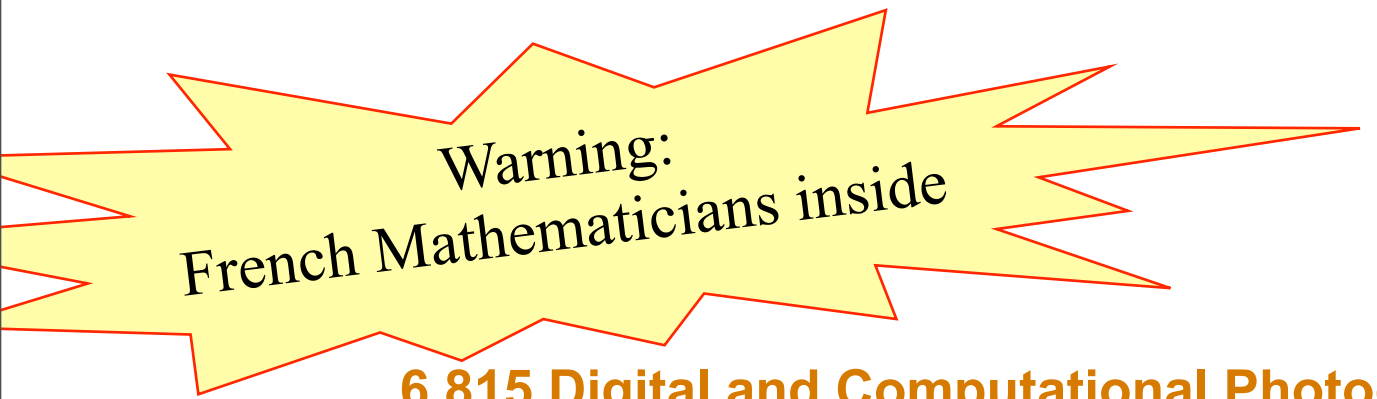
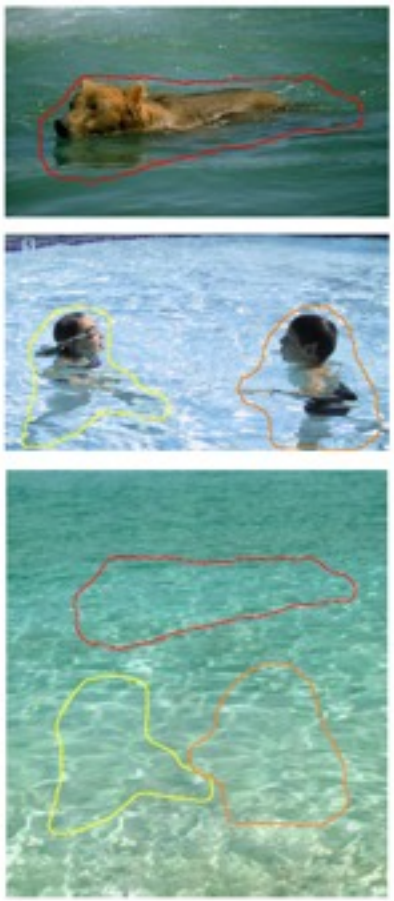


6.815 Digital and Computational Photography
6.865 Advanced Computational Photography

Gradient Image Processing

Frédo Durand
MIT - EECS

Problems with direct copy/paste

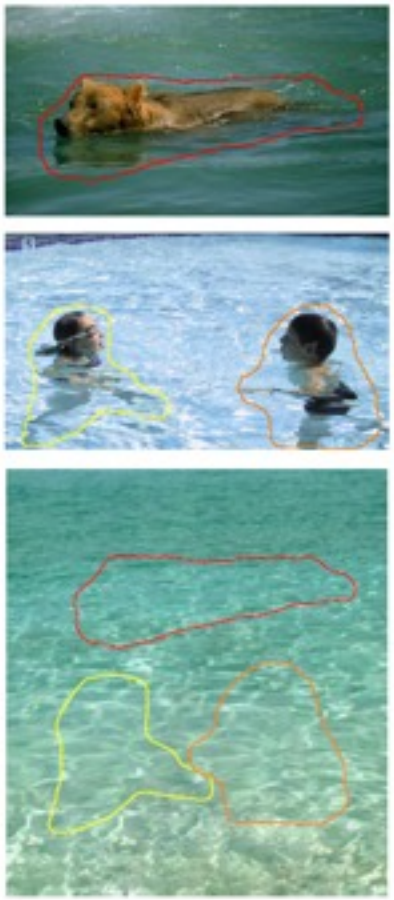


sources/destinations

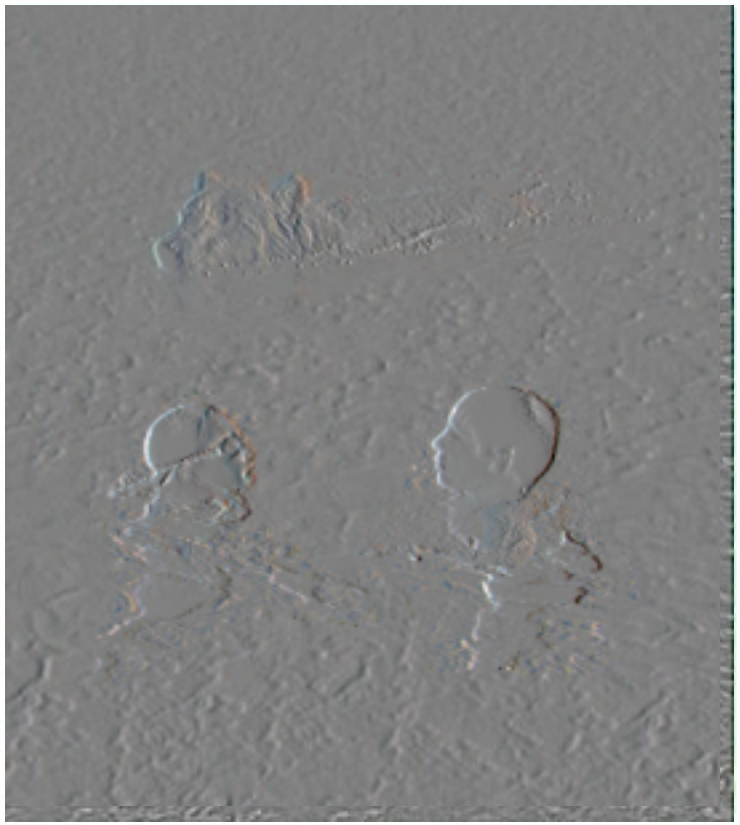
cloning

From Perez et al. 2003

Solution: paste gradient



sources/destinations



hacky visualization of gradient

seamless cloning

Demo of healing brush

- **Slightly smarter version of what we learn today**
  - higher-order derivative in particular

What is a gradient?

- **derivative of a multivariate function**
- **for example, for f(x,y)**

$$\nabla f = \left(\frac{df}{dx}, \frac{df}{dy} \right)$$

- **For a discrete image, can be approximated with finite differences**

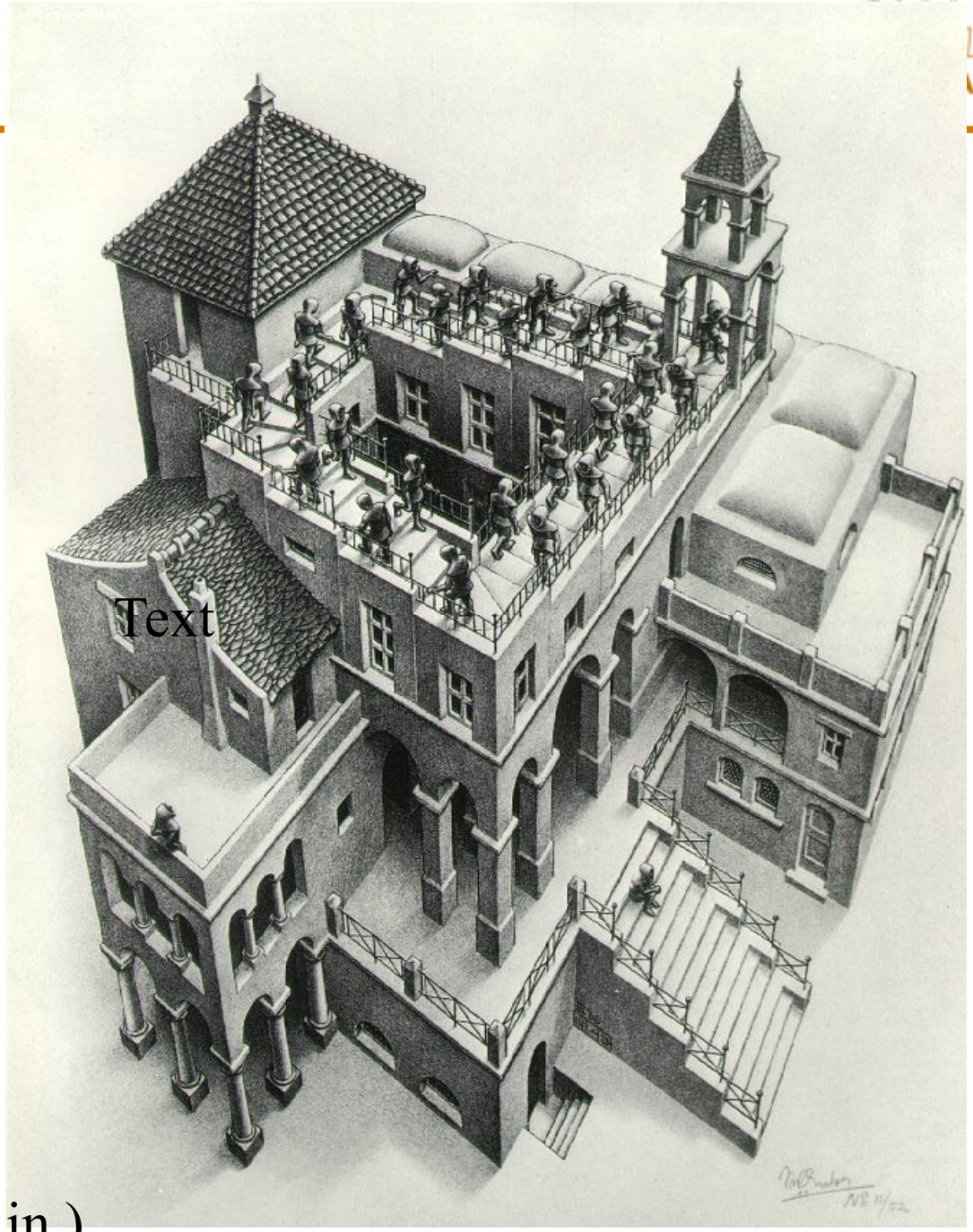
$$\frac{df}{dx} \approx f(x+1, y) - f(x, y)$$

$$\frac{df}{dy} \approx f(x, y+1) - f(x, y)$$

Gradient: intuition

Gradients and grayscale images

- **Grayscale image: n×n scalars**
- **Gradient: n×n *2D vectors***
- **Two many numbers!**
- **What's up with this?**
- **Not all vector fields are the gradient of an image!**
- **Only if they are curl-free (a.k.a. conservative)**
  - But we'll see it does not matter for us



Escher, Maurits Cornelis
Ascending and Descending
1960
Lithograph
35.5 x 28.5 cm (14 x 11 1/4 in.)

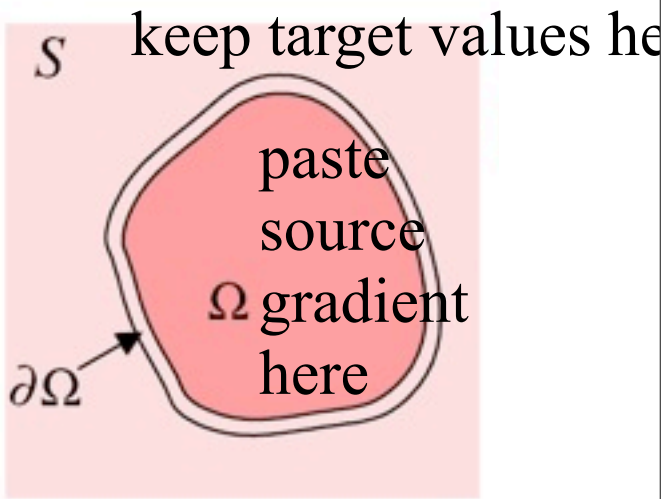
Color images

- 3 gradients, one for each channel.
- We'll sweep this under the rug for this lecture
- In practice, treat each channel independently

Questions?

Seamless Poisson cloning

- **Paste source gradient into target image inside a selected region**
- **Make the new gradient as close as possible to the source gradient while respecting pixel values at the boundary**



Seamless Poisson cloning

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

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

Poisson equation with Dirichlet conditions

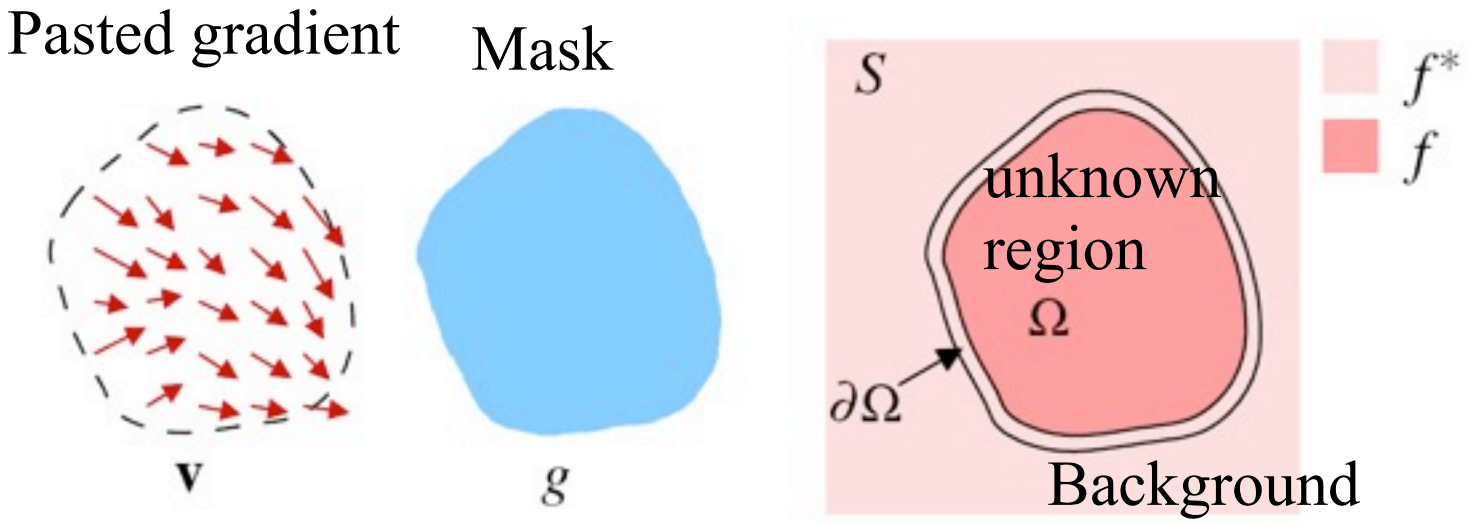
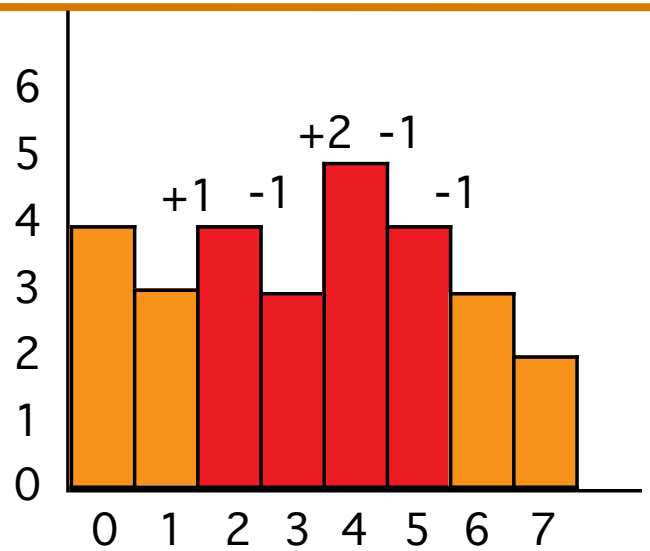


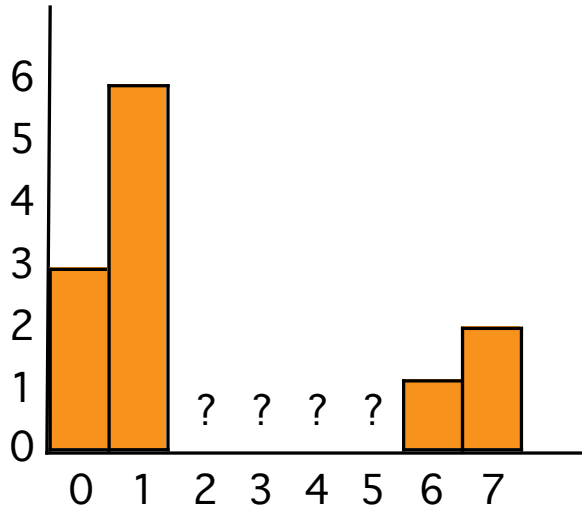
Figure 1: **Guided interpolation notations**. Unknown function f interpolates in domain Ω the destination function f^*, under guidance of vector field $\mathbf{v}$, which might be or not the gradient field of a source function g.

