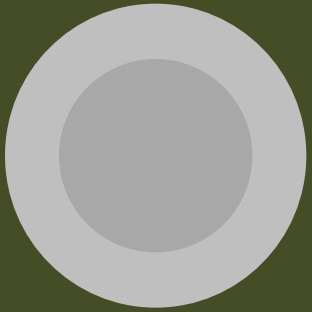
Craik-O'Brien Cornsweet Effect

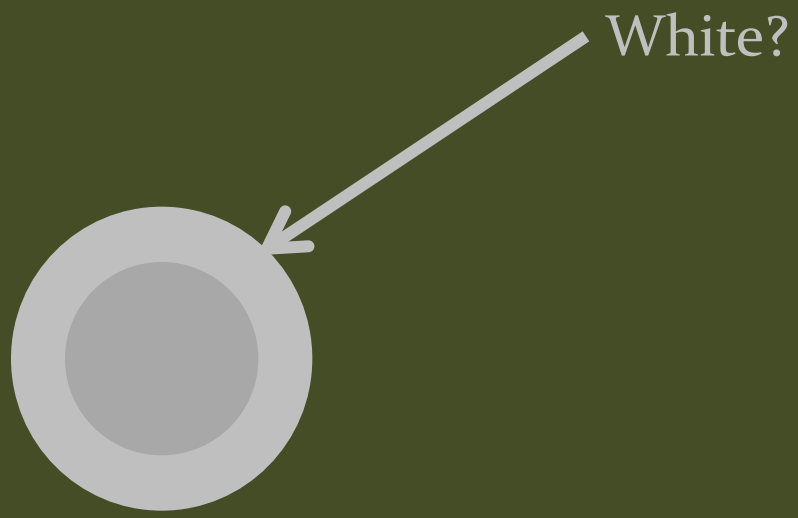


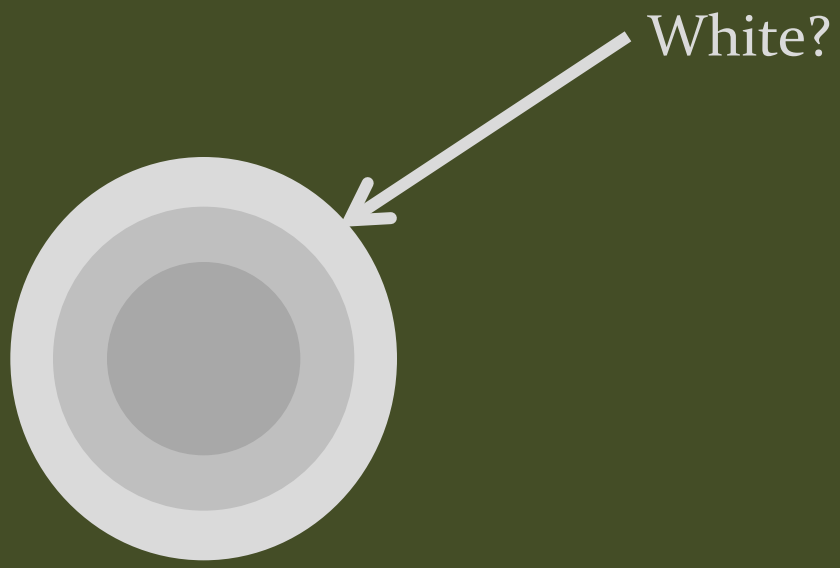
Gilchrist Illusion

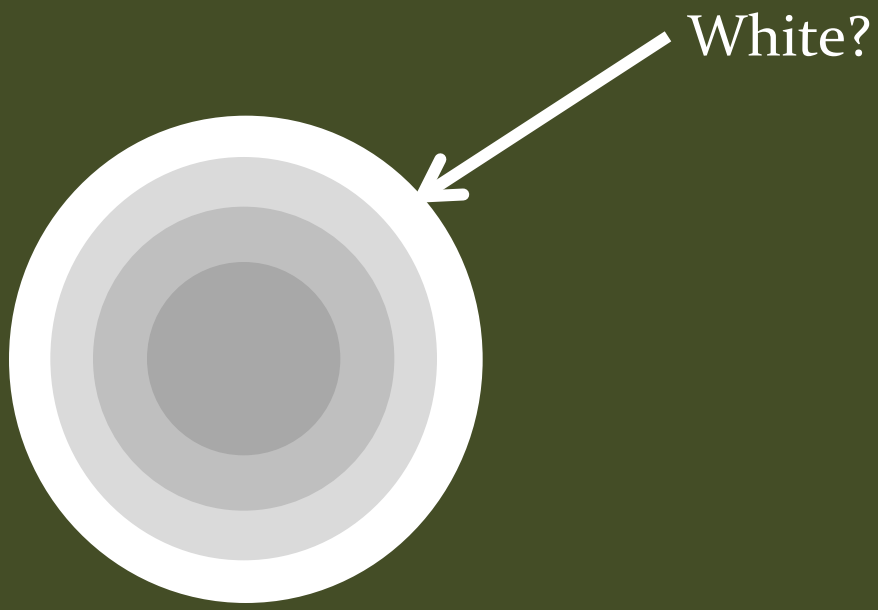
(c.f. Exploratorium)

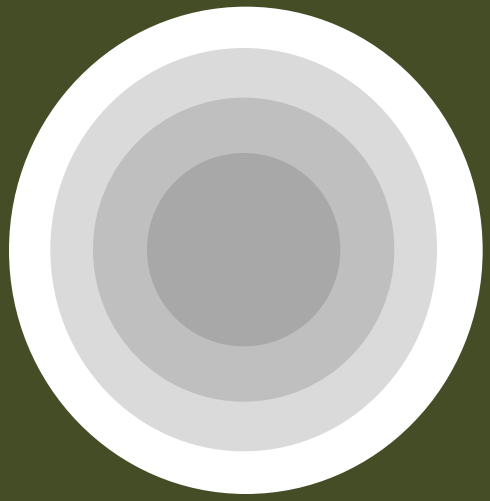




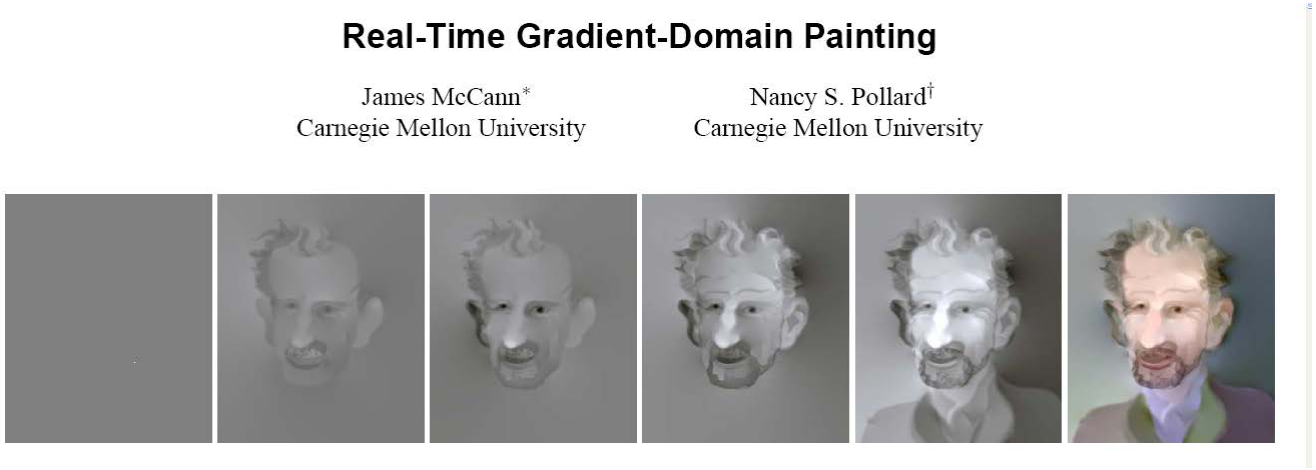

White?


White?


White?



Drawing in Gradient Domain



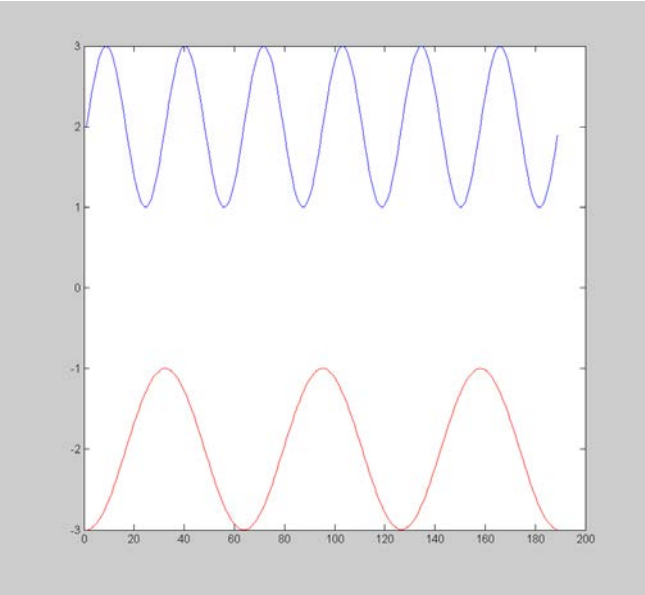
James McCann & Nancy Pollard
Real-Time Gradient-Domain Painting,
SIGGRAPH 2009

(paper came out of this class!)

http://www.youtube.com/watch?v=RvhkAfrA0-w&feature=youtu.be

Gradient Domain blending (1D)