Discrete 1D example: minimization

- **Copy**



to

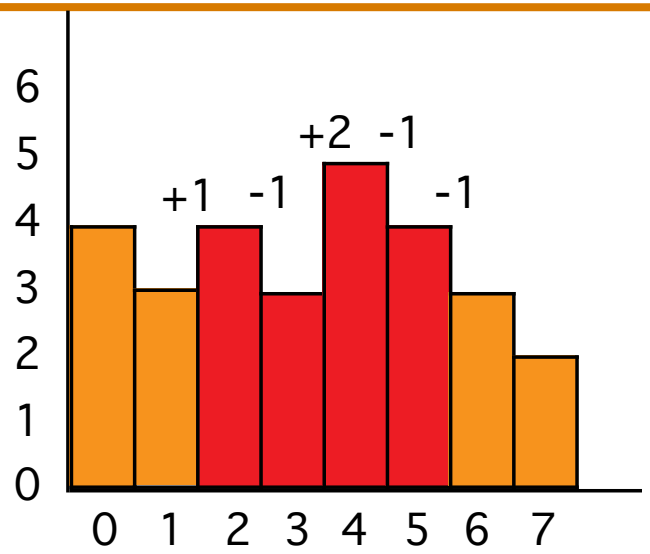


$$\text{Min } [(f_2-f_1)-1]^2$$
$$+ [(f_3-f_2)-(-1)]^2$$
$$+ [(f_4-f_3)-2]^2$$
$$+ [(f_5-f_4)-(-1)]^2$$
$$+ [(f_6-f_5)-(-1)]^2$$

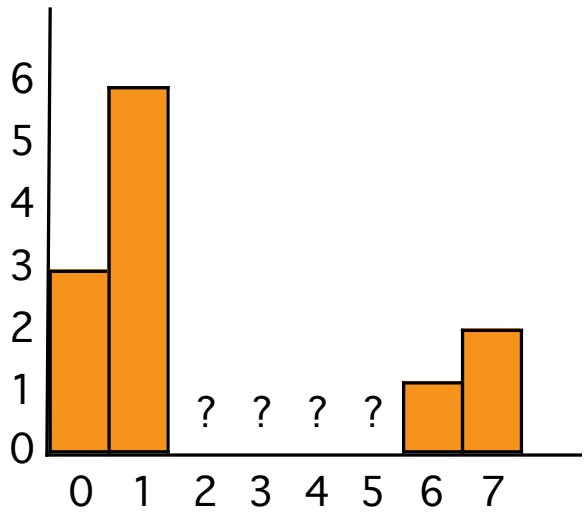
With
$f_1=6$
$f_6=1$

1D example: minimization

- **Copy**



to



$$\text{Min } [(f_2-f_1)-1]^2 \quad ==> \quad f_2^2+49-14f_2$$

$$+ [(f_3-f_2)-(-1)]^2 \quad ==> \quad f_3^2+f_2^2+1-2f_3f_2 +2f_3-2f_2$$

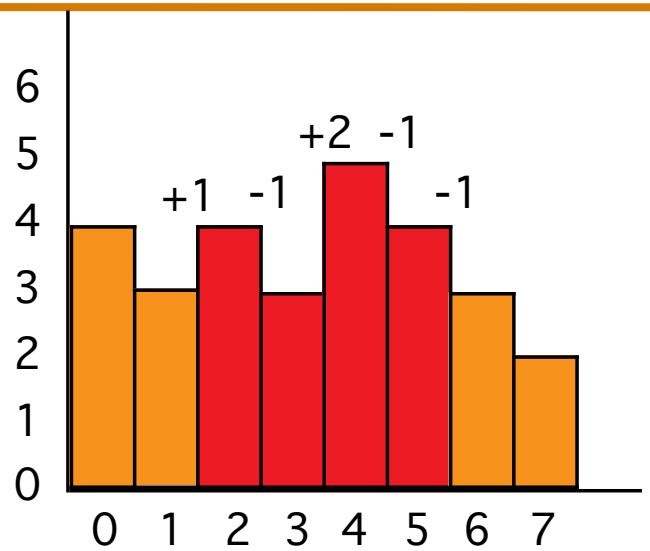
$$+ [(f_4-f_3)-2]^2 \quad ==> \quad f_4^2+f_3^2+4-2f_3f_4 -4f_4+4f_3$$

$$+ [(f_5-f_4)-(-1)]^2 \quad ==> \quad f_5^2+f_4^2+1-2f_5f_4 +2f_5-2f_4$$

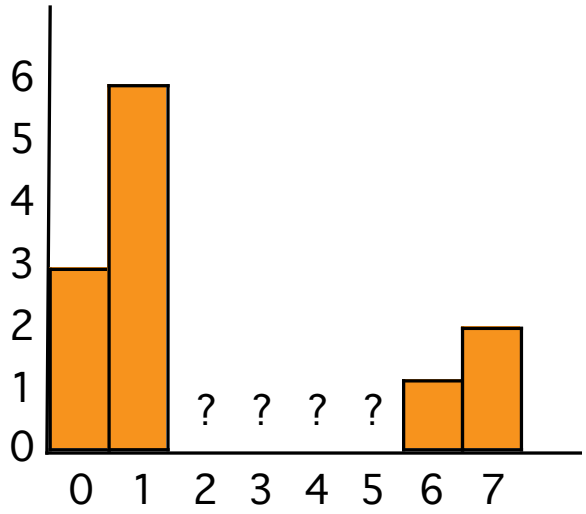
$$+ [(f_6-f_5)-(-1)]^2 \quad ==> \quad f_5^2+4-4f_5$$

1D example: big quadratic

- **Copy**



to



- **Min (f_2^2+49-14f_2**

  + f_3^2+f_2^2+1-2f_3f_2 +2f_3-2f_2

  + f_4^2+f_3^2+4-2f_3f_4 -4f_4+4f_3

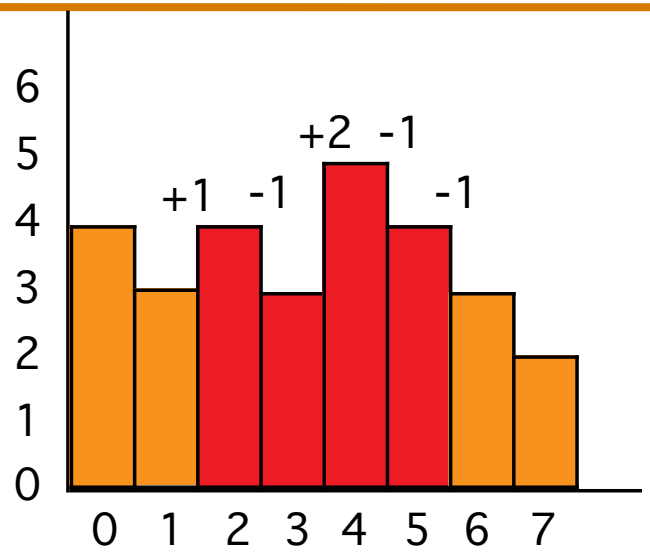
  + f_5^2+f_4^2+1-2f_5f_4 +2f_5-2f_4

  + f_5^2+4-4f_5)

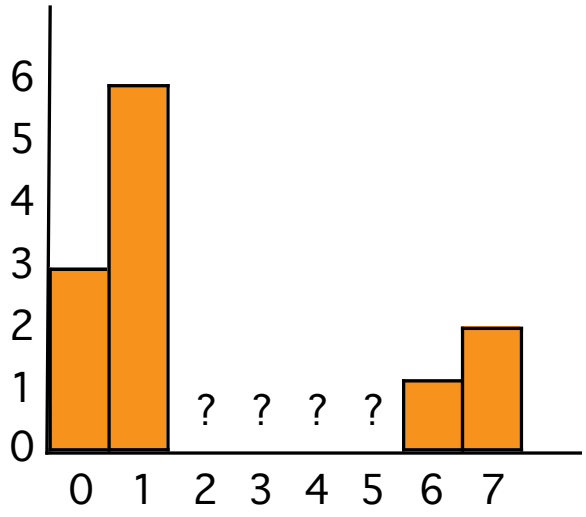
  Denote it Q

1D example: derivatives

- **Copy**



to



Min (f_2^2+49-14f_2

+ f_3^2+f_2^2+1-2f_3f_2 +2f_3-2f_2

+ f_4^2+f_3^2+4-2f_3f_4 -4f_4+4f_3

+ f_5^2+f_4^2+1-2f_5f_4 +2f_5-2f_4

+ f_5^2+4-4f_5)

Denote it Q

$$\frac{dQ}{df_2} = 2f_2 + 2f_2 - 2f_3 - 16$$

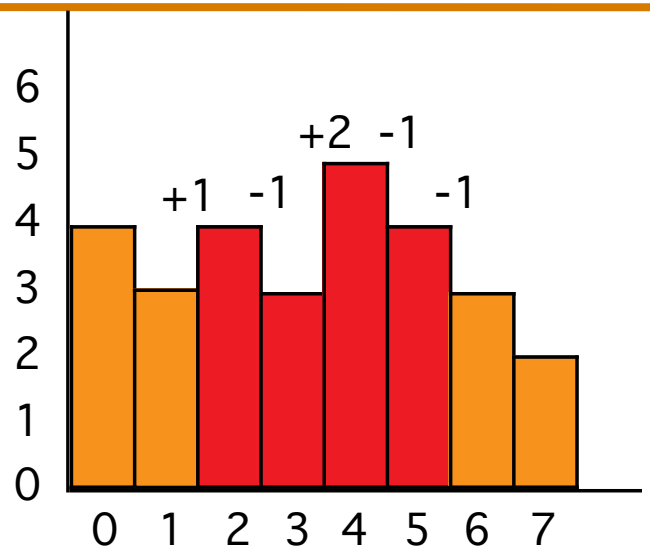
$$\frac{dQ}{df_3} = 2f_3 - 2f_2 + 2 + 2f_3 - 2f_4 + 4$$

$$\frac{dQ}{df_4} = 2f_4 - 2f_3 - 4 + 2f_4 - 2f_5 - 2$$

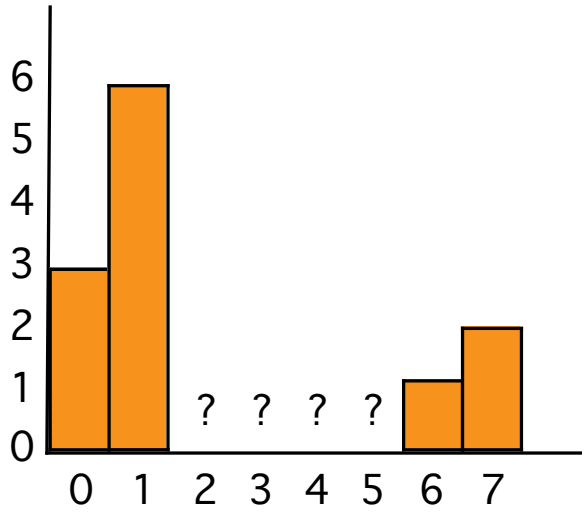
$$\frac{dQ}{df_5} = 2f_5 - 2f_4 + 2 + 2f_5 - 4$$

1D example: set derivatives to zero

- **Copy**



to



$$\frac{dQ}{df_2} = 2f_2 + 2f_2 - 2f_3 - 16 \quad =0$$

$$\frac{dQ}{df_3} = 2f_3 - 2f_2 + 2 + 2f_3 - 2f_4 + 4 \quad =0$$

$$\frac{dQ}{df_4} = 2f_4 - 2f_3 - 4 + 2f_4 - 2f_5 - 2 \quad =0$$

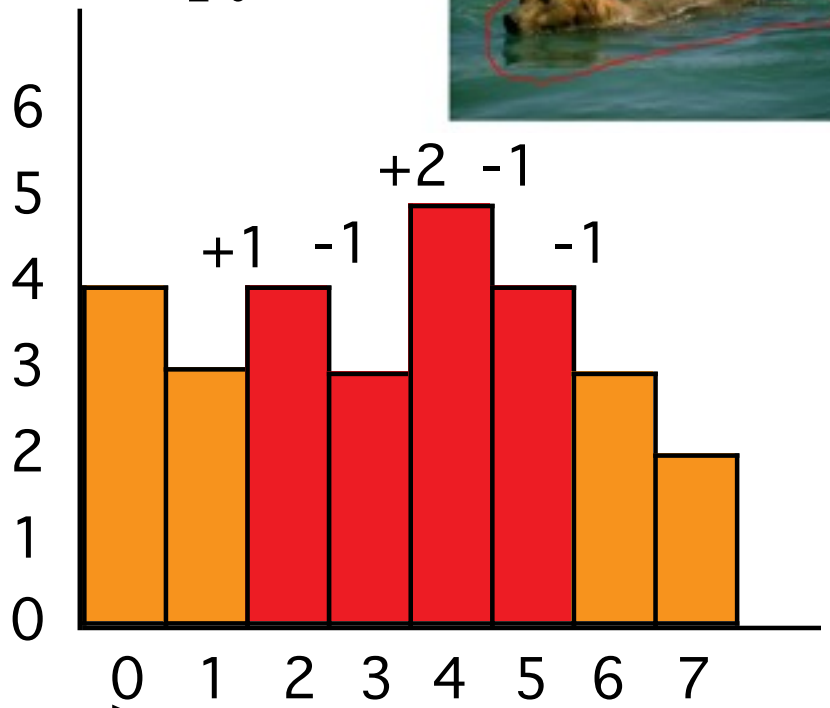
$$\frac{dQ}{df_5} = 2f_5 - 2f_4 + 2 + 2f_5 - 4 \quad =0$$

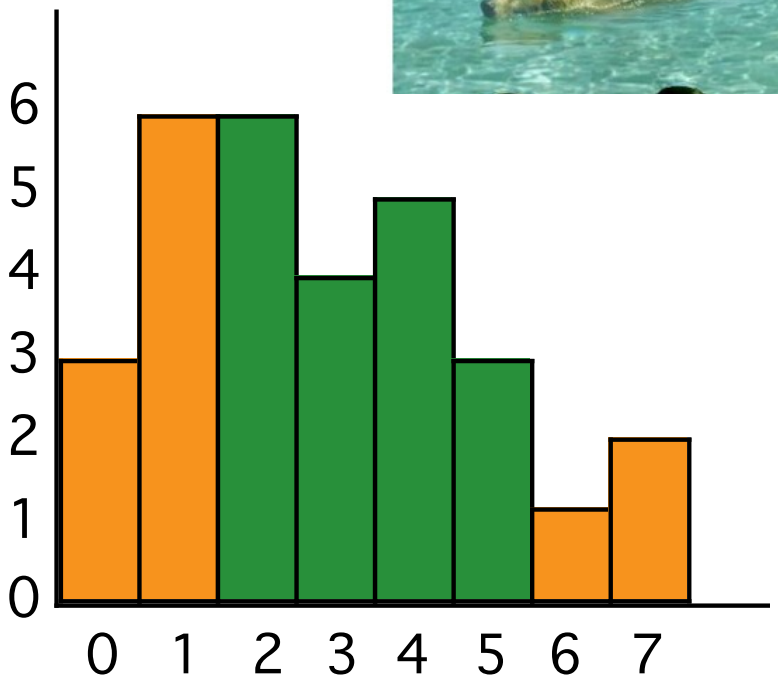
$$\Longrightarrow \begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 16 \\ -6 \\ 6 \\ 2 \end{pmatrix}$$

1D example recap

- **Copy**

to





==>

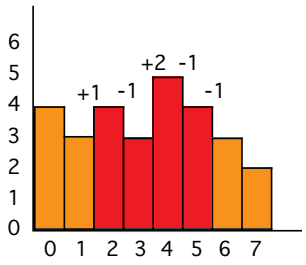
$$\begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 16 \\ -6 \\ 6 \\ 2 \end{pmatrix}$$

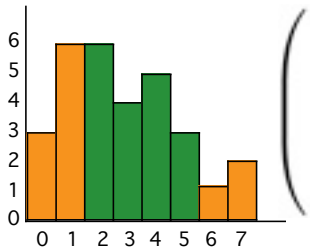
$$\begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \\ 5 \\ 3 \end{pmatrix}$$

Questions?