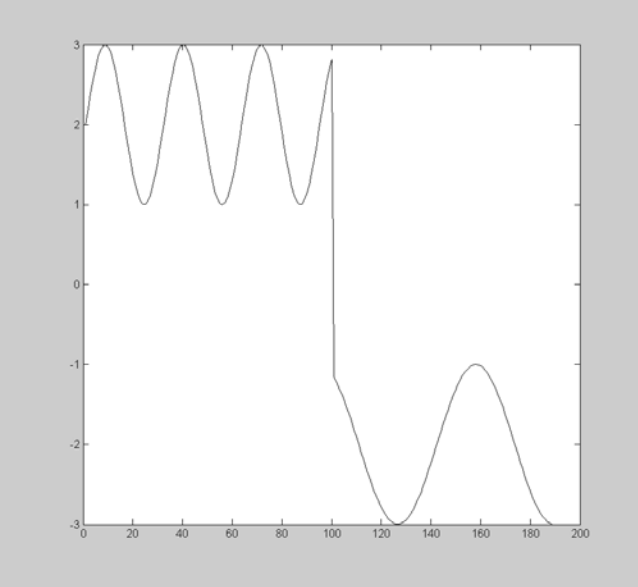
Two signals

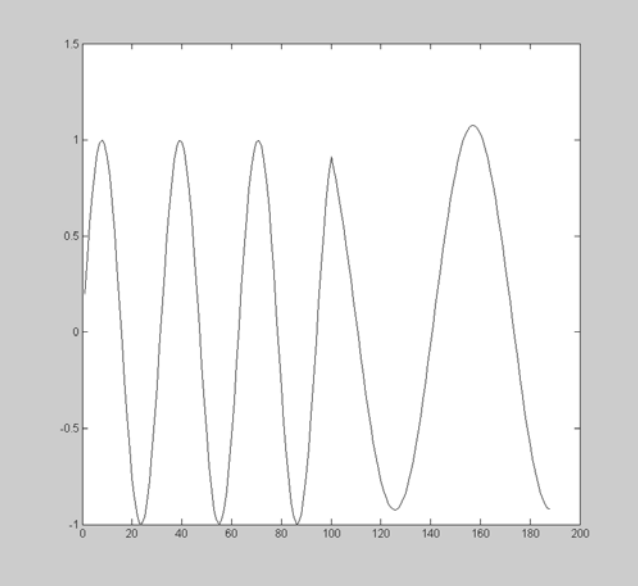


bright

dark

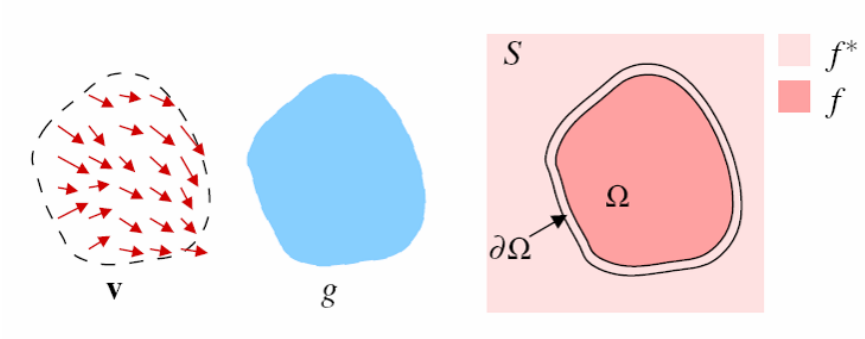
Regular blending





Blending derivatives

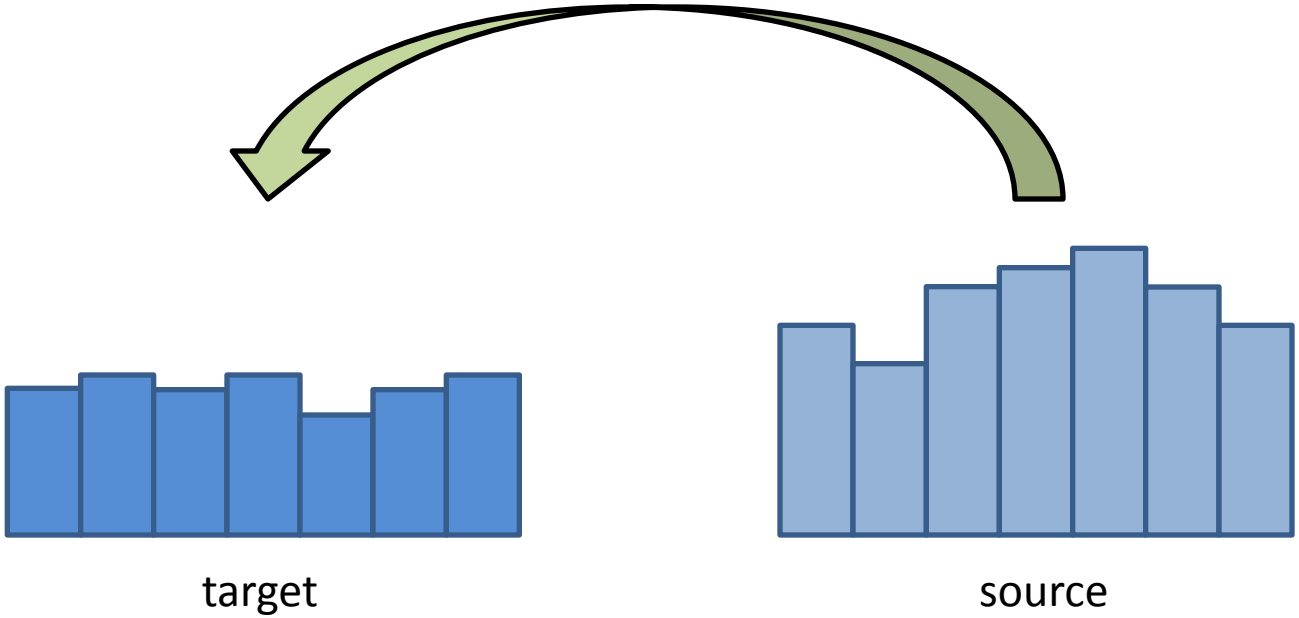
Gradient Domain Blending (2D)

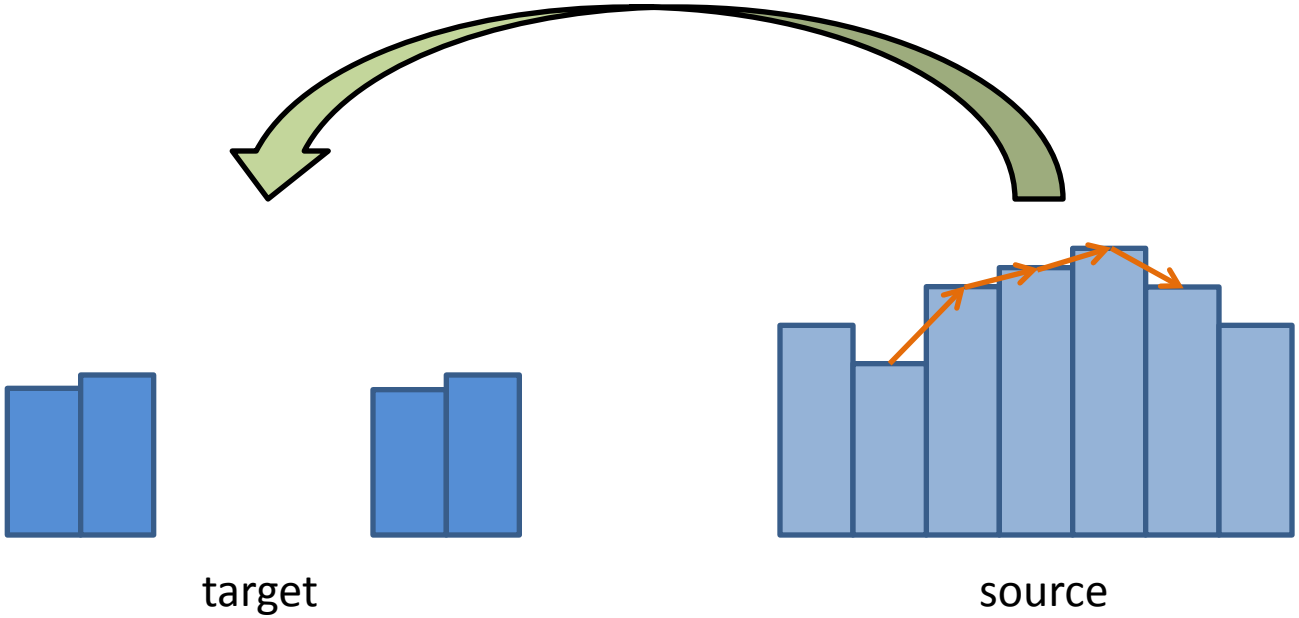


Trickier in 2D:

- Take partial derivatives dx and dy (the gradient field)
- Fiddle around with them (copy, blend, smooth, feather, etc)
- Reintegrate
  - But now integral(dx) might not equal integral(dy)
- Find the most agreeable solution
  - Equivalent to solving Poisson equation
  - Can be done using least-squares (\ in Matlab)

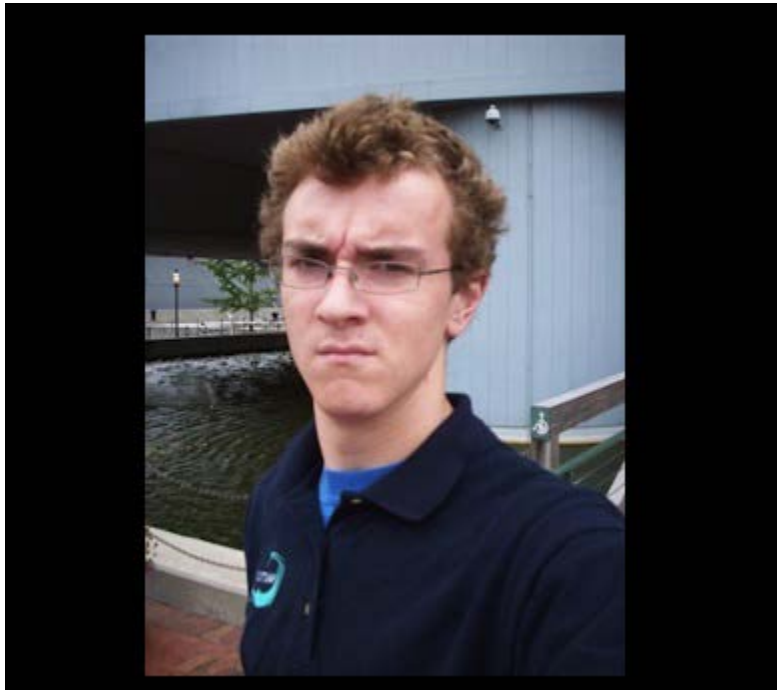
Gradient hole-filling (1D)





It is impossible to faithfully preserve the gradients

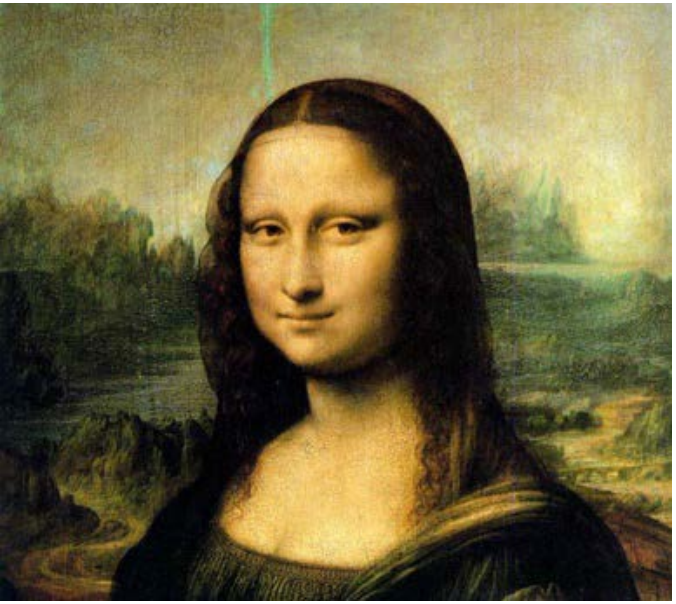
Example



Gradient Visualization

Source: Evan Wallace

+



Specify object region

Source: Evan Wallace

Poisson Blending Algorithm

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

Treat pixels as variables to be solved

- Minimize squared difference between gradients of foreground region and gradients of target region
- Keep background pixels constant

$$\mathbf{v} = \underset{\mathbf{v}}{\operatorname{argmin}} \sum_{i \in S, j \in N_i \cap S} ((v_i - v_j) - (s_i - s_j))^2 + \sum_{i \in S, j \in N_i \cap \neg S} ((v_i - t_j) - (s_i - s_j))^2$$