- **Recap:**
  - copy gradient, not pixel values
  - enforce boundary condition
  - solve linear least square:
    minimize square difference with source gradient

1D example: remarks

- **Copy** 

to 

$$\begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 16 \\ -6 \\ 6 \\ 2 \end{pmatrix}$$

- **Matrix is sparse**
  - many zero coefficients
  - because gradient only depends on neighboring pixels
- **Matrix is symmetric**
- **Everything is a multiple of 2**
  - because square and derivative of square
- **Matrix is a convolution (kernel -2 4 -2)**
  - all the rows are the same, just shifted
- **Matrix is independent of gradient field. Only RHS is**
- **Matrix is a second derivative**

Questions?

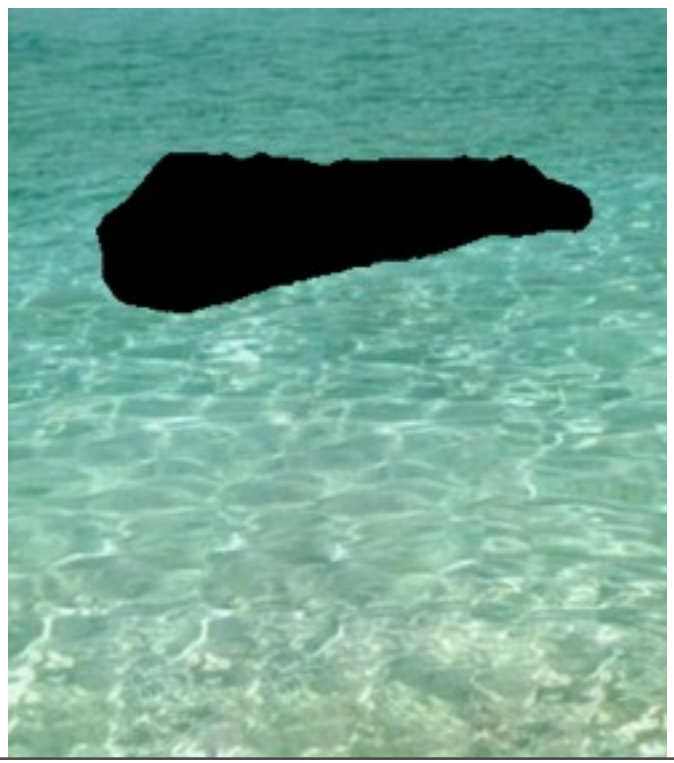
Let's try to further analyze

- **What is a simple case?**

Membrane interpolation

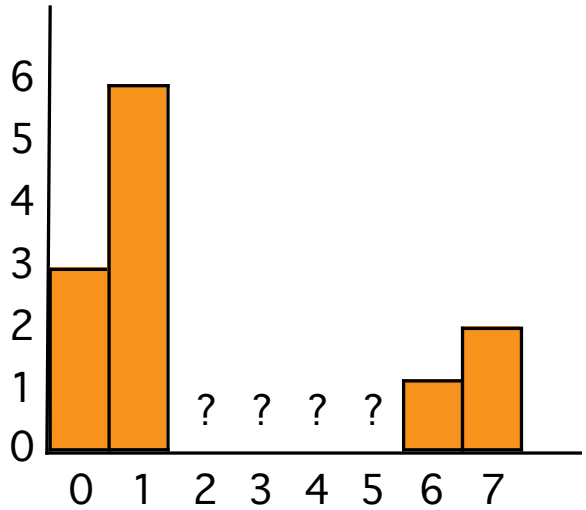
- **What if *v* is null?**
- **Laplace equation (a.k.a. membrane equation)**

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



1D example: minimization

- **Minimize derivatives to interpolate**

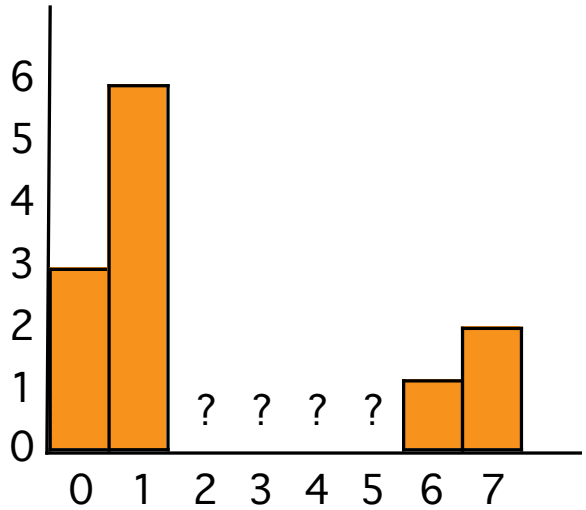


$$\text{Min } (f_2-f_1)^2 + (f_3-f_2)^2 + (f_4-f_3)^2 + (f_5-f_4)^2 + (f_6-f_5)^2$$

With
$f_1=6$
$f_6=1$

1D example: derivatives

- **Minimize derivatives to interpolate**



Min (f_2^2+36-12f_2

+ f_3^2+f_2^2-2f_3f_2

+ f_4^2+f_3^2-2f_3f_4

+ f_5^2+f_4^2-2f_5f_4

+ f_5^2+1-2f_5)

Denote it Q

$$\frac{dQ}{df_2} = 2f_2 + 2f_2 - 2f_3 - 12$$

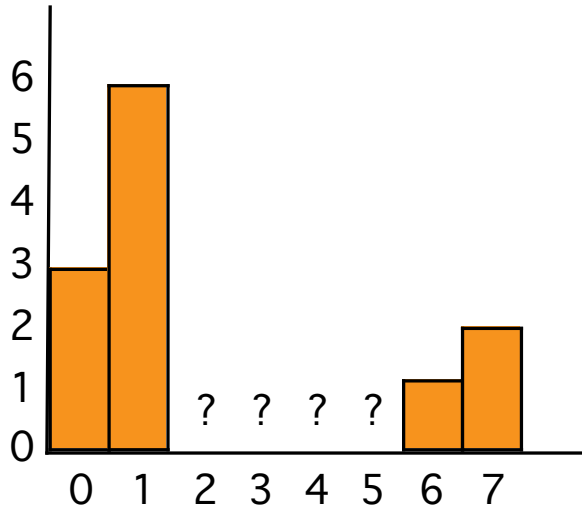
$$\frac{dQ}{df_3} = 2f_3 - 2f_2 + 2f_3 - 2f_4$$

$$\frac{dQ}{df_4} = 2f_4 - 2f_3 + 2f_4 - 2f_5$$

$$\frac{dQ}{df_5} = 2f_5 - 2f_4 + 2f_5 - 2$$

1D example: set derivatives to zero

- **Minimize derivatives to interpolate**



$$\frac{dQ}{df_2} = 2f_2 + 2f_2 - 2f_3 - 12$$

$$\frac{dQ}{df_3} = 2f_3 - 2f_2 + 2f_3 - 2f_4$$

$$\frac{dQ}{df_4} = 2f_4 - 2f_3 + 2f_4 - 2f_5$$

$$\frac{dQ}{df_5} = 2f_5 - 2f_4 + 2f_5 - 2$$

$$\Longrightarrow \begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 12 \\ 0 \\ 0 \\ 2 \end{pmatrix}$$

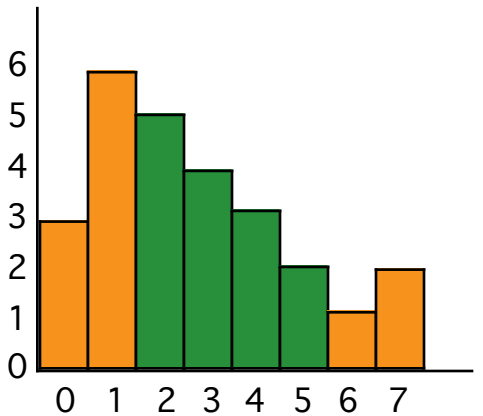
1D example

- **Minimize derivatives to interpolate**

- **Pretty much says that second derivative should be zero**

  (-1 2 -1)

  is a second derivative filter

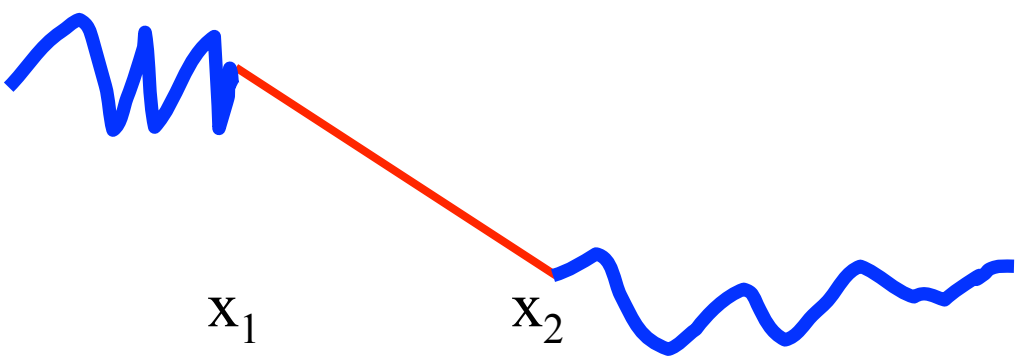


$$\begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 12 \\ 0 \\ 0 \\ 2 \end{pmatrix}$$

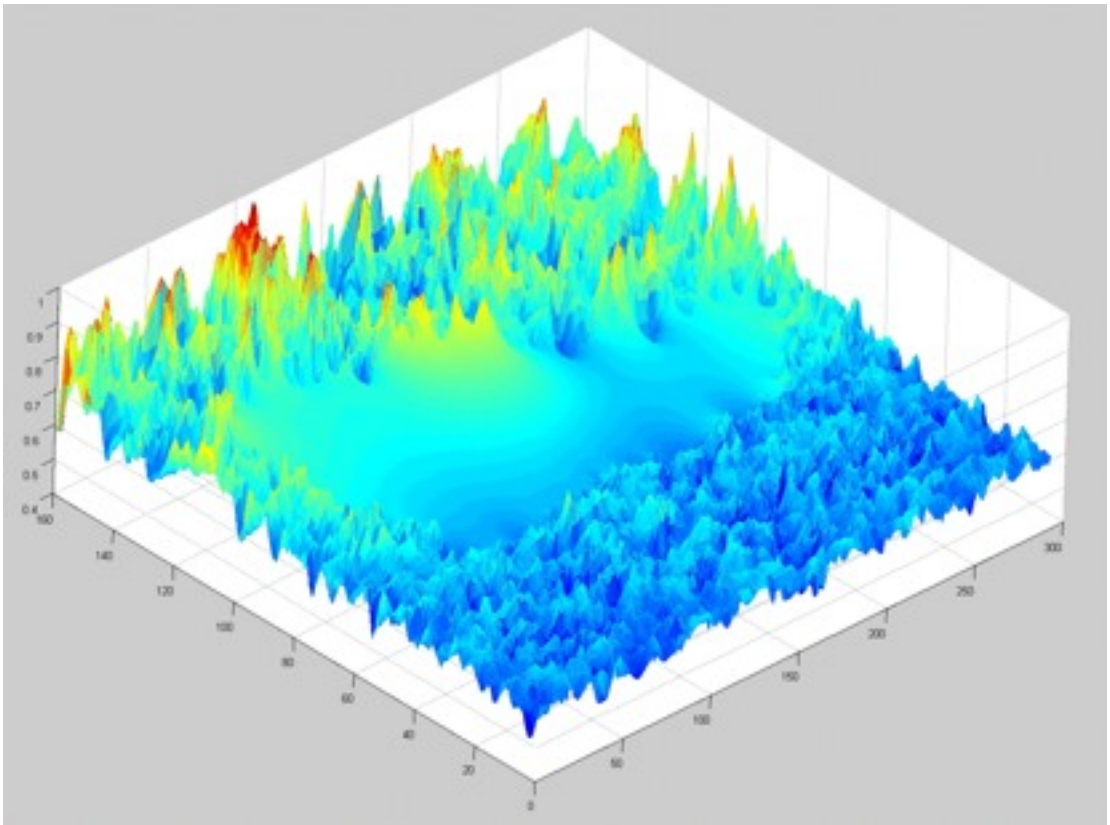
$$\begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 5 \\ 4 \\ 3 \\ 2 \end{pmatrix}$$

Intuition

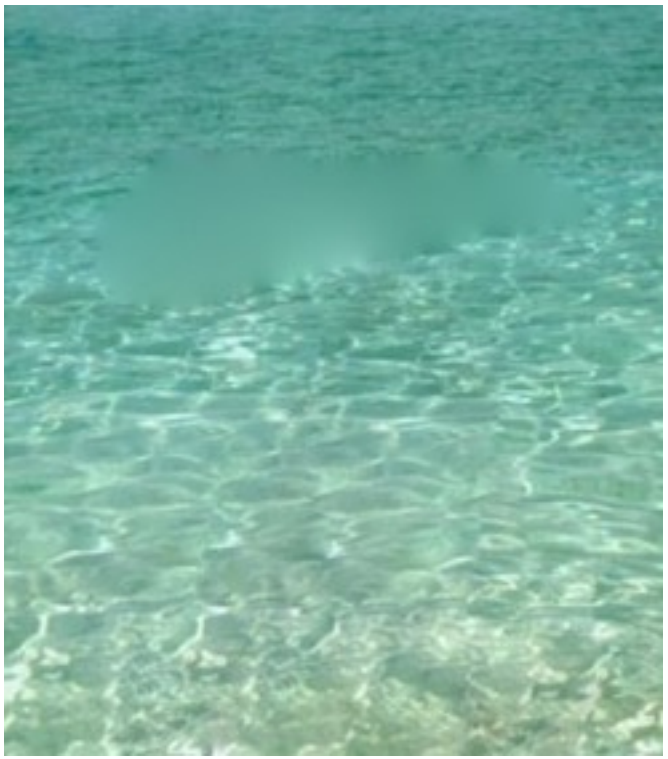
- **In 1D; just linear interpolation!**
- **Locally, if the second derivative was not zero, this would mean that the first derivative is varying, which is bad since we want $(\nabla f)^2$ to be minimized**
- **Note that, in 1D: by setting f'', we leave two degrees of freedom. This is exactly what we need to control the boundary condition at x_1 and x_2**

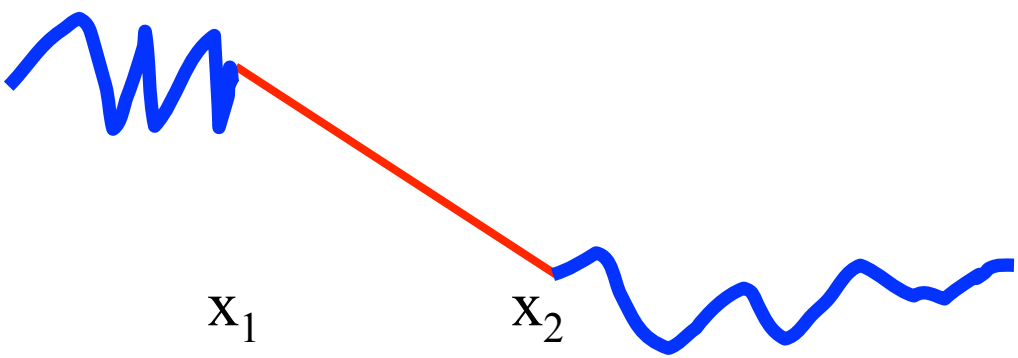


In 2D: membrane interpolation



Not as simple

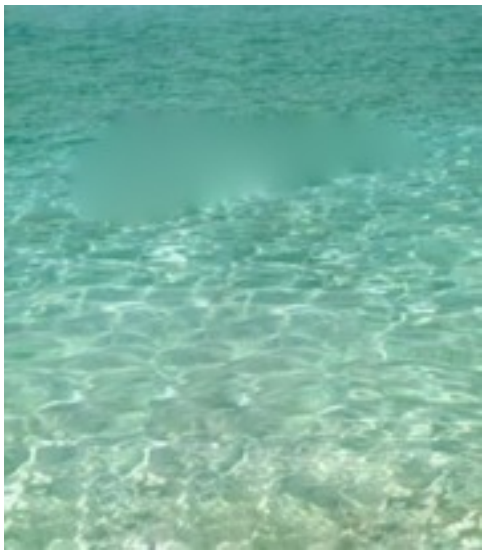




Membrane interpolation

- **When *v* is null:**
- **Laplace equation (a.k.a. membrane equation)**

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



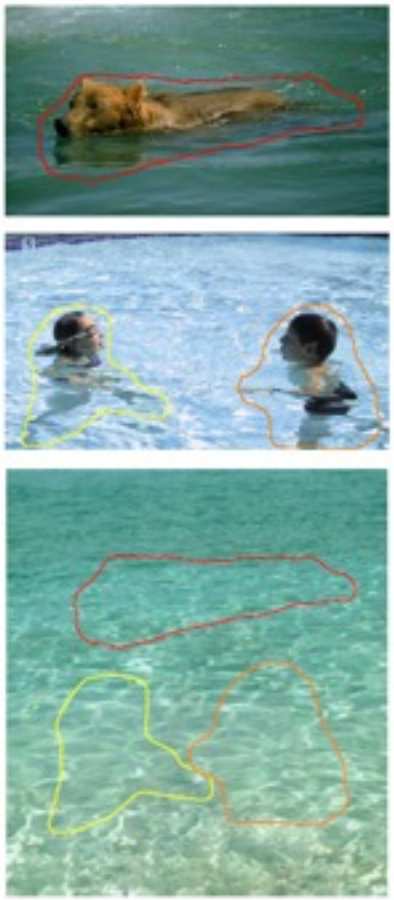
- **Mathematicians will tell you there is an Associated Euler-Lagrange equation:**