Perez et al. 2003

Examples

Gradient domain processing

$$\mathbf{v} = \underset{\mathbf{v}}{\operatorname{argmin}} \sum_{i \in S, j \in N_i \cap S} ((v_i - v_j) - (s_i - s_j))^2 + \sum_{i \in S, j \in N_i \cap \neg S} ((v_i - t_j) - (s_i - s_j))^2$$

source image

1 **20**	5 **20**	9 **20**	13 **20**
2 **20**	6 **80**	10 **20**	14 **20**
3 **20**	7 **20**	11 **80**	15 **20**
4 **20**	8 **20**	12 **20**	16 **20**

background image

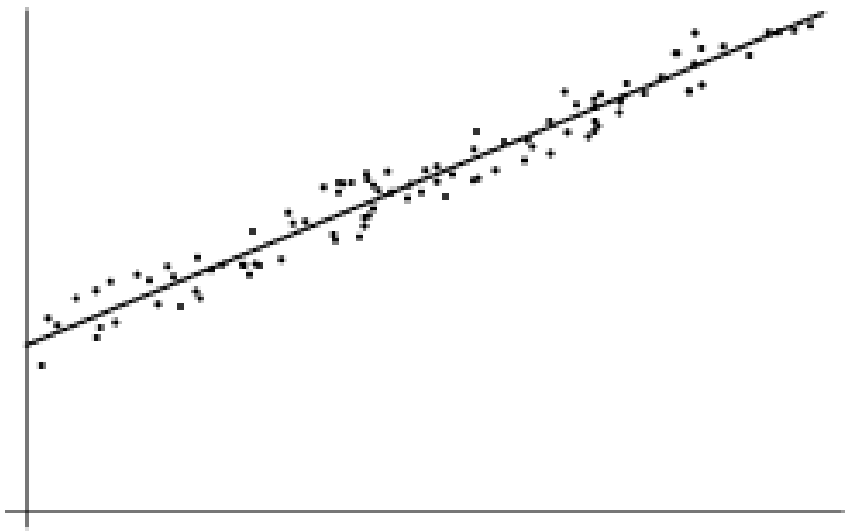
1 **10**	5 **10**	9 **10**	13 **10**
2 **10**	6 **10**	10 **10**	14 **10**
3 **10**	7 **10**	11 **10**	15 **10**
4 **10**	8 **10**	12 **10**	16 **10**

target image

1 **10**	5 **10**	9 **10**	13 **10**
2 **10**	6 $\mathbf{v_1}$	10 $\mathbf{v_3}$	14 **10**
3 **10**	7 $\mathbf{v_2}$	11 $\mathbf{v_4}$	15 **10**
4 **10**	8 **10**	12 **10**	16 **10**

Gradient-domain editing

Creation of image = least squares problem in terms of: 1) pixel intensities; 2) differences of pixel intensities



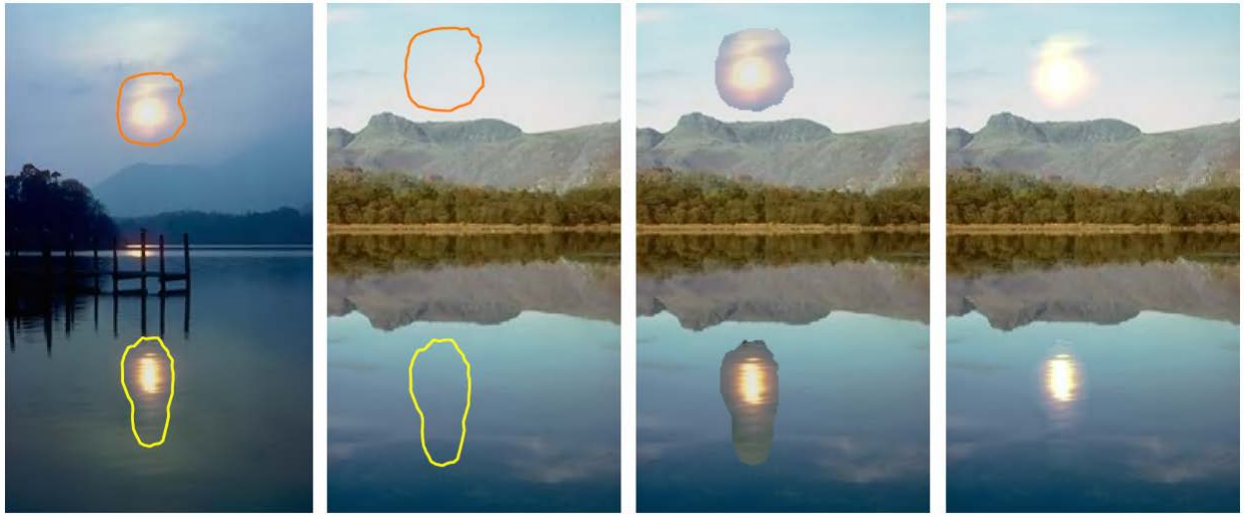
Least Squares Line Fit in 2 Dimensions

$$\hat{\mathbf{v}} = \arg\min_{\mathbf{v}} \sum_i \left(\mathbf{a}_i^T \mathbf{v} - b_i\right)^2$$

$$\hat{\mathbf{v}} = \arg\min_{\mathbf{v}} \left(\mathbf{A}\mathbf{v} - \mathbf{b}\right)^2$$

Use Matlab least-squares solvers for numerically stable solution with sparse A

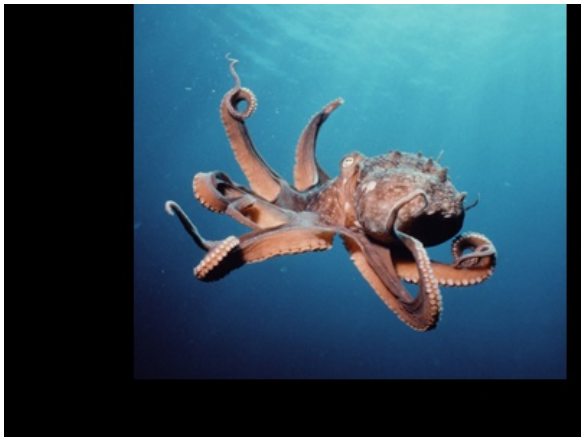
Perez et al., 2003



sources destinations cloning seamless cloning

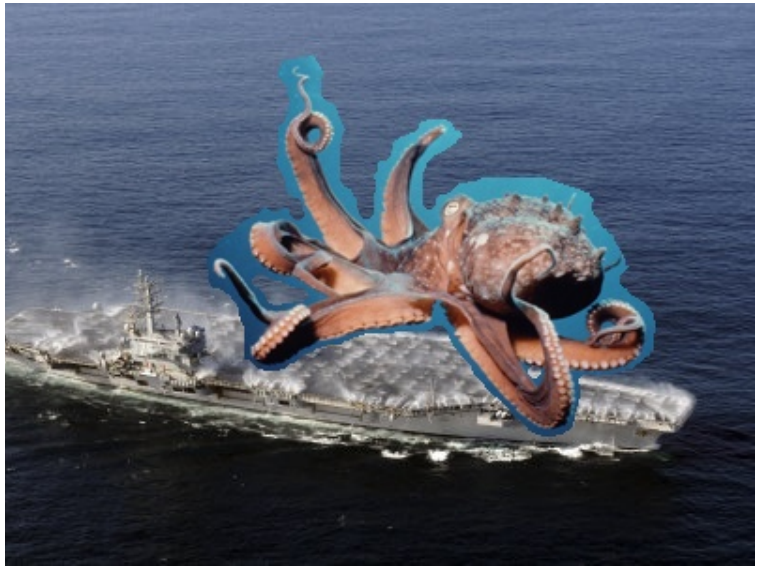
sources/destinations cloning seamless cloning

target



source

mask

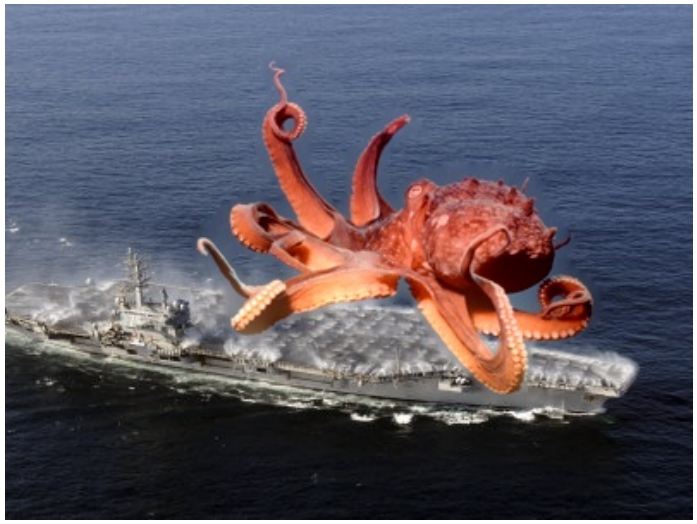


no blending

gradient domain blending

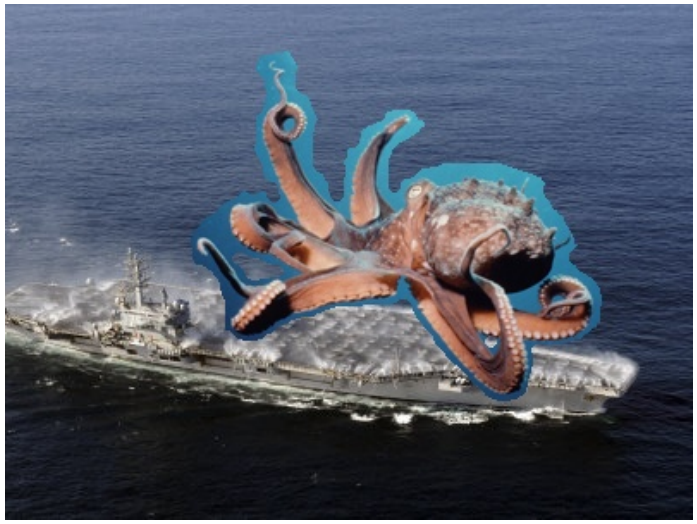
Slide by Mr. Hays

What’s the difference?