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

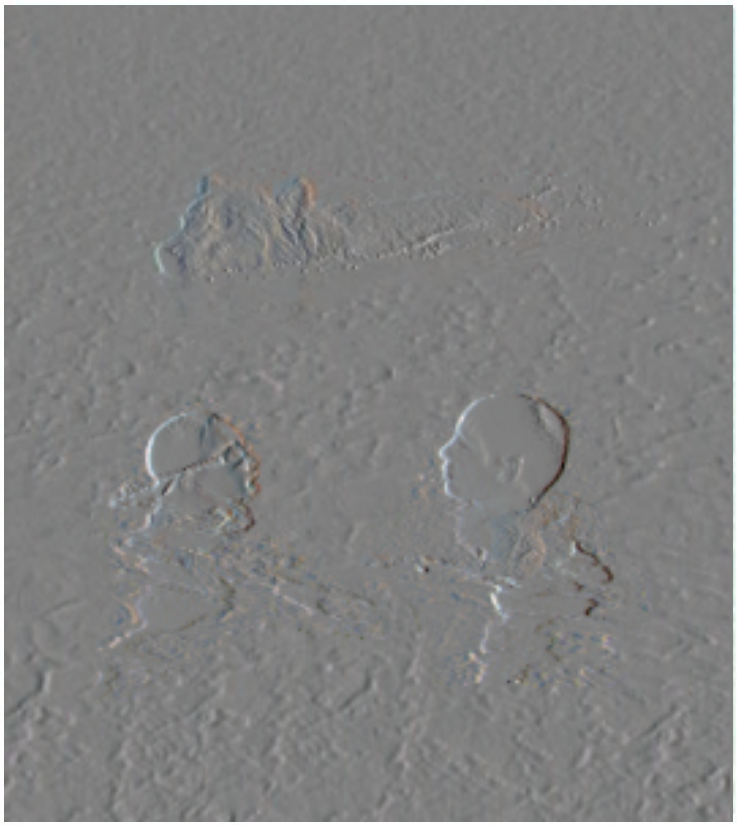
  - Where the Laplacian Δ is similar to -1 2 -1 in 1D
- **Kind of the idea that we want a minimum, so we kind of derive and get a simpler equation**

Questions?

What if *v* is not null?



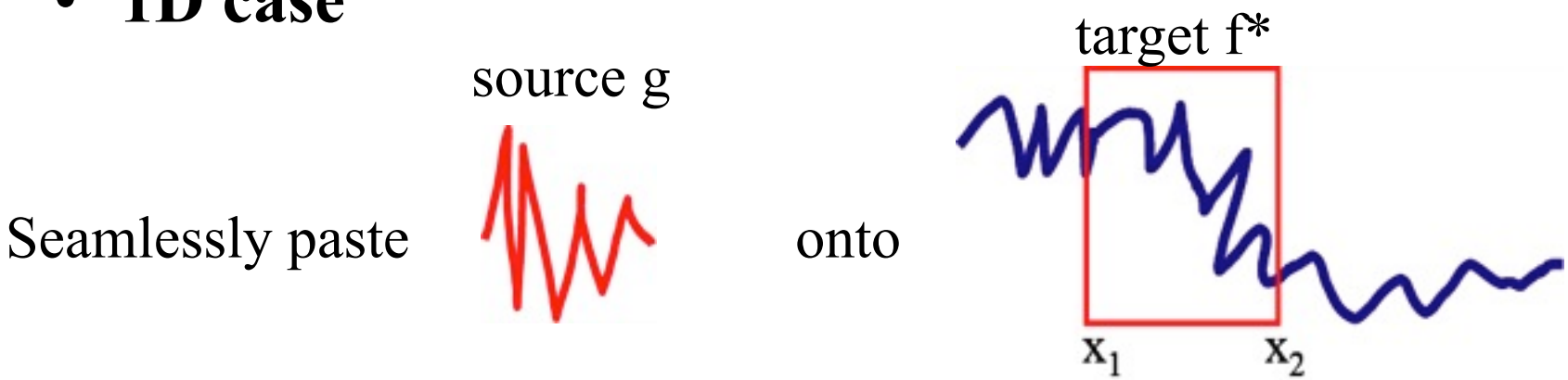
sources/destinations



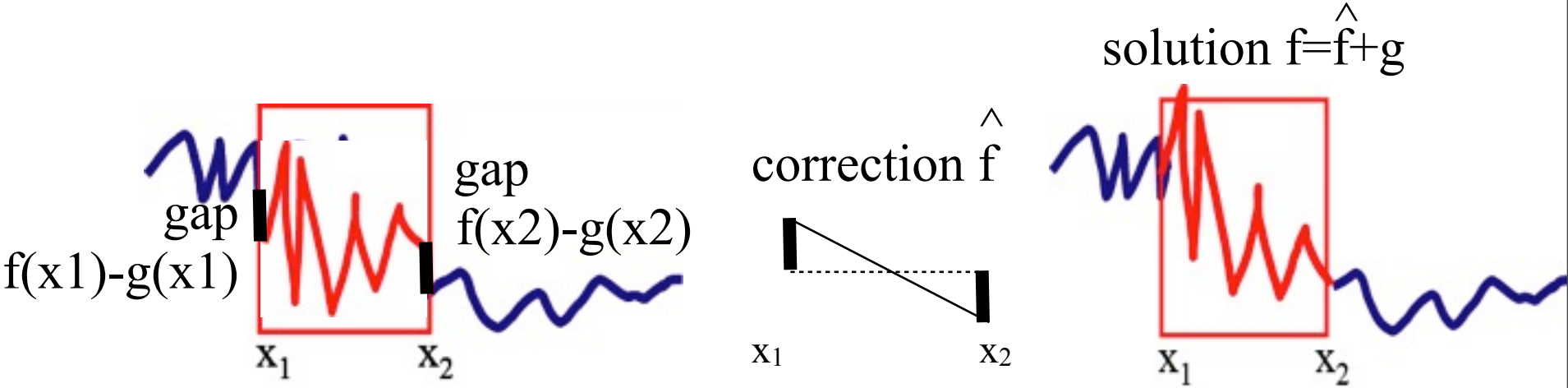
seamless cloning

What if *v* is not null?

- **1D case**



Just add a linear function so that the boundary condition is respected



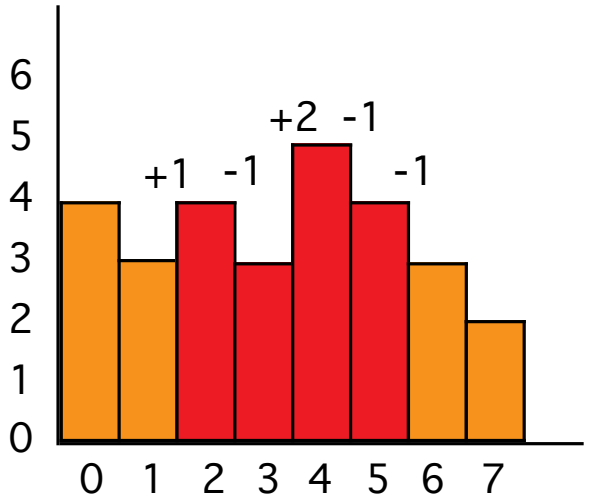
Recap 1D case

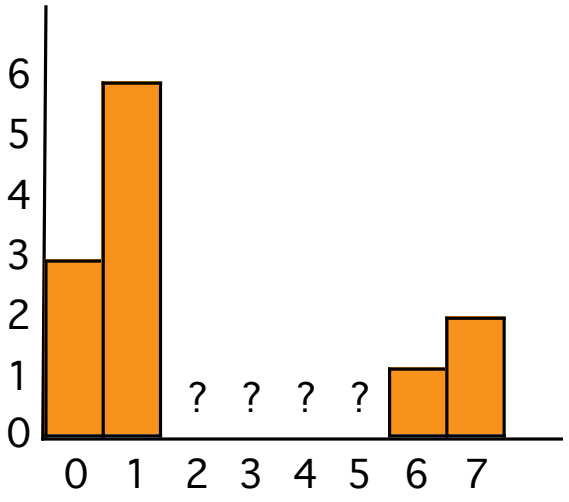
- **Poisson clone of g into f* between x1 and x2**
- **if g is null, simple linear function**
  - $f(x) = (x2-x)/(x2-x1)f^*(x1)+(x-x1)/(x2-x1)f^*(x2)$
- **otherwise, add a correction function to g in order to linearly interpolate between f*(x1)-g(x1) and f*(x2)-g(x2)**
  - $f(x)=\hat{f}(x)+g(x)$
  - where

    $f(x) = (x2-x)/(x2-x1)(f^*(x1)-g(x1)$

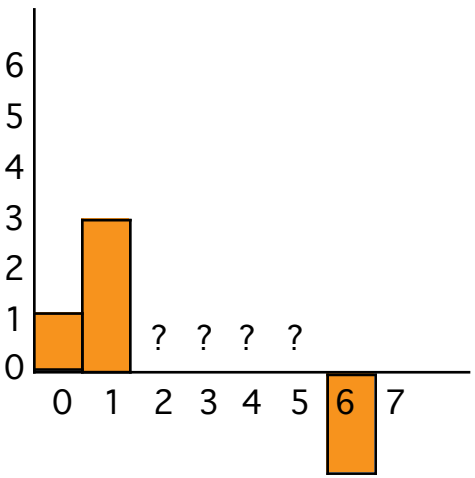
    $+(x-x1)/(x2-x1)(f^*(x2)-g(x2))$
  - Note that boundary conditions are respected and the difference to g is spread uniformly

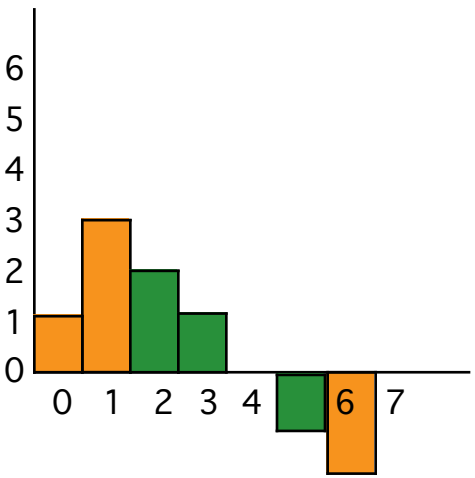
1D example

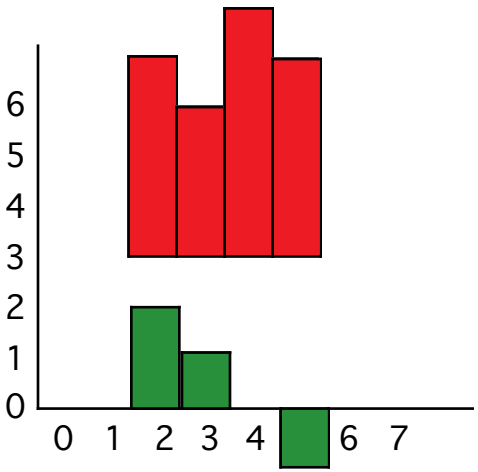
- **Copy** 

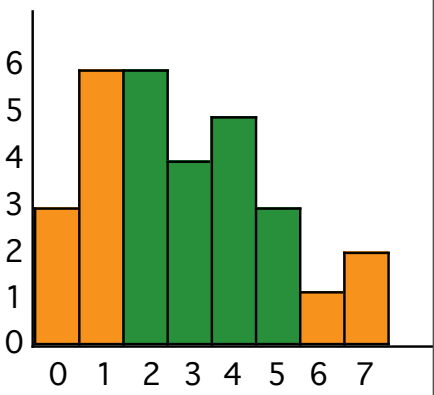
to 

Difference Solve Laplace Add Result







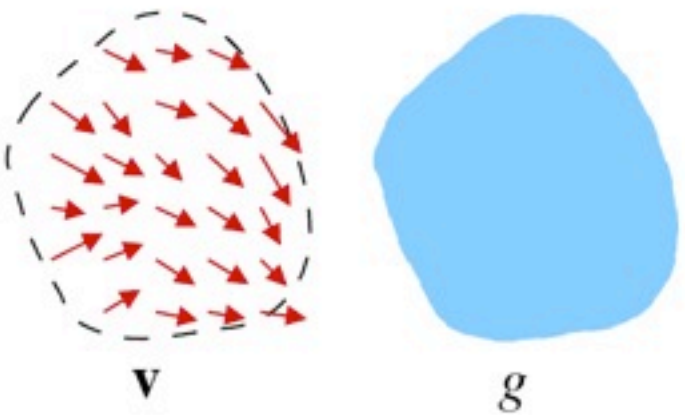


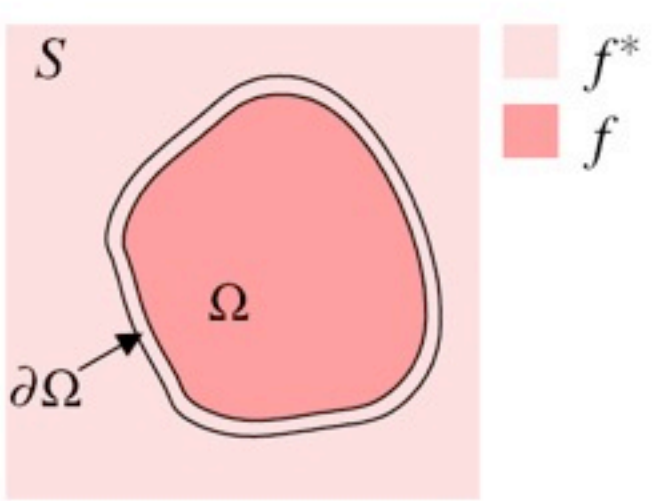
Questions?

In 2D, if *v* is conservative

- If *v* is the gradient of an image g (it is conservative)
- Correction function $\hat{f}$ so that $f = g + \hat{f}$
- $\hat{f}$ performs membrane interpolation over Ω:

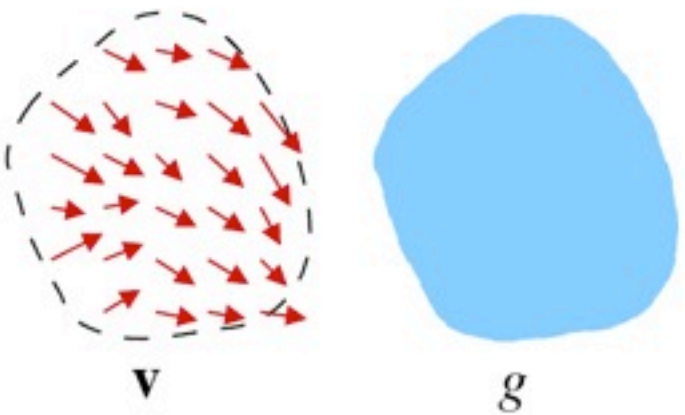
$$\Delta \tilde{f} = 0 \text{ over } \Omega, \; \tilde{f}|_{\partial\Omega} = (f^* - g)|_{\partial\Omega}$$

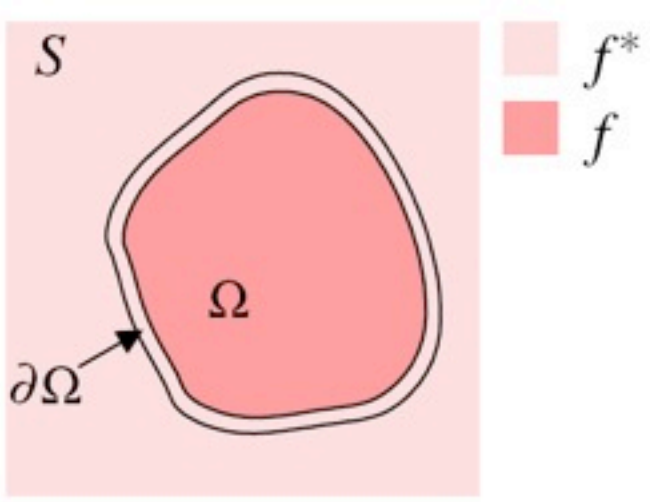




In 2D, if *v* is NOT conservative

- Also need to project the vector field v to a conservative field
- And do the membrane thing
- Of course, we do not need to worry about it, it's all handled naturally by the least square approach





Questions?