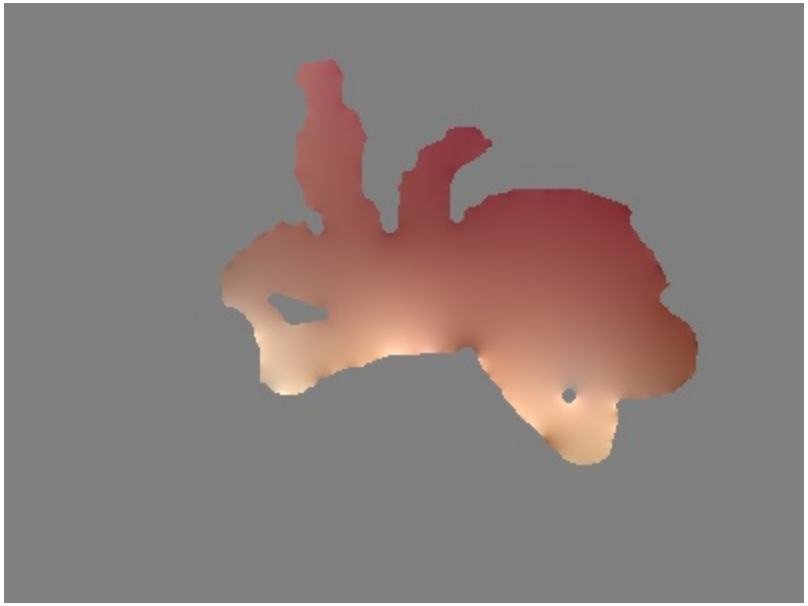
gradient domain blending

-



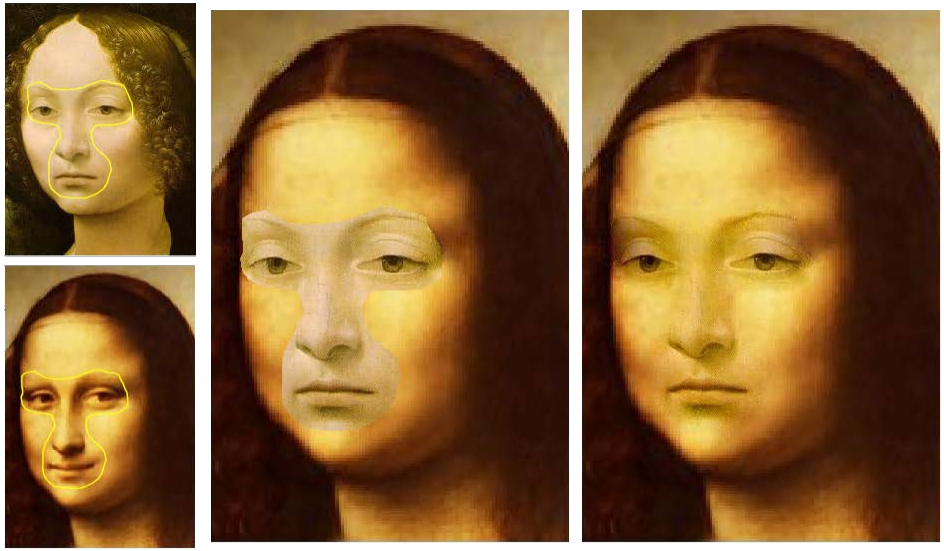
no blending

=



Slide by Mr. Hays

Perez et al, 2003



source/destination cloning seamless cloning

editing

Limitations:

- Can’t do contrast reversal (gray on black -> gray on white)
- Colored backgrounds “bleed through”
- Images need to be very well aligned

Gradient Domain as Image Representation

See GradientShop paper as good example:

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

Pravin Bhat[1] C. Lawrence Zitnick[2] Michael Cohen[1,2] Brian Curless[1]
[1]University of Washington [2]Microsoft Research

http://www.gradientshop.com/

Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images

Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features

Motivation for gradient-domain filtering?

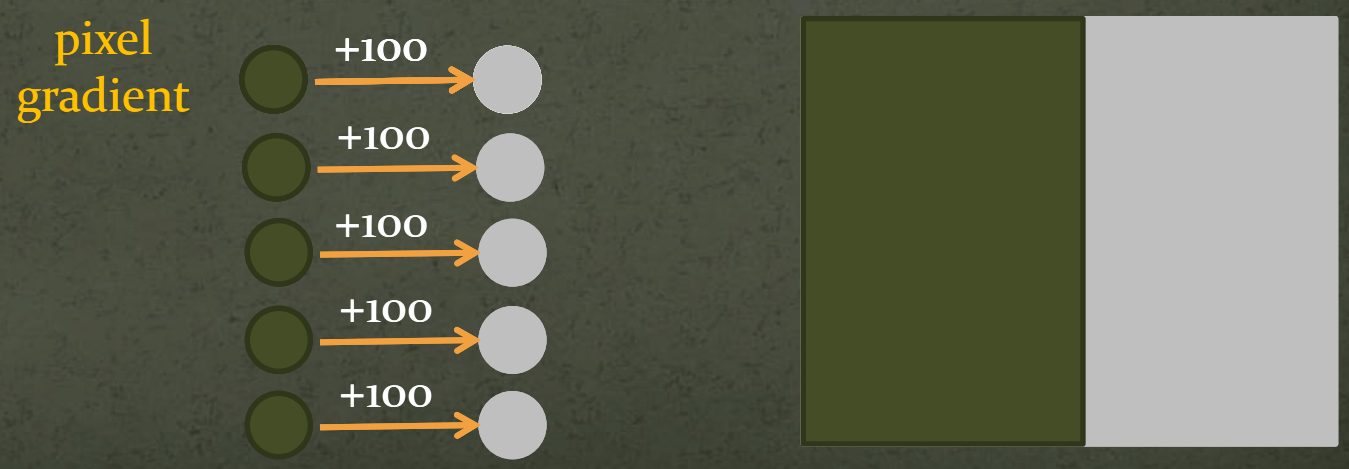
- Can be used to exert high-level control over images
  - gradients – low level image-features

Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features

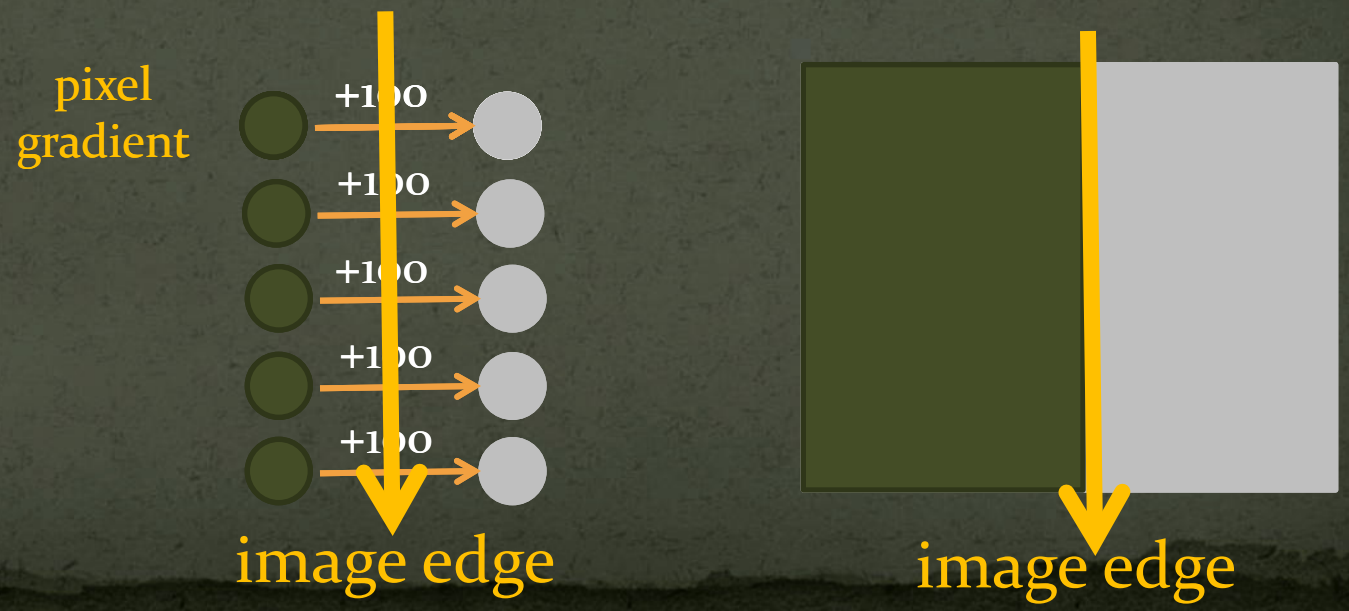
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features



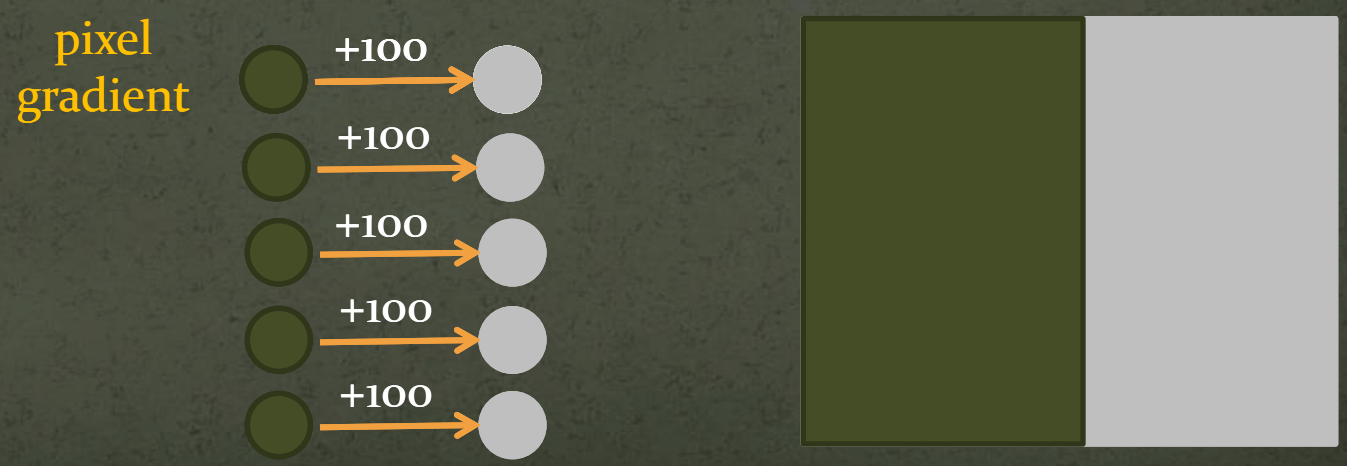
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features



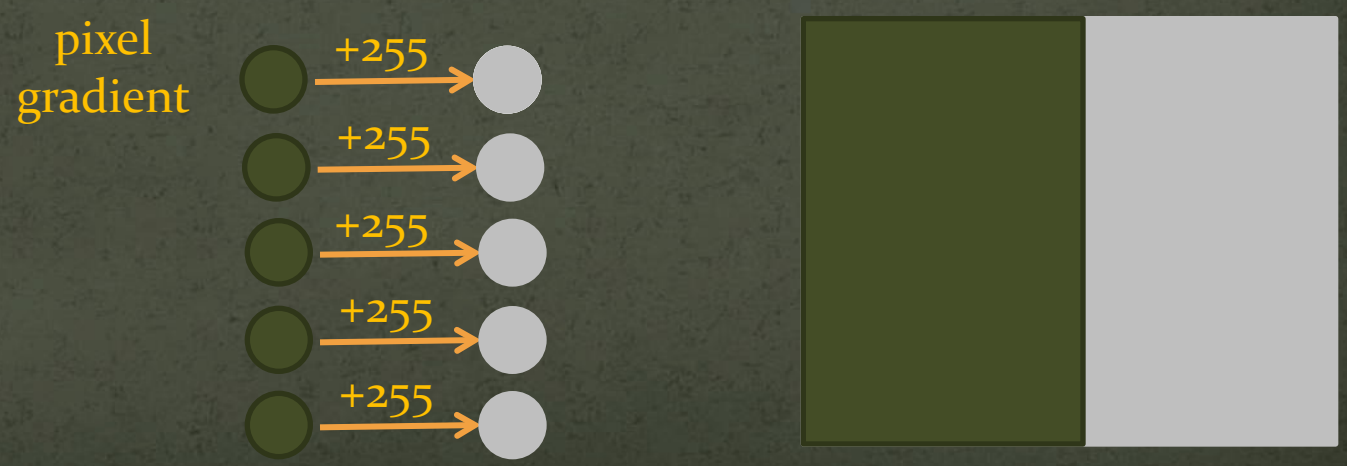
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features
  - manipulate local gradients to manipulate global image interpretation