Recap

- **Find image whose gradient best approximates the input gradient**
  - least square Minimization
- **Discrete case: turns into linear equation**
  - Set derivatives to zero
  - Derivatives of quadratic ==> linear
- **When gradient is null, membrane interpolation**
  - Linear interpolation in 1D

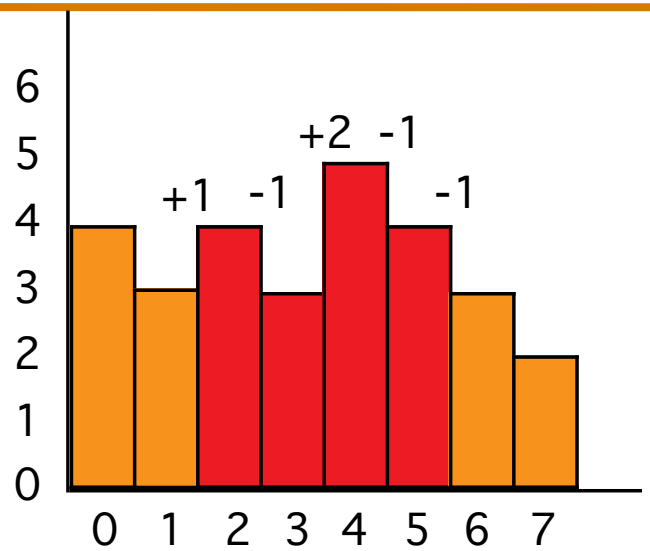
Fourier interpretation

- **Least square on gradient** $\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$
- **Parseval anybody?**
  - Integral of squared stuff is the same in Fourier and primal
- **What is the gradient/derivative in Fourier?**
  - Multiply coefficients by frequency and i
- **Seen in Fourier domain, Poisson editing does a weighted least square of the image where low frequencies have a small weight and high frequencies a big weight**

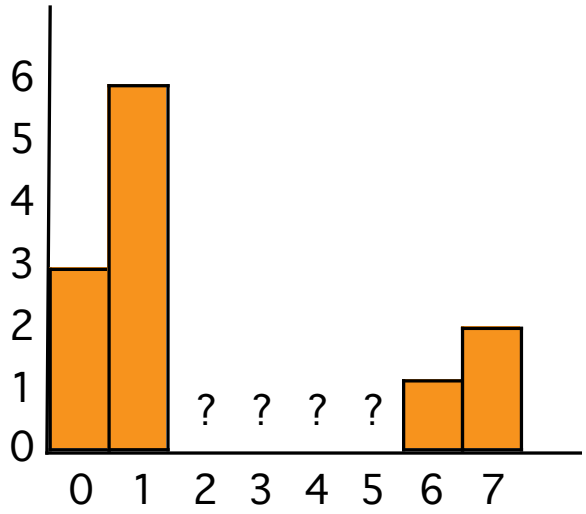
Questions?

Discrete solver: Recall 1D

- **Copy**



to



$$\frac{dQ}{df_2} = 2f_2 + 2f_2 - 2f_3 - 16$$

$$\frac{dQ}{df_3} = 2f_3 - 2f_2 + 2 + 2f_3 - 2f_4 + 4$$

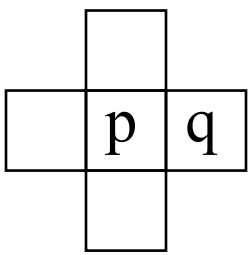
$$\frac{dQ}{df_4} = 2f_4 - 2f_3 - 4 + 2f_4 - 2f_5 - 2$$

$$\frac{dQ}{df_5} = 2f_5 - 2f_4 + 2 + 2f_5 - 4$$

$$\Longrightarrow \begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 16 \\ -6 \\ 6 \\ 2 \end{pmatrix}$$

Discrete Poisson solver

- **Minimize variational problem** $\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega},$
- **Discretize derivatives**
  - Finite differences over pairs of pixel neighbors
  - We are going to work using pairs of pixels



Discrete Poisson solver

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

Discretized gradient

$$\min_{f|_\Omega} \sum_{\langle p,q\rangle \cap \Omega \neq \emptyset} (f_p - f_q - v_{pq})^2, \text{ with } f_p = f_p^*, \text{for all } p \in \partial\Omega$$

(all pairs that are in Ω)

Discretized v: g(p)-g(q)

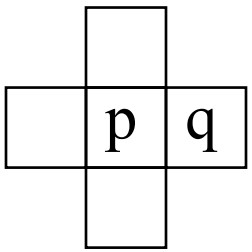
Boundary condition

- **Derive, rearrange and call N_p the neighbors of p**

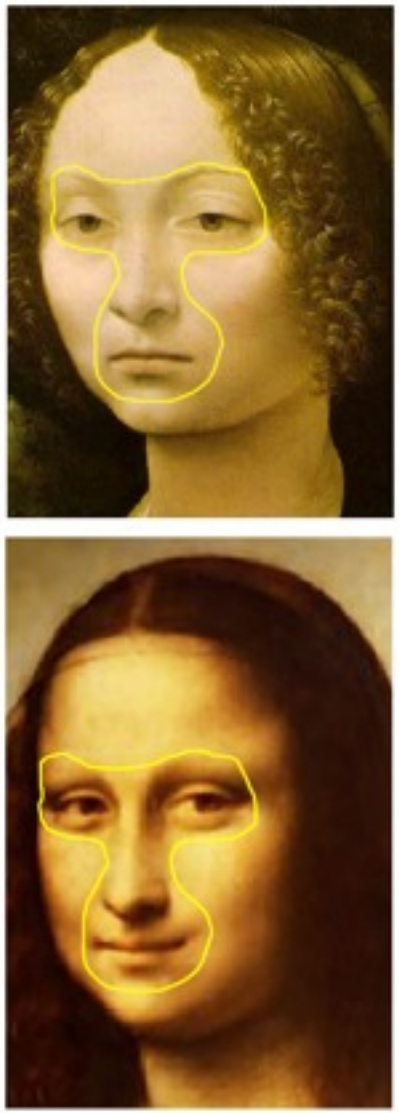
$$\text{for all } p \in \Omega, \quad |N_p| f_p - \sum_{q \in N_p \cap \Omega} f_q = \sum_{q \in N_p \cap \partial\Omega} f_q^* + \sum_{q \in N_p} v_{pq}$$

Only for boundary pixels

- **Big yet sparse linear system**



Result (eye candy)



source/destination

cloning

seamless cloning

Questions?

Recap

- **Find image whose gradient best approximates the input gradient**
  - least square Minimization
- **Discrete case: turns into big sparse linear equation**
  - Set derivatives to zero
  - Derivatives of quadratic ==> linear

Solving big matrix systems

- **Ax=b**
- **You can use Matlab's **
  - (Gaussian elimination)
  - But not very scalable

Typical sizes

- **e.g. solve Poisson in a 100x100 image region**
- **10,000 unknowns**
- **10,000x10,000 matrix!**

Iterative solvers

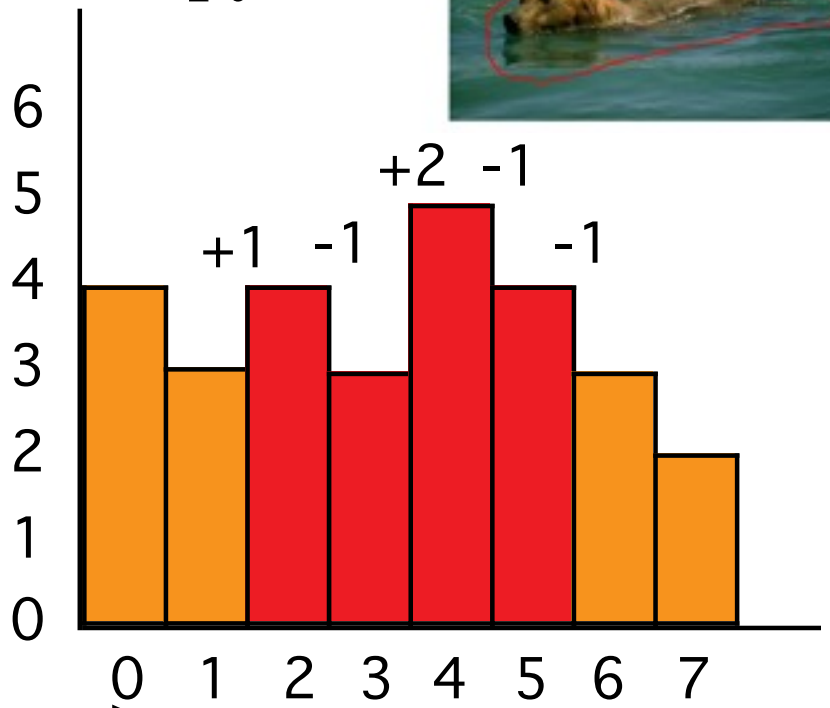
Important ideas

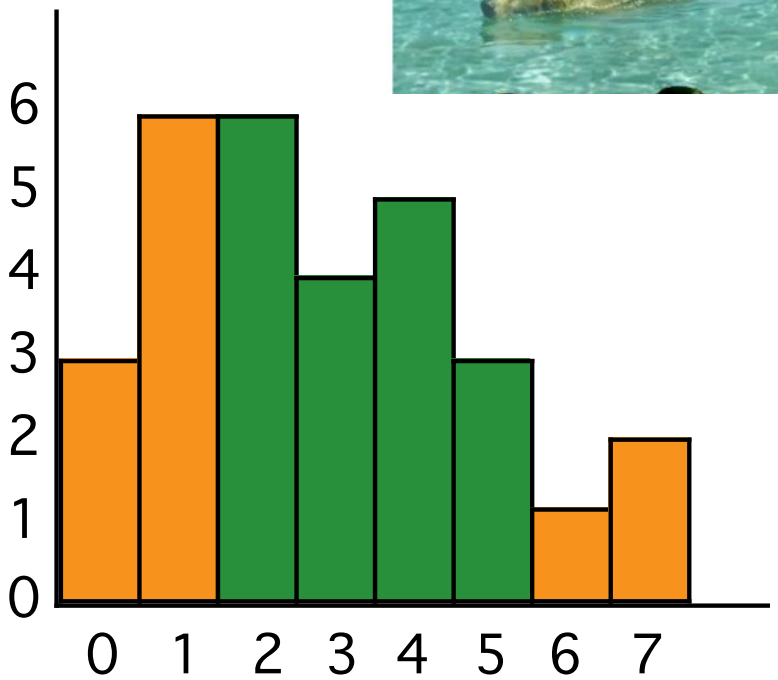
- **Do not inverse matrix**
- **Maintain a vector x' that progresses towards the solution**
- **Updates mostly require to *apply* the matrix.**
  - In many cases, it means you do no even need to store the matrix (e.g. for a convolution matrix you only need the kernel)
- **Usually, you don't even wait until convergence**
- **Big questions: in which direction do you walk?**
  - Yes, very similar to gradient descent
  - How far do you go?

1D example recap

- **Copy**

to



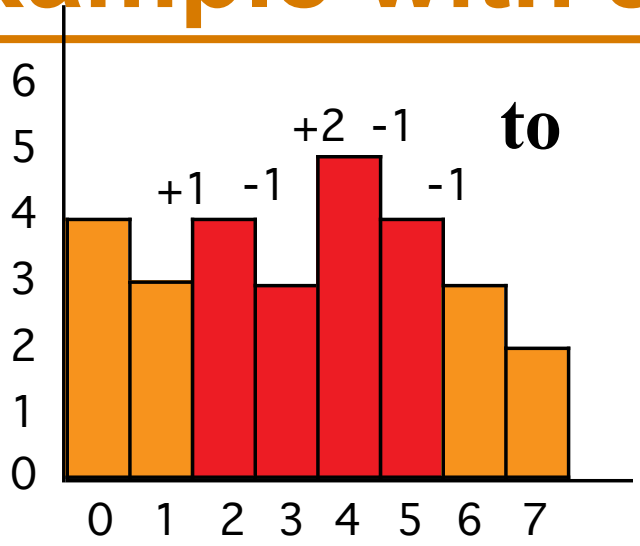


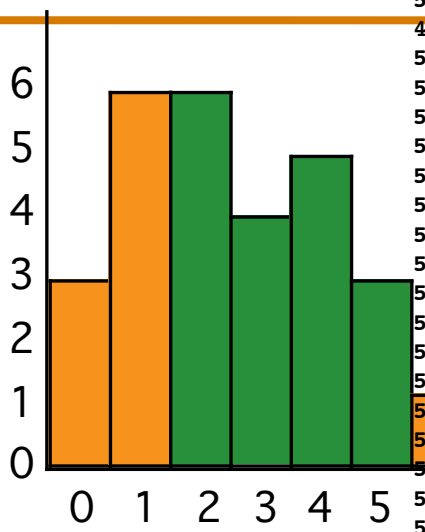
==>

$$\begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 16 \\ -6 \\ 6 \\ 2 \end{pmatrix}$$

$$\begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \\ 5 \\ 3 \end{pmatrix}$$

1D example with Jacobi

- **Copy** 

to 

$$\begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 16 \\ -6 \\ 6 \\ 2 \end{pmatrix}$$

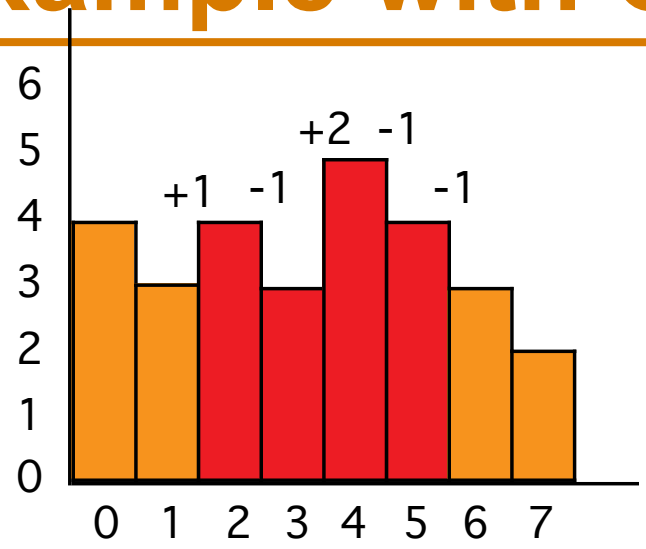
System
(I+A')x=b

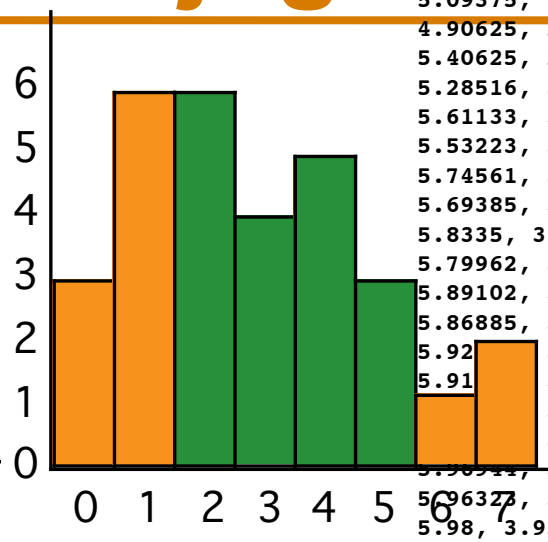
Iterations:
xn+1=b- A'xn

$$\begin{matrix} 4 \\ -1.5 \\ 1.5 \\ 0.5 \end{matrix} \quad +1/2 \quad \begin{matrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{matrix} \quad xn$$

```
4, -1.5, 1.5, 0.5,
3.25, 1.25, 1, 1.25,
4.625, 0.625, 2.75, 1,
4.3125, 2.1875, 2.3125, 1.875,
5.09375, 1.8125, 3.53125, 1.65625,
4.90625, 2.8125, 3.23438, 2.26562,
5.40625, 2.57031, 4.03906, 2.11719,
5.28516, 3.22266, 3.84375, 2.51953,
5.61133, 3.06445, 4.37109, 2.42188,
5.53223, 3.49121, 4.24316, 2.68555,
5.74561, 3.3877, 4.58838, 2.62158,
5.69385, 3.66699, 4.50464, 2.79419,
5.8335, 3.59924, 4.73059, 2.75232,
5.79962, 3.78204, 4.67578, 2.8653,
5.89102, 3.7377, 4.82367, 2.83789,
5.86885, 3.85735, 4.7878, 2.91183,
5.92867, 3.82832, 4.88459, 2.8939,
5.91416, 3.90663, 4.86111, 2.9423,
5.95332, 3.88764, 4.92446, 2.93056,
5.94382, 3.93889, 4.9091, 2.96223,
5.96511, 3.92646, 4.95056, 2.95455,
5.96327, 3.96, 4.9405, 2.97528,
5.98, 3.95187, 4.96764, 2.97025,
5.97593, 3.97382, 4.96106, 2.98382,
5.98691, 3.9685, 4.97882, 2.98053,
5.98425, 3.98287, 4.97451, 2.98941,
5.99143, 3.97938, 4.98614, 2.98726,
5.98969, 3.98879, 4.98332, 2.99307,
5.99439, 3.9865, 4.99093, 2.99166,
5.99325, 3.99266, 4.98908, 2.99546,
5.99633, 3.99117, 4.99406, 2.99454,
5.99558, 3.9952, 4.99285, 2.99703,
5.9976, 3.99422, 4.99611, 2.99643,
5.99711, 3.99686, 4.99532, 2.99806,
5.99843, 3.99622, 4.99746, 2.99766,
5.99811, 3.99794, 4.99694, 2.99873,
5.99897, 3.99752, 4.99834, 2.99847,
5.99876, 3.99865, 4.998, 2.99917,
5.99933, 3.99838, 4.99891, 2.999,
5.99919, 3.99912, 4.99869, 2.99946,
5.99956, 3.99894, 4.99929, 2.99934,
5.99947, 3.99942, 4.99914, 2.99964,
5.99971, 3.99931, 4.99953, 2.99957,
5.99965, 3.99962, 4.99944, 2.99977,
5.99981, 3.99955, 4.99969, 2.99972,
5.99977, 3.99975, 4.99963, 2.99985,
5.99988, 3.9997, 4.9998, 2.99982,
5.99985, 3.99984, 4.99976, 2.9999,
```