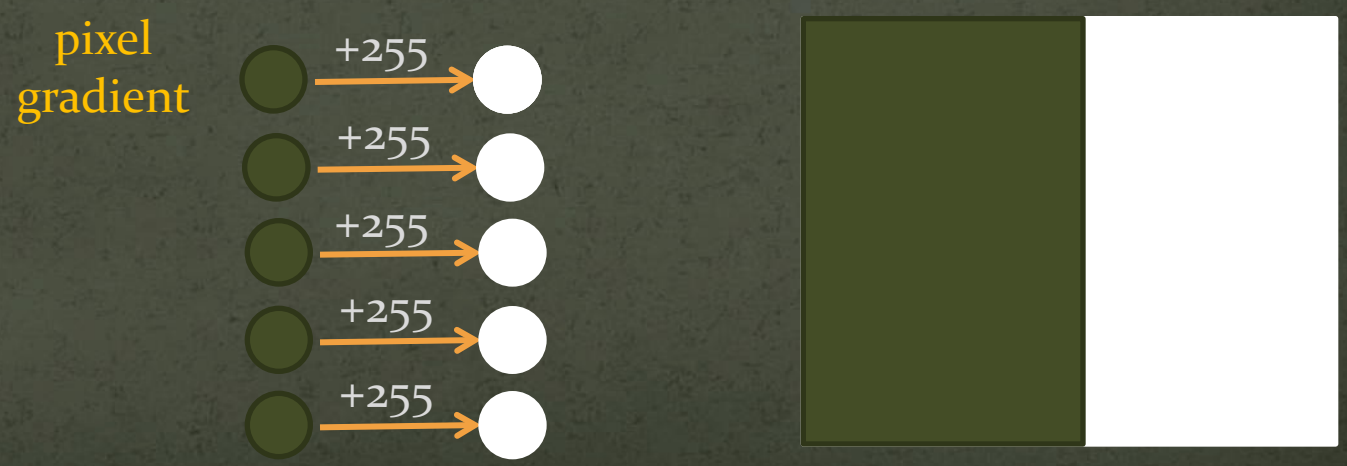
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features
  - manipulate local gradients to manipulate global image interpretation



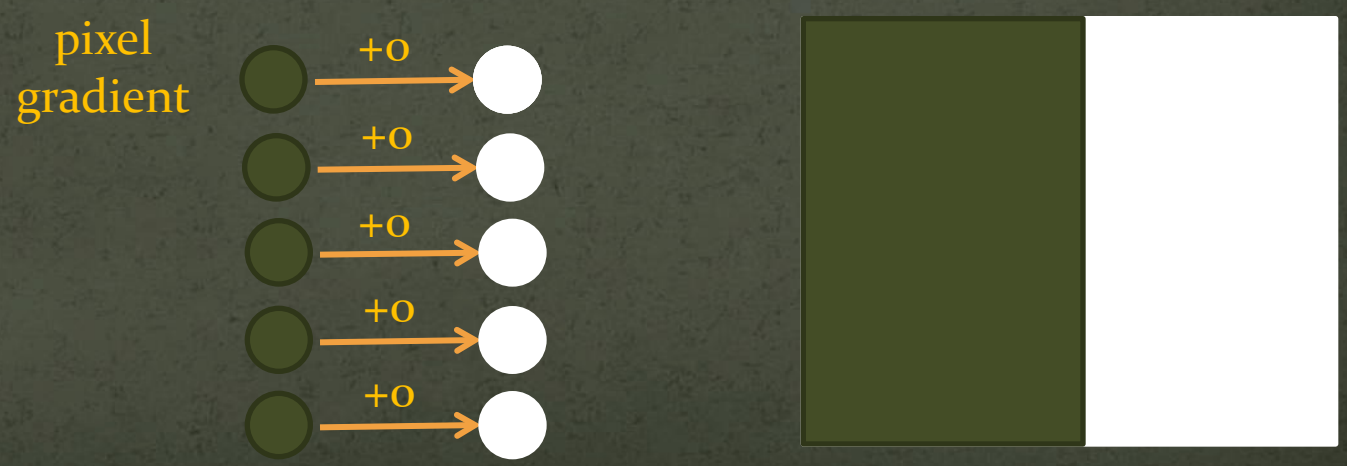
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features
  - manipulate local gradients to manipulate global image interpretation



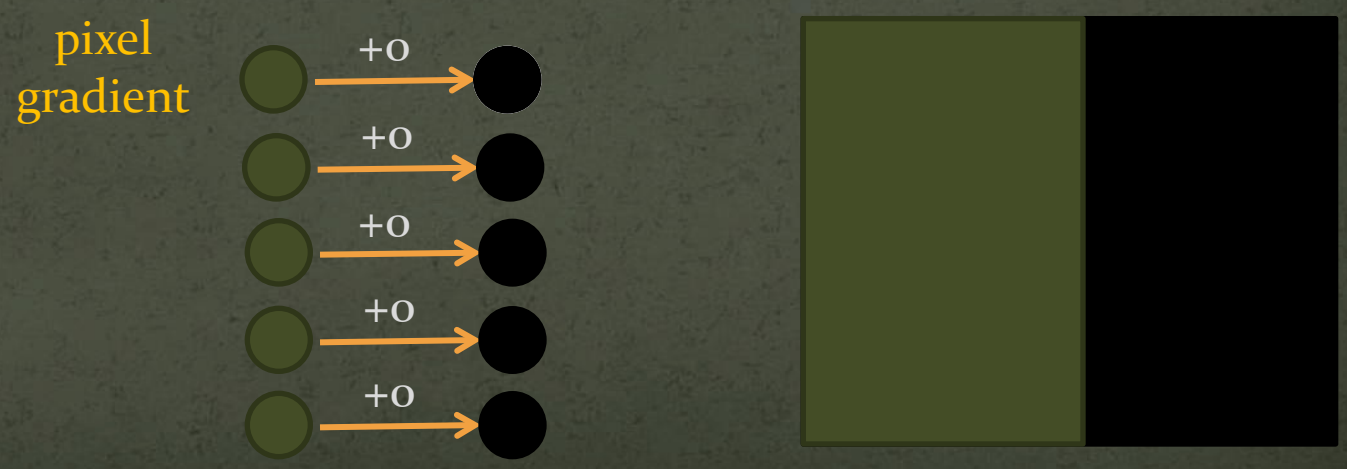
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features
  - manipulate local gradients to manipulate global image interpretation



Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features
  - manipulate local gradients to manipulate global image interpretation



Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images

GradientShop

- Optimization framework

Pravin Bhat et al

GradientShop

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

GradientShop

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

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

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

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

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

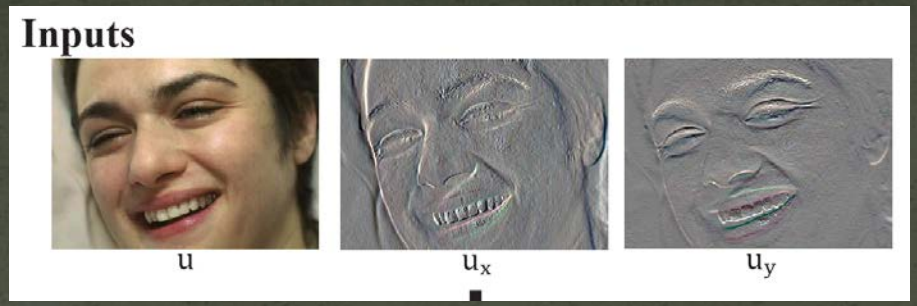
GradientShop

- 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 – $(\boldsymbol{w}^x, \boldsymbol{w}^y, \boldsymbol{w}^d)$