1D example with Conjugate





Jacobi:

```
4, -1.5, 1.5, 0.5,
3.25, 1.25, 1, 1.25,
4.625, 0.625, 2.75, 1,
4.3125, 2.1875, 2.3125, 1.875,
5.09375, 1.8125, 3.53125, 1.65625,
4.90625, 2.8125, 3.23438, 2.26562,
5.40625, 2.57031, 4.03906, 2.11719,
5.28516, 3.22266, 3.84375, 2.51953,
5.61133, 3.06445, 4.37109, 2.42188,
5.53223, 3.49121, 4.24316, 2.68555,
5.74561, 3.3877, 4.58838, 2.62158,
5.69385, 3.66699, 4.50464, 2.79419,
5.8335, 3.59924, 4.73059, 2.75232,
5.79962, 3.78204, 4.67578, 2.8653,
5.89102, 3.7377, 4.82367, 2.83789,
5.86885, 3.85735, 4.7878, 2.91183,
5.92 3.82832, 4.88459, 2.8939,
5.91 3.90663, 4.86111, 2.9423,
3.88764, 4.92446, 2.93056,
3.93889, 4.9091, 2.96223,
3.92646, 4.95056, 2.95455,
3.96, 4.9405, 2.97528,
5.98, 3.95187, 4.96764, 2.97025,
5.97593, 3.97382, 4.96106, 2.98382,
5.98691, 3.9685, 4.97882, 2.98053,
5.98425, 3.98287, 4.97451, 2.98941,
5.99143, 3.97938, 4.98614, 2.98726,
5.98969, 3.98879, 4.98332, 2.99307,
5.99439, 3.9865, 4.99093, 2.99166,
5.99325, 3.99266, 4.98908, 2.99546,
5.99633, 3.99117, 4.99406, 2.99454,
5.99558, 3.9952, 4.99285, 2.99703,
5.9976, 3.99422, 4.99611, 2.99643,
5.99711, 3.99686, 4.99532, 2.99806,
5.99843, 3.99622, 4.99746, 2.99766,
5.99811, 3.99794, 4.99694, 2.99873,
5.99897, 3.99752, 4.99834, 2.99847,
5.99876, 3.99865, 4.998, 2.99917,
5.99933, 3.99838, 4.99891, 2.999,
5.99919, 3.99912, 4.99869, 2.99946,
5.99956, 3.99894, 4.99929, 2.99934,
5.99947, 3.99942, 4.99914, 2.99964,
5.99971, 3.99931, 4.99953, 2.99957,
5.99965, 3.99962, 4.99944, 2.99977,
5.99981, 3.99955, 4.99969, 2.99972,
5.99977, 3.99975, 4.99963, 2.99985,
5.99988, 3.9997, 4.9998, 2.99982,
5.99985, 3.99984, 4.99976, 2.9999,
```

Conjugate gradient:

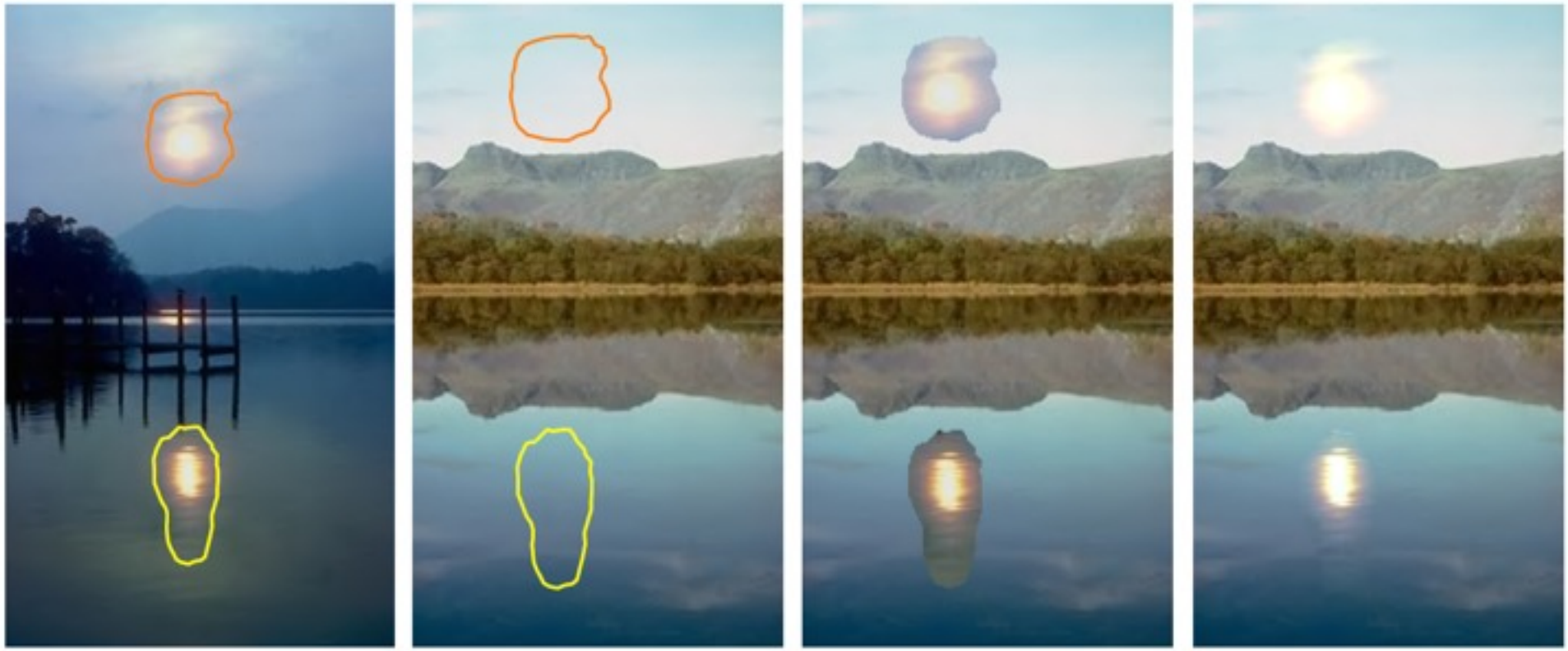
2.9381	-1.1018	1.1018	0.3673
5.2027	1.5933	1.6370	1.8617
6.1724	3.9337	4.3370	2.0983
6.0000	4.0000	5.0000	3.0000

More about all this next time

Recap

- **Poisson image cloning: paste gradient, enforce boundary condition**
- **Variational formulation** $\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}.$
- **Discretize variational version, leads to big but sparse linear system**
- **There are smart iterative technique to solve it without full Gaussian elimination**
  - In fact without ever storing the full matrix

Questions?



sources | destinations | cloning | seamless cloning

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.

Manipulate the gradient

- **Mix gradients of g & f: take the max**

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.

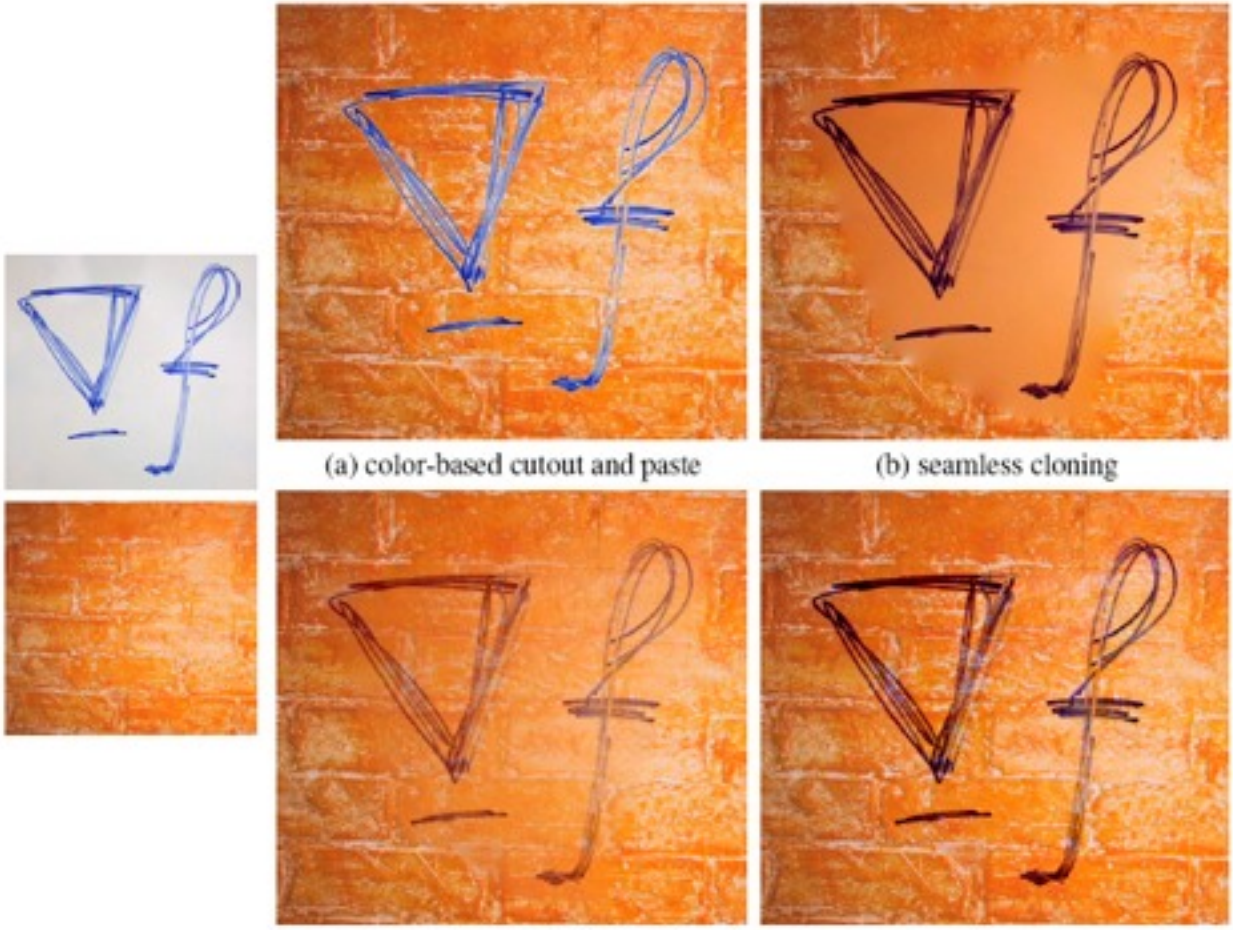
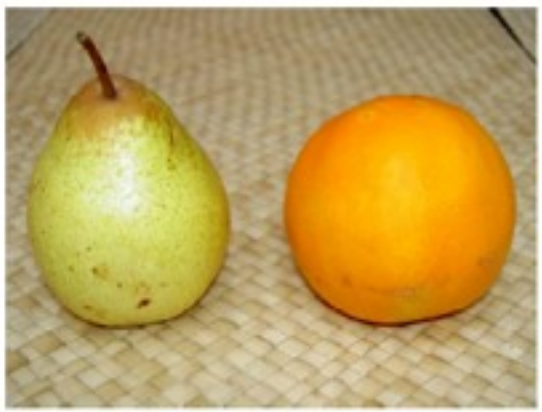


Figure 6: **Inserting objects with holes**. (a) The classic method, color-based selection and alpha masking might be time consuming and often leaves an undesirable halo; (b-c) seamless cloning, even averaged with the original image, is not effective; (d) mixed seamless cloning based on a loose selection proves effective.



swapped textures

source

destination

Figure 7: **Inserting transparent objects**. Mixed seamless cloning facilitates the transfer of partly transparent objects, such as the rainbow in this example. The non-linear mixing of gradient fields picks out whichever of source or destination structure is the more salient at each location.

Questions?

Issues with Poisson cloning

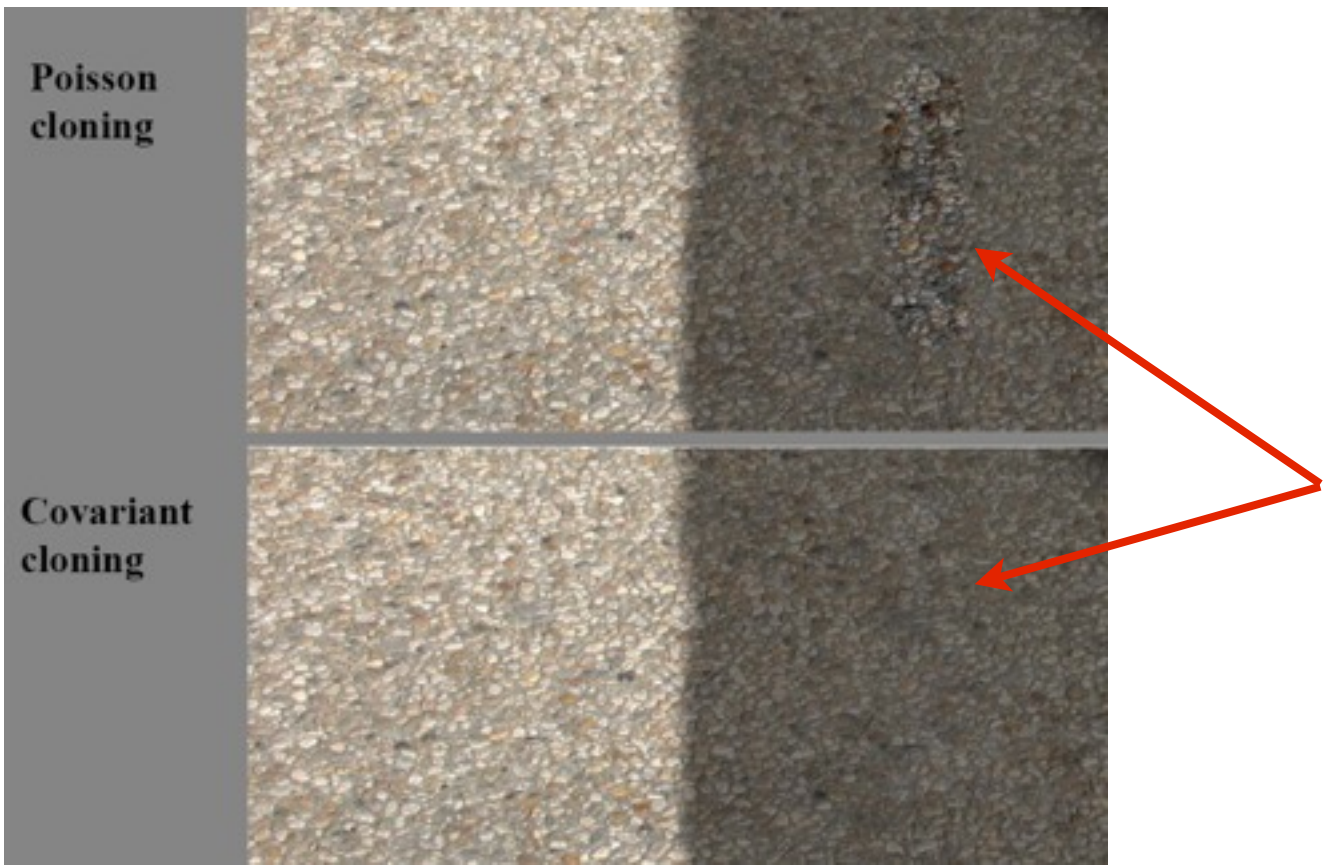
- **Colors**
- **Contrast**
- **The backgrounds in f & g should be similar**