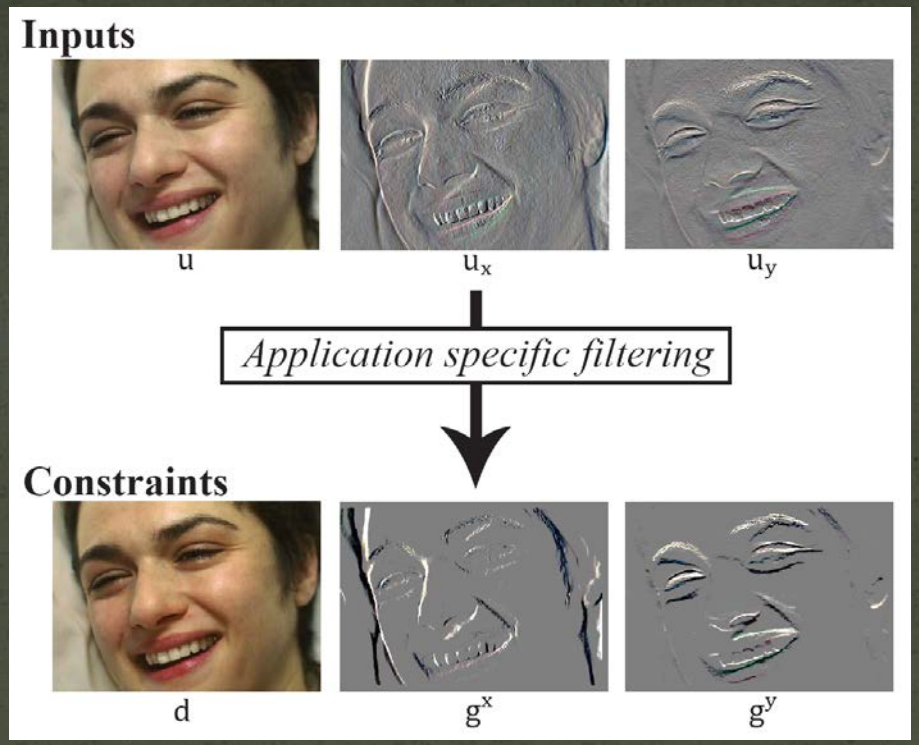
Energy function

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

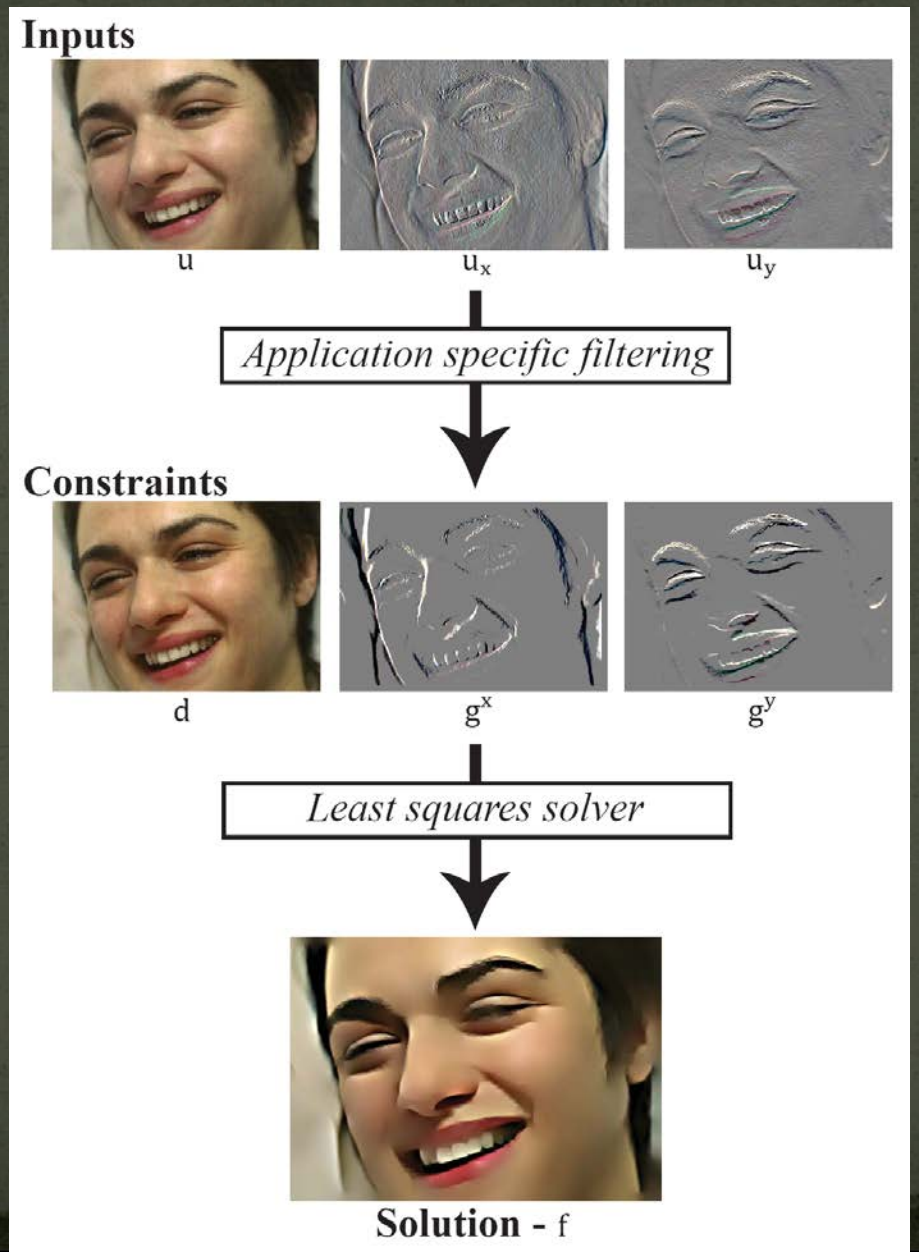
GradientShop



GradientShop



GradientShop



Pseudo image relighting

- change scene illumination in post-production
- example

input

Pseudo image relighting

- change scene illumination in post-production
- example

manual relight

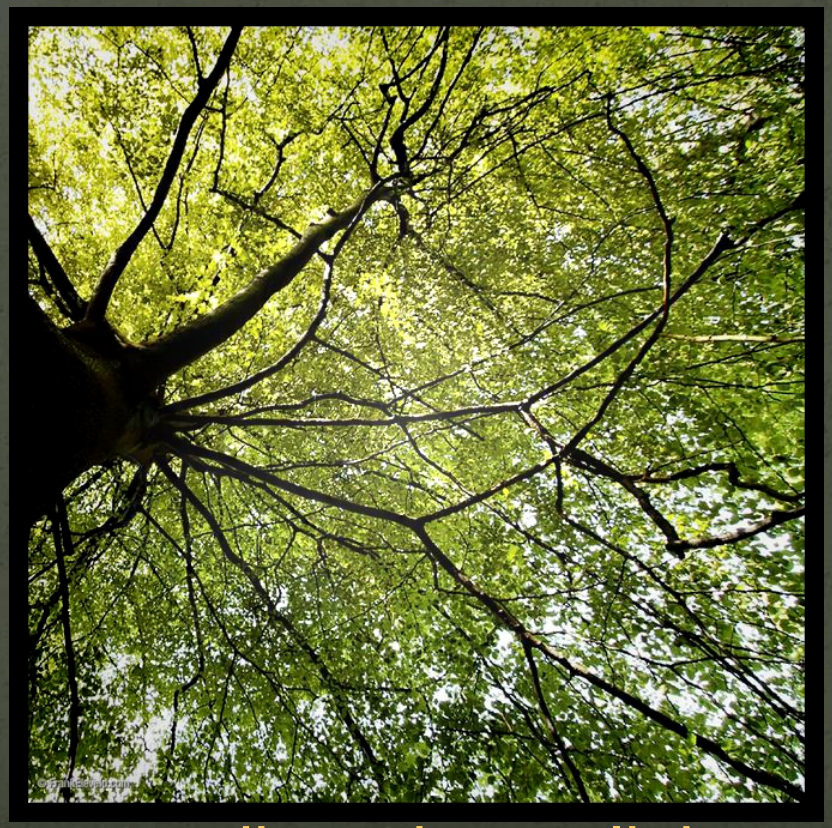
Pseudo image relighting

- change scene illumination in post-production
- example

input

Pseudo image relighting

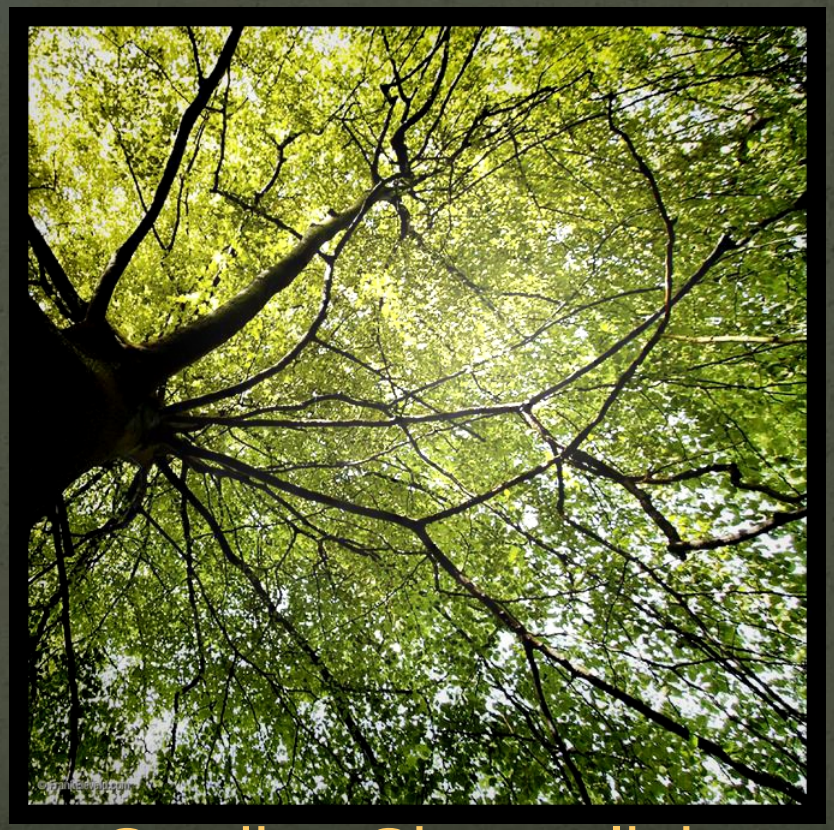
- change scene illumination in post-production
- example



GradientShop relight

Pseudo image relighting

- change scene illumination in post-production
- example



GradientShop relight

Pseudo image relighting

- change scene illumination in post-production
- example

GradientShop relight

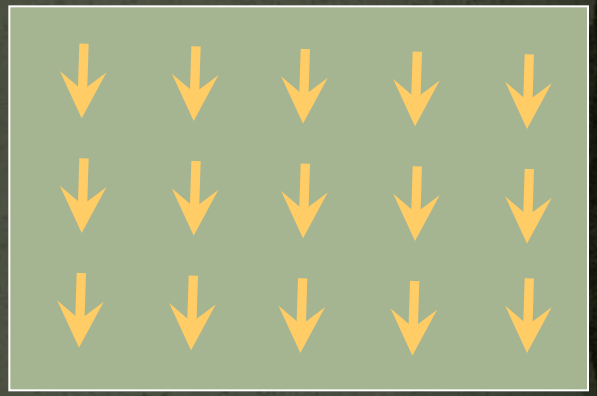
Pseudo image relighting

- change scene illumination in post-production
- example

GradientShop relight

Pseudo image relighting

u



o

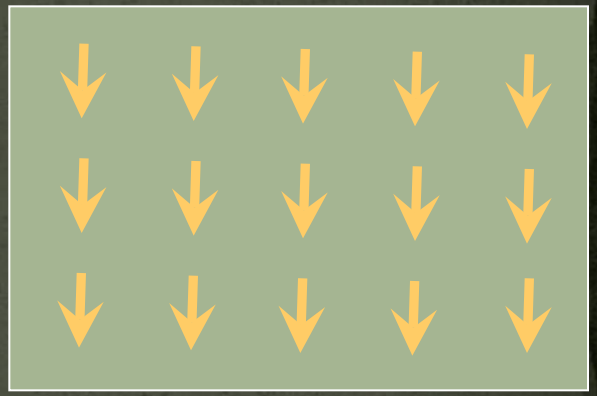
f

Pseudo image relighting

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



o

f

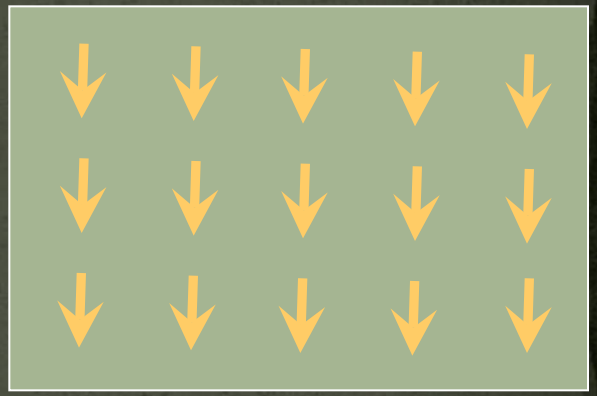
Pseudo image relighting

Energy function

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

- Definition:
  - $d = u$

u



o

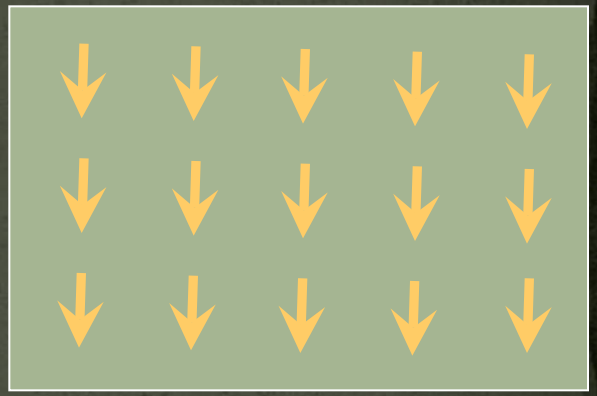
f

Pseudo image relighting

Energy function

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

u



o

- Definition:
  - $d = u$
  - $g^x(p) = u_x(p) * (1 + a(p))$
  - $a(p) = \max(0, -\nabla u(p).o(p))$