Improvement: local contrast

- **Use the log**
- **Or use covariant derivatives (next slides)**

Covariant derivatives & Photoshop

- **Photoshop Healing brush**
- **Developed independently from Poisson editing by Todor Georgiev (Adobe)**



From Todor Georgiev's slides http://photo.csail.mit.edu/posters/todor_slides.pdf

Seamless Image Stitching in the Gradient Domain

- Anat Levin, Assaf Zomet, Shmuel Peleg, and Yair Weiss
  http://www.cs.huji.ac.il/~alevin/papers/eccv04-blending.pdf
  http://eprints.pascal-network.org/archive/00001062/01/tips05-blending.pdf
- **Various strategies (optimal cut, feathering)**

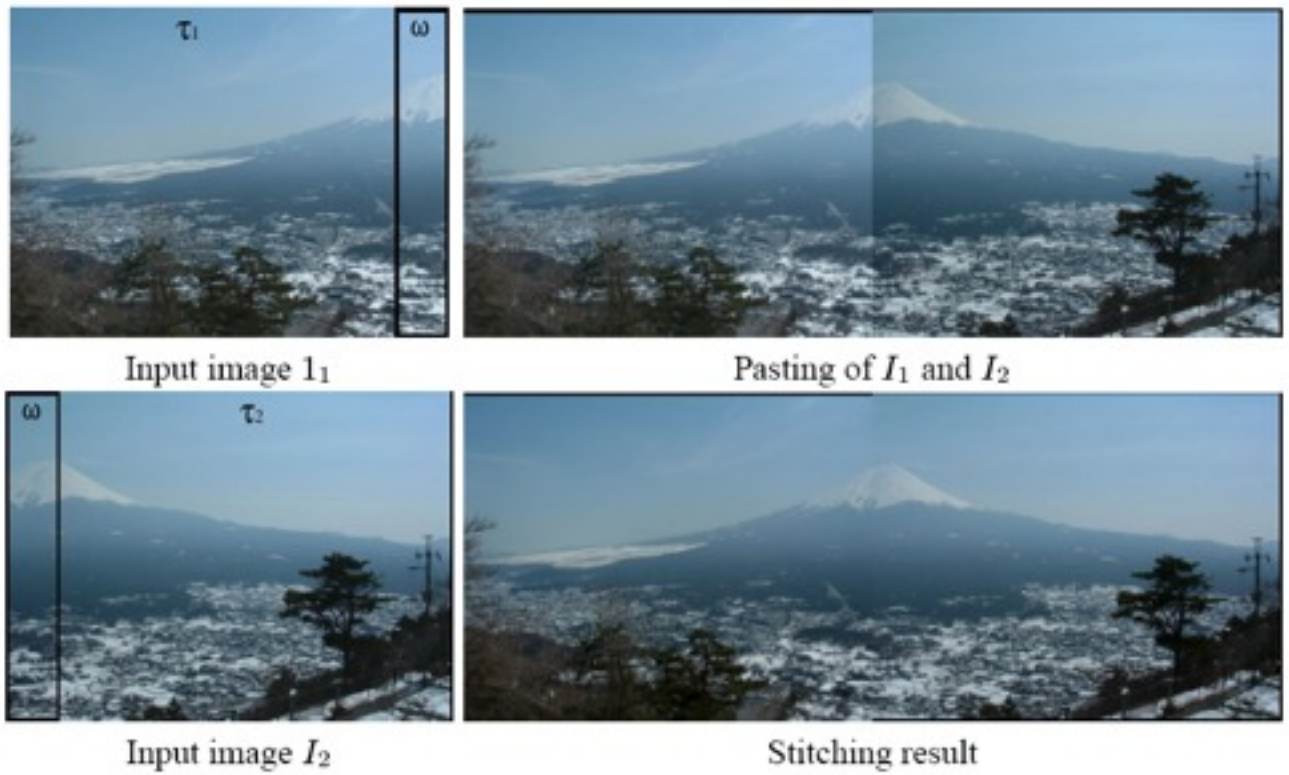


Fig. 1. Image stitching. On the left are the input images. ω is the overlap region. On top right is a simple pasting of the input images. On the bottom right is the result of the GIST1 algorithm.

Photomontage

- http://grail.cs.washington.edu/projects/photomontage/photomontage.pdf

Figure 6 We use a set of portraits (first row) to mix and match facial features, to either improve a portrait, or create entirely new people. The faces are first hand-aligned, for example, to place all the noses in the same location. In the first two images in the second row, we replace the closed eyes of a portrait with the open eyes of another. The user paints strokes with the *designated source* objective to specify desired features. Next, we create a fictional person by combining three source portraits. Gradient-domain fusion is used to smooth out skin tone differences. Finally, we show two additional mixed portraits.

Elder's edge representation

- http://elderlab.yorku.ca/~elder/publications/journals/ElderPAMI01.pdf

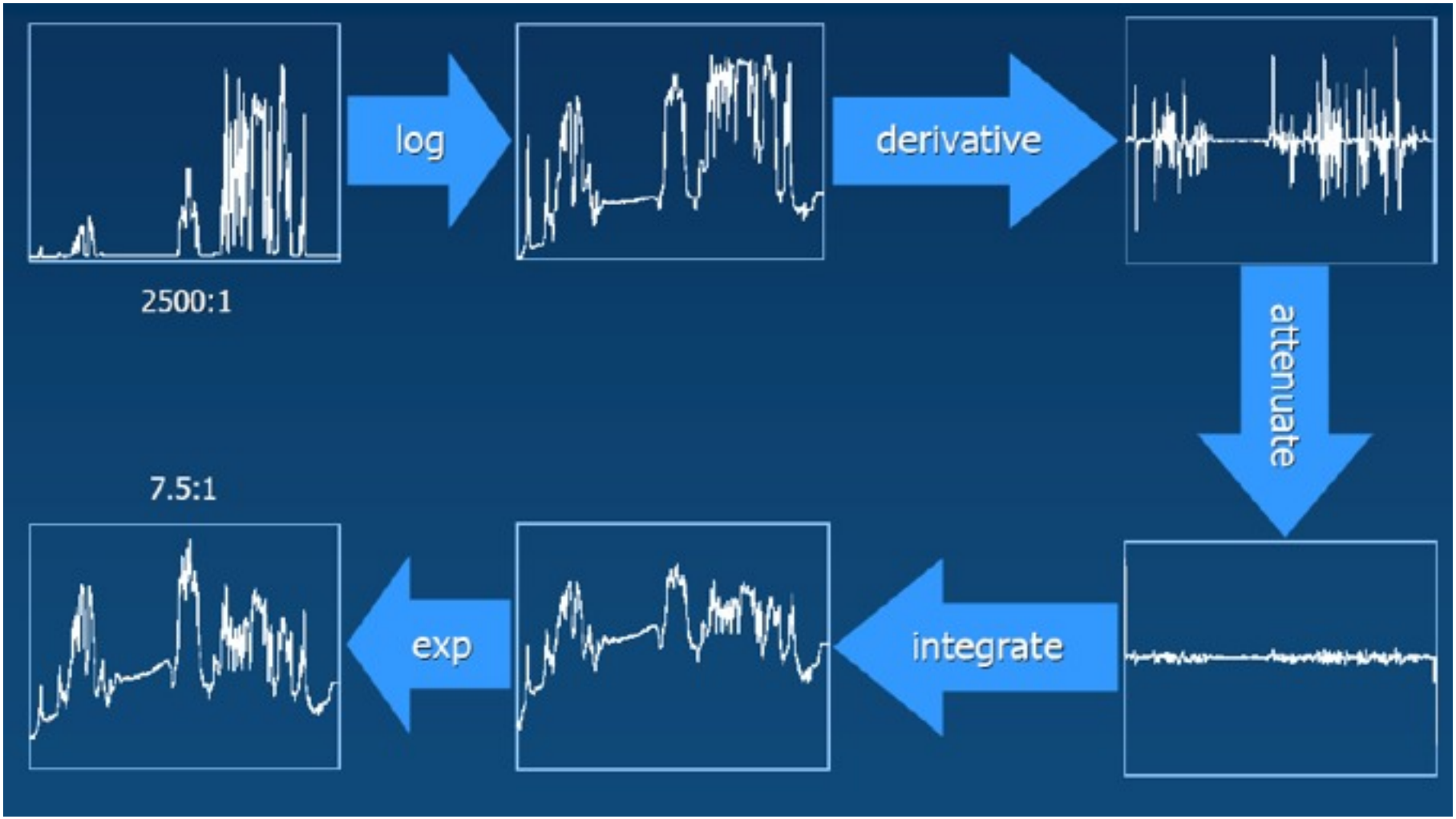
Reduce big gradients

- **Dynamic range compression**
- **See Fattal et al. 2002**

Figure 10: **Local illumination changes**. Applying an appropriate non-linear transformation to the gradient field inside the selection and then integrating back with a Poisson solver, modifies locally the apparent illumination of an image. This is useful to highlight under-exposed foreground objects or to reduce specular reflections.

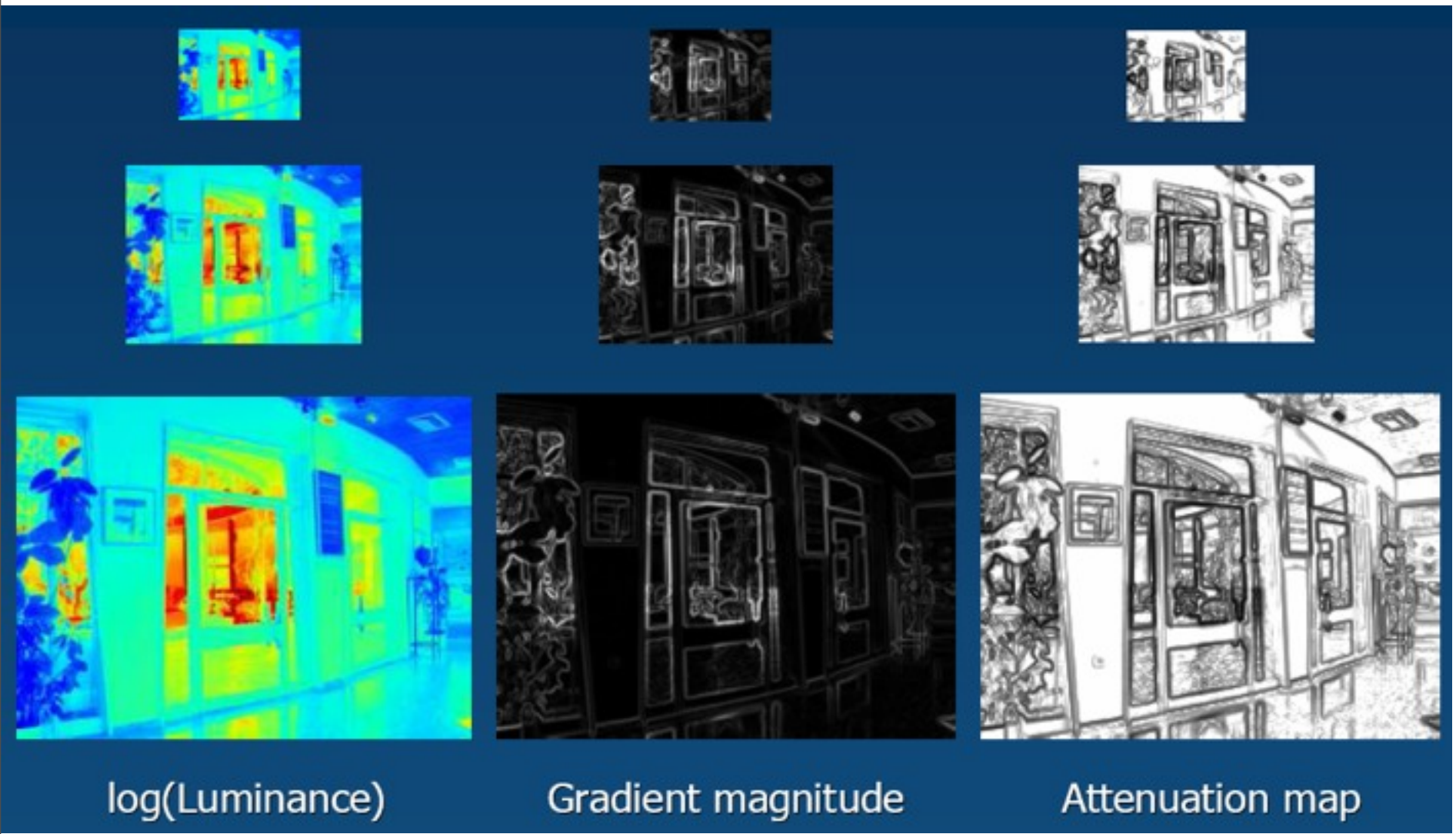
Gradient tone mapping

- **Fattal et al. Siggraph 2002**



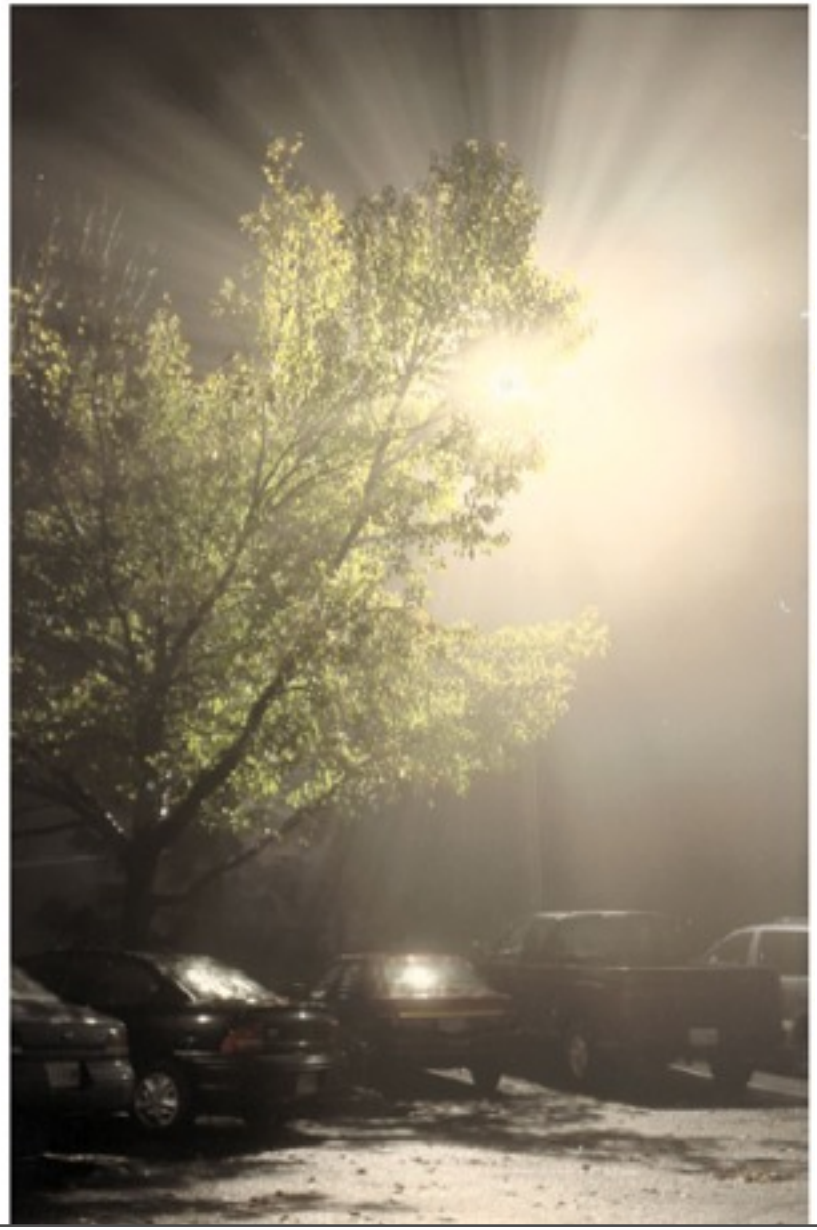
Slide from Siggraph 2005 by Raskar (Graphs by Fattal et al.)

Gradient attenuation



From Fattal et al.

Fattal et al. Gradient tone mapping



Gradient tone mapping

- **Socolinsky, D. *Dynamic Range Constraints in Image Fusion and Visualization* , in Proceedings of Signal and Image Processing 2000, Las Vegas, November 2000.**

Fig. 1. (a) Mediastinal window of thoracic CT scan. (b) Lung window of thoracic CT scan. (c) Clipped solution of equation (2) for the fusion of (a) and (b). (d) Linearly scaled solution of (2) for the fusion of (a) and (b). (e) Solution of equation (6) for the fusion of(a) and (b).