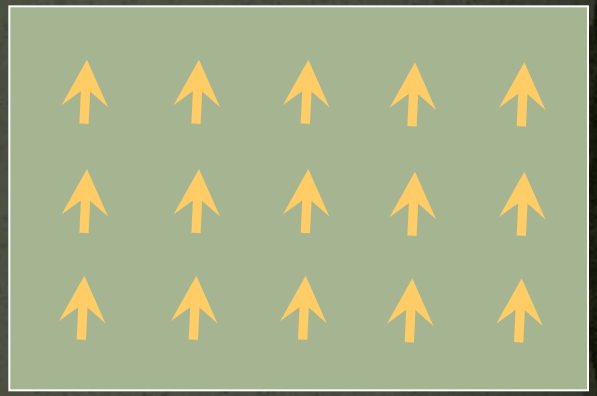
f

Pseudo image relighting

Energy function

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

u



o

- Definition:
  - $d = u$
  - $g^x(p) = u_x(p) * (1 + a(p))$
  - $a(p) = \max(0, -\nabla u(p).o(p))$

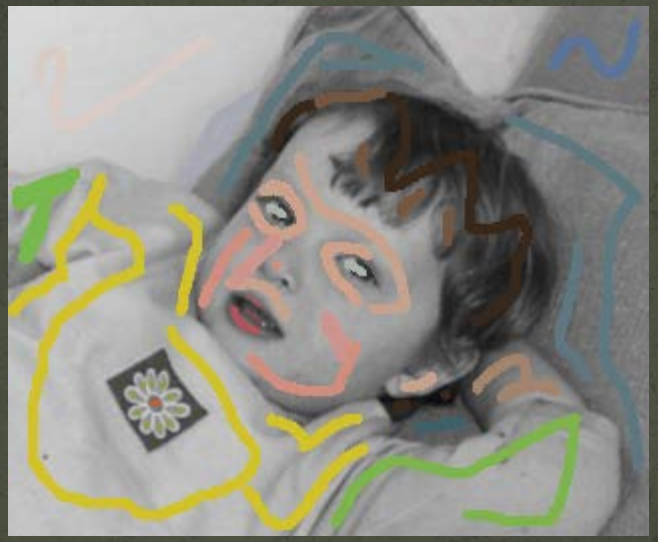
f

Sparse data interpolation

- Interpolate scattered data over images/video

Sparse data interpolation

- Interpolate scattered data over images/video
- Example app: Colorization*



input

output

*Levin et al. – SIGRAPH 2004

Sparse data interpolation

u

user data

f

Sparse data interpolation

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

user data

f

Sparse data interpolation

Energy function

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

- Definition:
  - ***d* = user_data**

u

user data

f

Sparse data interpolation

Energy function

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

- Definition:
  - d = **user_data**
  - **if user_data(p) defined**
    $w^d(p) = 1$
    else
    $w^d(p) = 0$

u

user data

f

Sparse data interpolation

Energy function

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

- Definition:
  - d = user_data
  - if user_data(p) defined
    $w^d(p) = 1$
    else
    $w^d(p) = 0$
  - $g^x(p) = 0$; $g^y(p) = 0$

u

user data

f

Sparse data interpolation

Energy function

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

u

user data

- Definition:
  - d = user_data
  - if user_data(p) defined
    $w^d(p) = 1$
    else
    $w^d(p) = 0$
  - $g^x(p) = 0$; $g^y(p) = 0$
  - $w^x(p) = 1/(1 + c*|u_x(p)|)$
    $w^y(p) = 1/(1 + c*|u_y(p)|)$

f

Don’t blend, CUT!

Moving objects become ghosts

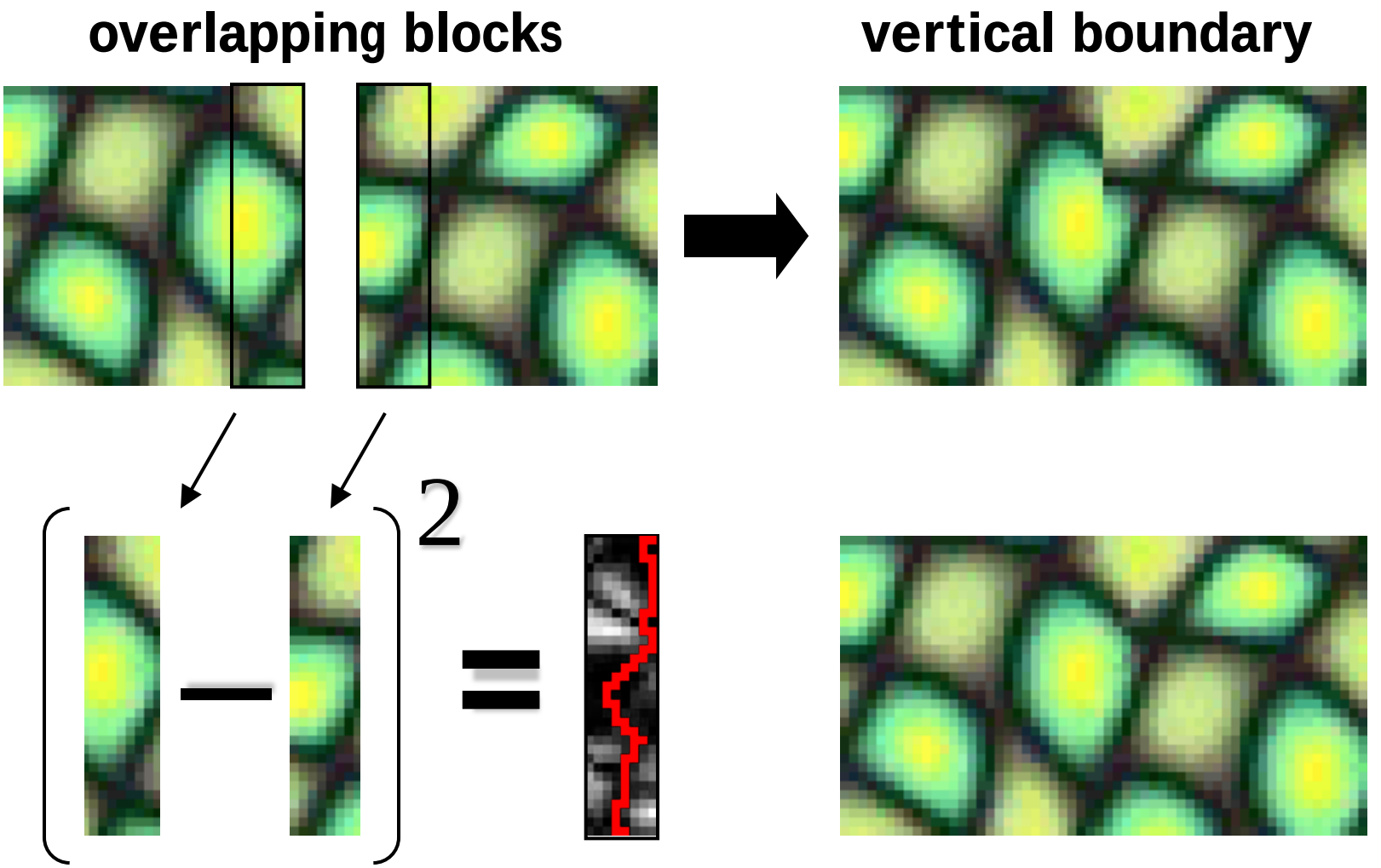
So far we only tried to blend between two images.
What about finding an optimal seam?

Davis, 1998

Segment the mosaic

- Single source image per segment
- Avoid artifacts along boundries
  - Dijkstra's algorithm

Minimal error boundary



Seam Carving

Seam Carving for Content-Aware Image Resizing

Shai Avidan
Mitsubishi Electric Research Labs

Ariel Shamir
The Interdisciplinary Center & MERL

http://www.youtube.com/watch?v=6NcIJXTlugc

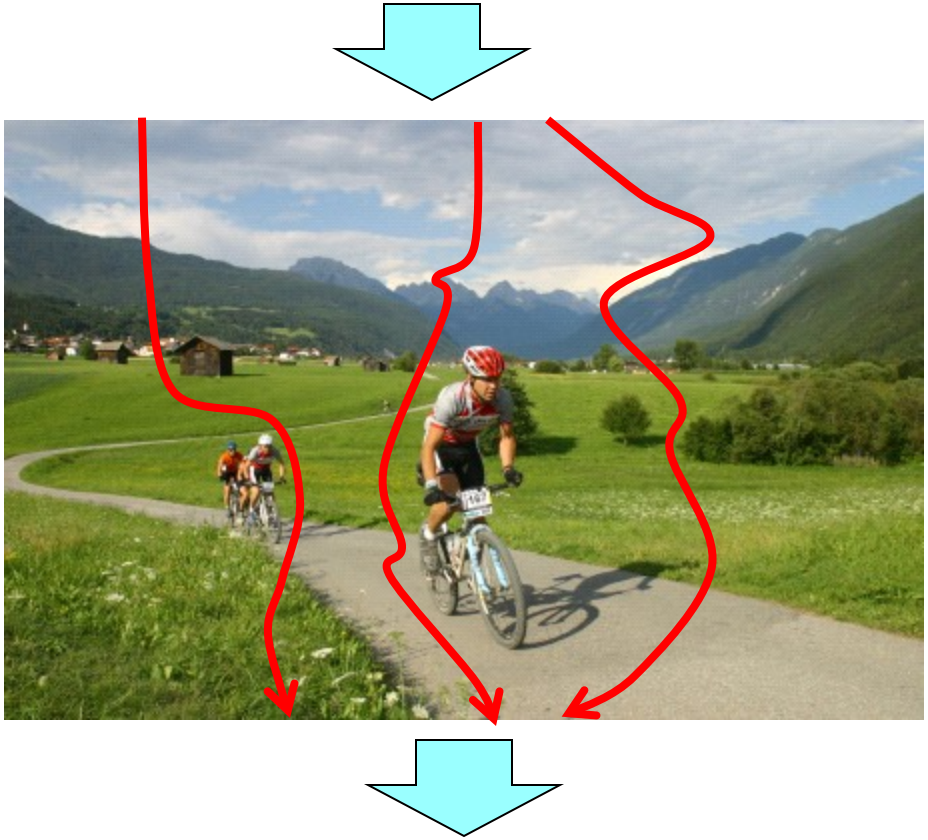
Seam Carving

- **Basic Idea: remove unimportant pixels from the image**
  - Unimportant = pixels with less “energy”

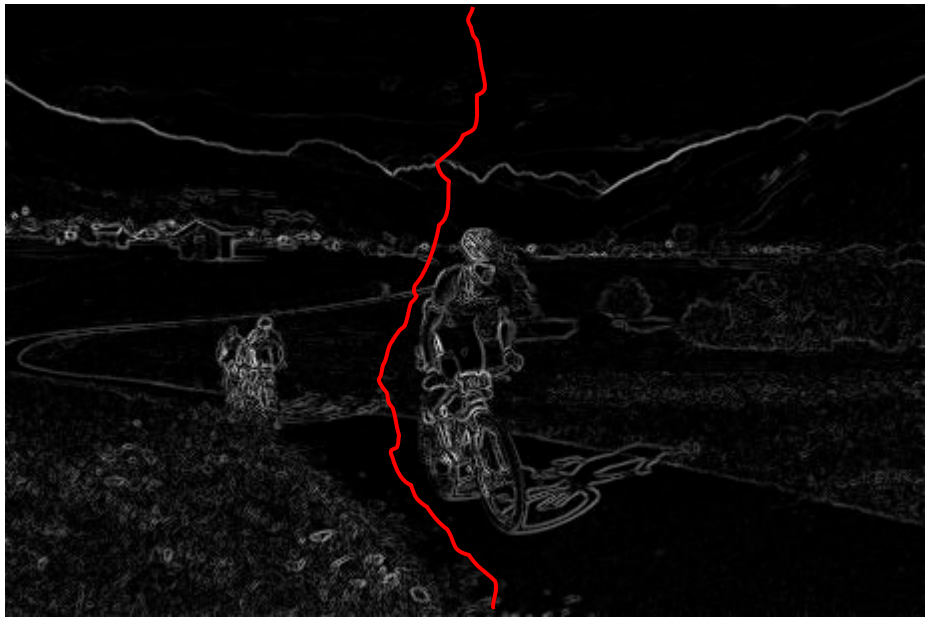
$$E_1(\mathbf{I}) = |\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}|.$$

- **Intuition for gradient-based energy:**
  - Preserve strong contours
  - Human vision more sensitive to edges – so try remove content from smoother areas
  - Simple, enough for producing some nice results
  - See their paper for more measures they have used

Finding the Seam?



The Optimal Seam



$$E(\mathbf{I}) = \left|\frac{\partial}{\partial x}\mathbf{I}\right| + \left|\frac{\partial}{\partial y}\mathbf{I}\right| \Rightarrow s^* = \arg\min_s E(s)$$

Dynamic Programming

- **Invariant property:**
  - M(i,j) = minimal cost of a seam going through (i,j) (satisfying the seam properties)

5	8	12	3
9	2	3	9
7	3	4	2
4	5	7	8

Dynamic Programming

$$\mathbf{M}(i, j) = E(i, j) + \min\left(\mathbf{M}(i-1, j-1), \mathbf{M}(i-1, j), \mathbf{M}(i-1, j+1)\right)$$

5	8	12	3
9	2+5	3	9
7	3	4	2
4	5	7	8

Dynamic Programming

$$\mathbf{M}(i, j) = E(i, j) + \min\left(\mathbf{M}(i-1, j-1), \mathbf{M}(i-1, j), \mathbf{M}(i-1, j+1)\right)$$

5	8	12	3
9	7	3+3	9
7	3	4	2
4	5	7	8

Dynamic Programming

$$\mathbf{M}(i,j) = E(i,j) + \min\left(\mathbf{M}(i-1,j-1), \mathbf{M}(i-1,j), \mathbf{M}(i-1,j+1)\right)$$

5	8	12	3
9	7	6	12
14	9	10	8
14	14	15	8+8

Searching for Minimum

- **Backtrack (can store choices along the path, but do not have to)**

5	8	12	3
9	7	6	12
14	9	10	8
14	14	15	16

Backtracking the Seam

5	8	12	3
9	7	6	12
14	9	10	8
14	14	15	16

Backtracking the Seam

5	8	12	3
9	7	6	12
14	9	10	8
14	14	15	16

Backtracking the Seam

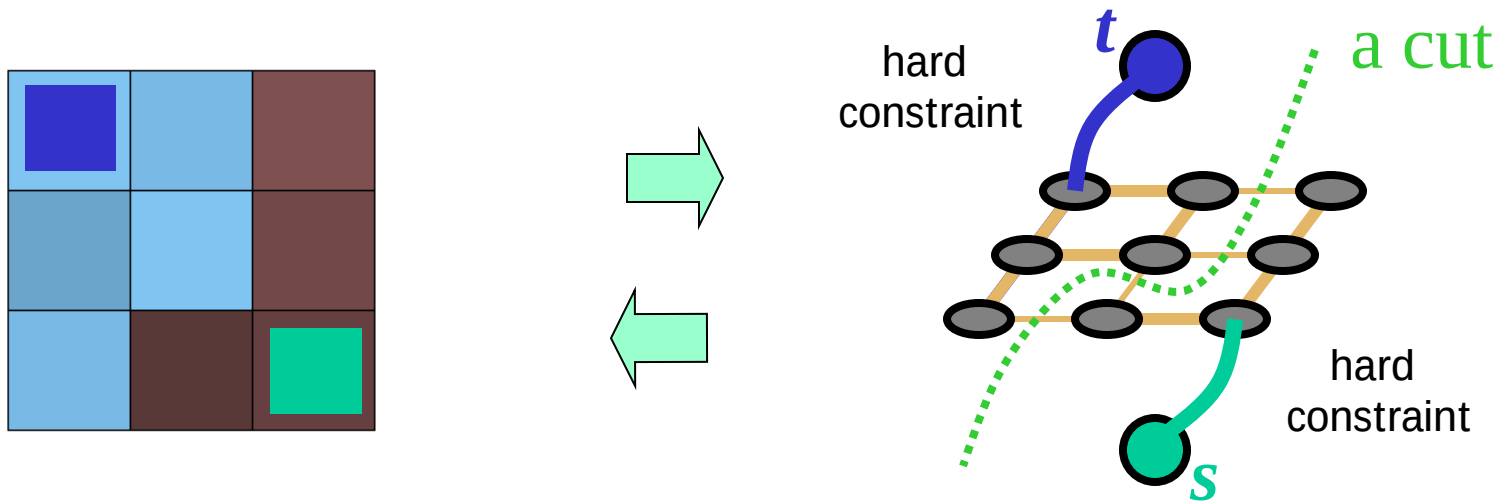
5	8	12	3
9	7	6	12
14	9	10	8
14	14	15	16

Graphcuts

What if we want similar “cut-where-things-agree” idea, but for closed regions?

- Dynamic programming can’t handle loops

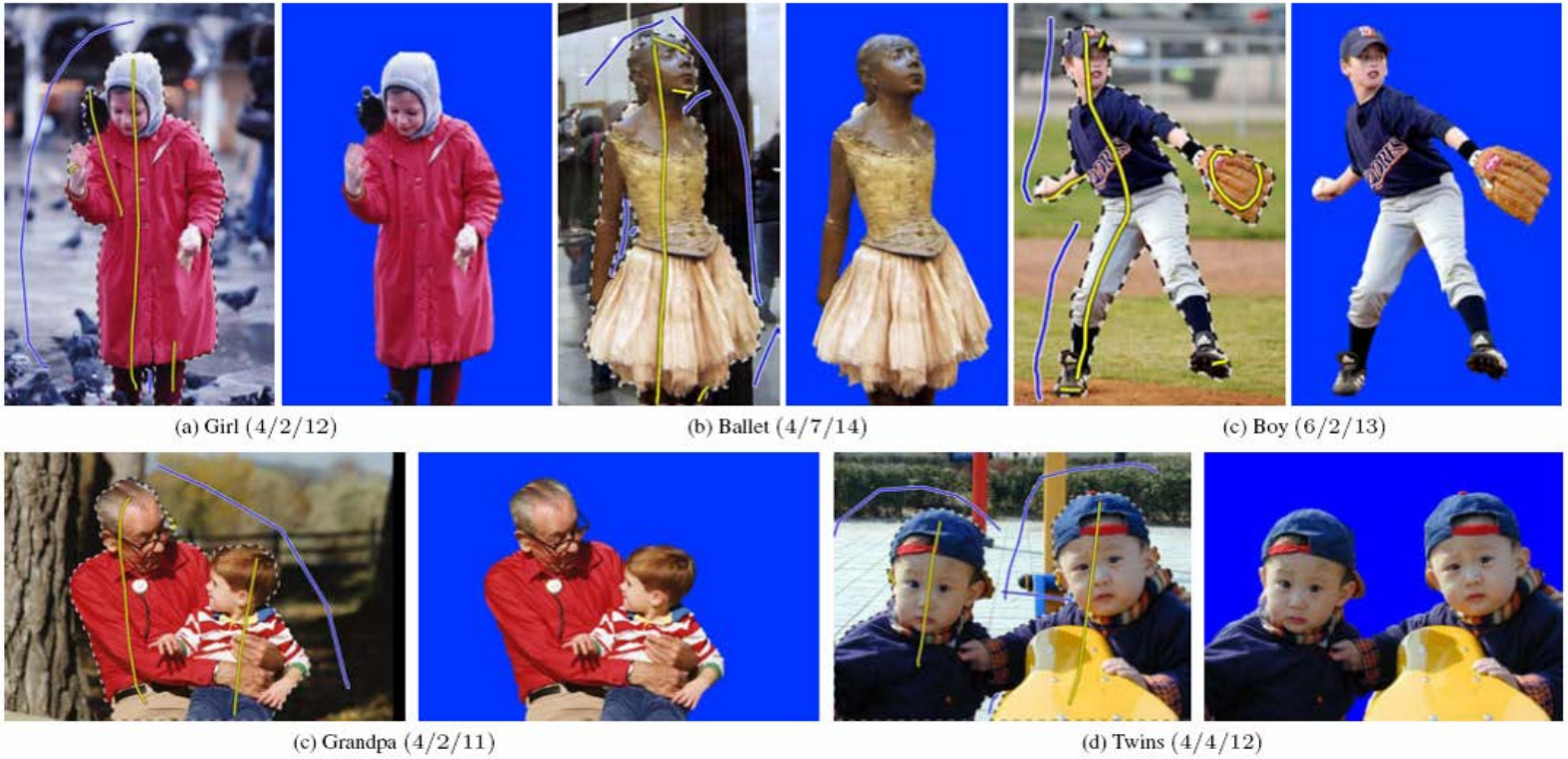
Graph cuts – a more general solution



Minimum cost cut can be computed in polynomial time

(max-flow/min-cut algorithms)

e.g. Lazy Snapping



(a) Girl (4/2/12) (b) Ballet (4/7/14) (c) Boy (6/2/13)

(c) Grandpa (4/2/11) (d) Twins (4/4/12)

Interactive segmentation using graphcuts

Also see the original Boykov&Jolly, ICCV'01, "GrabCut", etc, etc ,etc.

Putting it all together

Compositing images

- Have a clever blending function
  - Feathering
  - blend different frequencies differently
  - Gradient based blending
- Choose the right pixels from each image
  - Dynamic programming – optimal seams
  - Graph-cuts

Now, let’s put it all together:

- Interactive Digital Photomontage, 2004 (video)

Interactive Digital Photomontage

Aseem Agarwala, Mira Dontcheva
Maneesh Agrawala, Steven Drucker, Alex Colburn
Brian Curless, David Salesin, Michael Cohen

http://www.youtube.com/watch?v=kzV-5135bGA