Gradient tone mapping

- **Socolinsky, D. *Dynamic Range Constraints in Image Fusion and Visualization* , in Proceedings of Signal and Image Processing 2000.**

Fig. 4. Left: average of images in figure 2. Middle: rendering of the sum of the images in figure 2 through adaptive histogram compression. Right: fusion of images in figure 2 using the obstacle method.

- **Socolinsky, D. and Wolff, L.B., *A new paradigm for multispectral image visualization and data fusion*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Fort Collins, June 1999.**

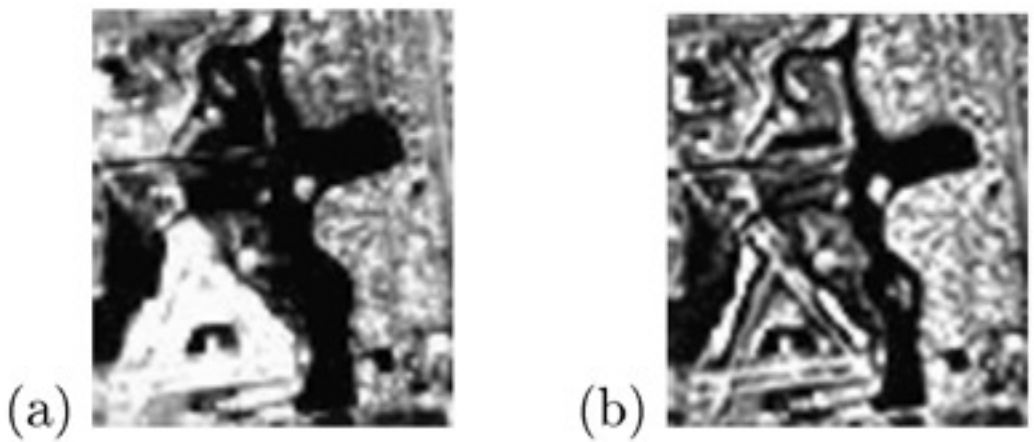
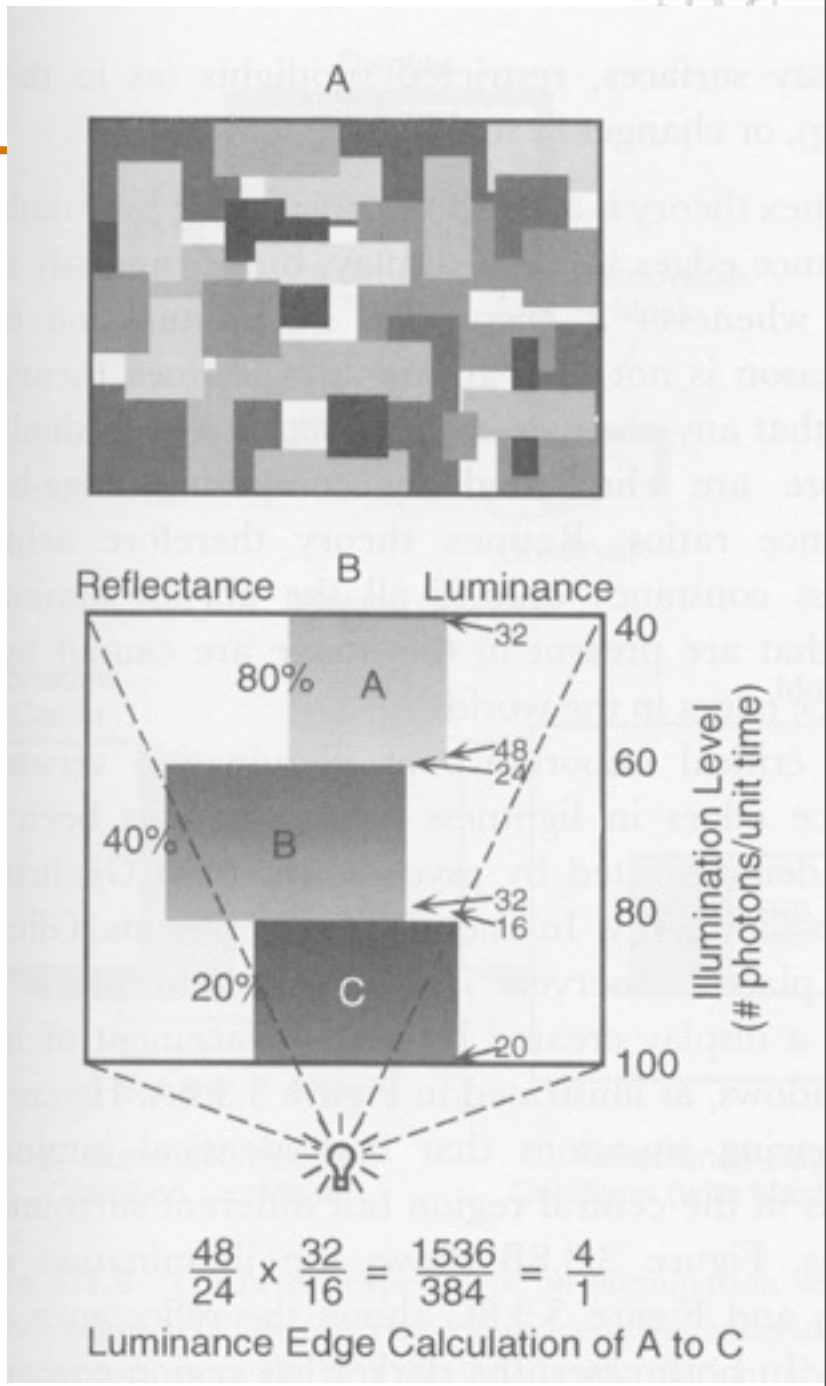


Figure 4: (a) Grayscale version of 9-band image computed through PCA. (b) Grayscale version of the same image computed through our algorithm.

Retinex

- **Land, Land and McCann (inventor/founder of polaroid)**
- **Theory of lightness perception (albedo vs. illumination)**
- **Strong gradients come from albedo, illumination is smooth**



Questions?

Color2gray

- **Use Lab gradient to create grayscale images**

Color2Gray: Salience-Preserving Color Removal

Amy A. Gooch Sven C. Olsen Jack Tumblin Bruce Gooch

Northwestern University *

Figure 1: A color image (Left) often reveals important visual details missing from a luminance-only image (Middle). Our Color2Gray algorithm (Right) maps visible color changes to grayscale changes. *Image: Impressionist Sunrise by Claude Monet, courtesy of Artcyclopedia.com.*

Gradient camera?

- **Tumblin et al. CVPR 2005** http://www.cfar.umd.edu/~aagrawal/gradcam/gradcam.html

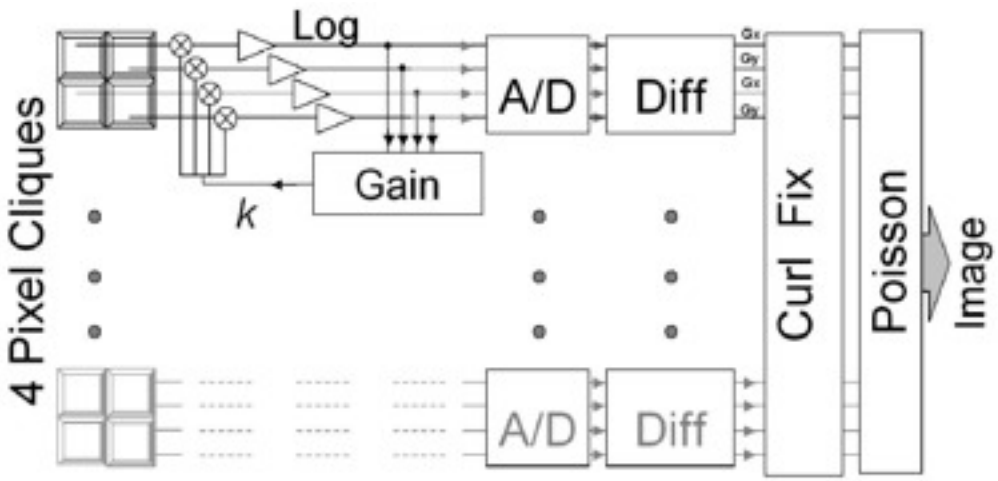


Figure 2. Log-gradient camera overview: intensity sensors organized into 4-pixel cliques share the same self-adjusting gain setting k, and send $\log(I_d)$ signals to A/D converter. Subtraction removes common-mode noise, and a linear 'curl fix' solver corrects saturated gradient values or 'dead' pixels, and a Poisson solver finds output values from gradients.

Poisson-ish mesh editing

- http://portal.acm.org/citation.cfm?id=1057432.1057456
- http://www.cad.zju.edu.cn/home/xudong/Projects/mesh_editing/main.htm
- http://people.csail.mit.edu/sumner/research/deftransfer/

Figure 1: An unknown mythical creature. Left: mesh components for merging and deformation (the arm), Right: final editing result.

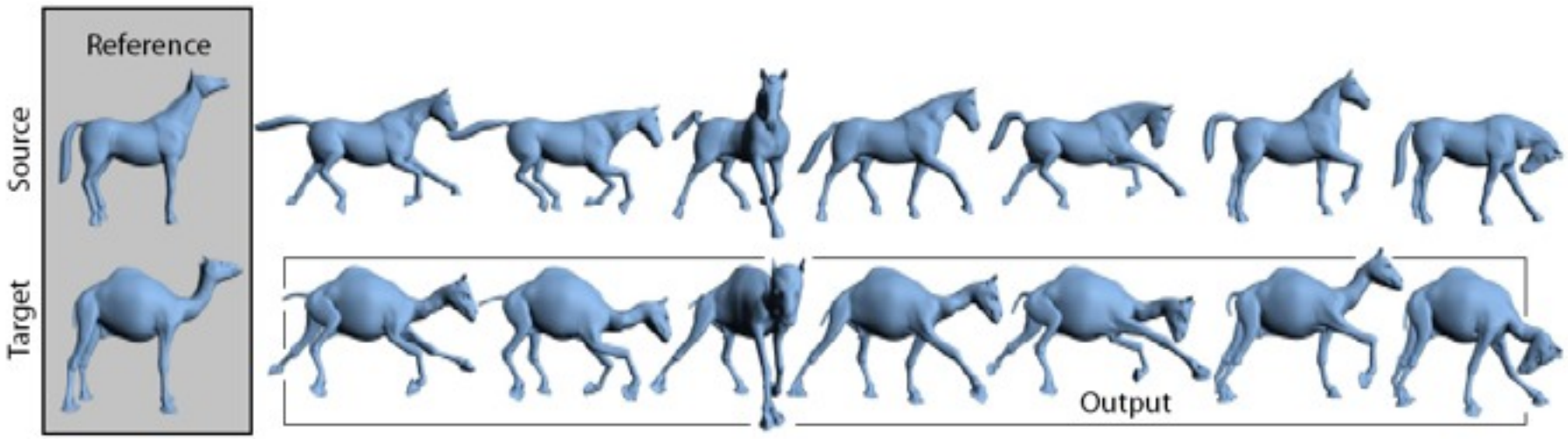


Figure 1: Deformation transfer copies the deformations exhibited by a source mesh onto a different target mesh. In this example, deformations of the reference horse mesh are transfered to the reference camel, generating seven new camel poses. Both gross skeletal changes as well as more subtle skin deformations are successfully reproduced.

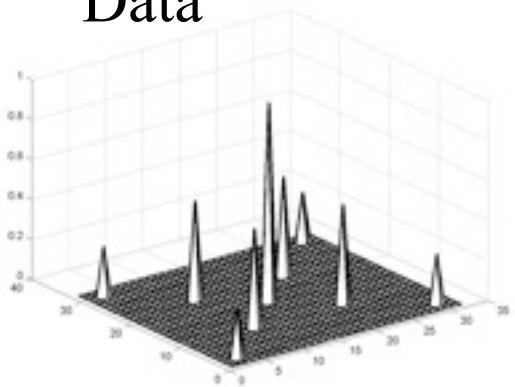
Questions?

Alternative to membrane

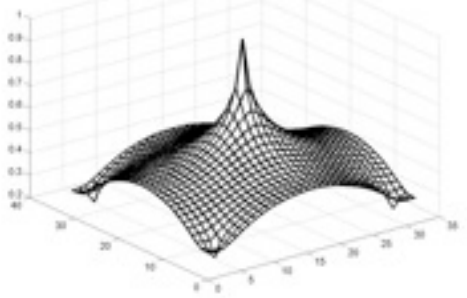
- **Thin plate:**
  minimize *second* derivative

$$\min_f \int\int f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2 \mathrm{dxdy}$$

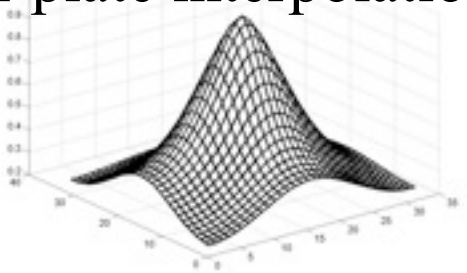
Data



Membrane interpolation



Thin-plate interpolation



Inpainting

- **More elaborate energy functional/PDEs**
- **http://www-mount.ee.umn.edu/~guille/inpainting.htm**

Chong color space

- log
- color opponents

Key references

- **Socolinsky, D. *Dynamic Range Constraints in Image Fusion and Visualization* 2000. http://www.equinoxsensors.com/news.html**
- **Elder, Image editing in the contour domain, 2001 http://elderlab.yorku.ca/~elder/publications/journals/ElderPAMI01.pdf**
- **Fattal et al. 2002 Gradient Domain HDR Compression http://www.cs.huji.ac.il/%7Edanix/hdr/**
- **Poisson Image Editing Perez et al. http://research.microsoft.com/vision/cambridge/papers/perez_siggraph03.pdf**
- **Covariant Derivatives and Vision, Todor Georgiev (Adobe Systems) ECCV 2006**

Poisson, Laplace, Lagrange, Fourier, Monge, Parseval

- **Fourier studied under Lagrange, Laplace & Monge, and Legendre & Poisson were around**
- **They all raised serious objections about Fourier's work on Trigomometric series**
- http://www.ece.umd.edu/~taylor/frame2.htm
- http://www.mathphysics.com/pde/history.html
- http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Fourier.html
- http://www.memagazine.org/contents/current/webonly/wex80905.html
- http://www.shsu.edu/~icc_cmf/bio/fourier.html
- http://en.wikipedia.org/wiki/Simeon_Poisson
- http://en.wikipedia.org/wiki/Pierre-Simon_Laplace
- http://en.wikipedia.org/wiki/Jean_Baptiste_Joseph_Fourier
- http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Parseval.html

Refs Laplace and Poisson

- http://www.ifm.liu.se/~boser/elma/Lect4.pdf
- http://farside.ph.utexas.edu/teaching/329/lectures/node74.html
- http://en.wikipedia.org/wiki/Poisson's_equation
- http://www.colorado.edu/engineering/CAS/courses.d/AFEM.d/AFEM.Ch03.d/AFEM.Ch03.pdf

Gradient image editing refs

- http://research.microsoft.com/vision/cambridge/papers/perez_siggraph03.pdf
- http://www.cs.huji.ac.il/~alevin/papers/eccv04-blending.pdf
- http://www.eg.org/EG/DL/WS/COMPAESTH/COMPAESTH05/075-081.pdf.abstract.pdf
- http://photo.csail.mit.edu/posters/Georgiev_Covariant.pdf
- **Covariant Derivatives and Vision, Todor Georgiev (Adobe Systems) ECCV 2006**
- http://www.mpi-sb.mpg.de/~hitoshi/research/image_restoration/index.shtml
- http://www.cs.tau.ac.il/~tommer/vidoegrad/
- http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=1467600
- http://grail.cs.washington.edu/projects/photomontage/
- http://www.cfar.umd.edu/~aagrawal/iccv05/surface_reconstruction.html
- http://www.merl.com/people/raskar/Flash05/
- http://research.microsoft.com/~carrot/new_page_1.htm
- http://www.idiom.com/~zilla/Work/scatteredInterpolation.pdf

Poisson image editing

- **Two aspects**
  - When the new gradient is conservative:
    Just membrane interpolation to ensure boundary condition
  - Otherwise: allows you to work with non-conservative vector fields and
- **Why is it good?**
  - More weight on high frequencies
    - Membrane tries to use low frequencies to match boundaries conditions
  - Manipulation of the gradient can be cool
    (e.g. max of the two gradients)
    - Manipulate local features (edge/gradient) and worry about global consistency later
- **Smart thing to do: work in log domain**
- **Limitations**
  - Color shift, contrast shift (depends strongly on the difference between the two respective backgrounds)