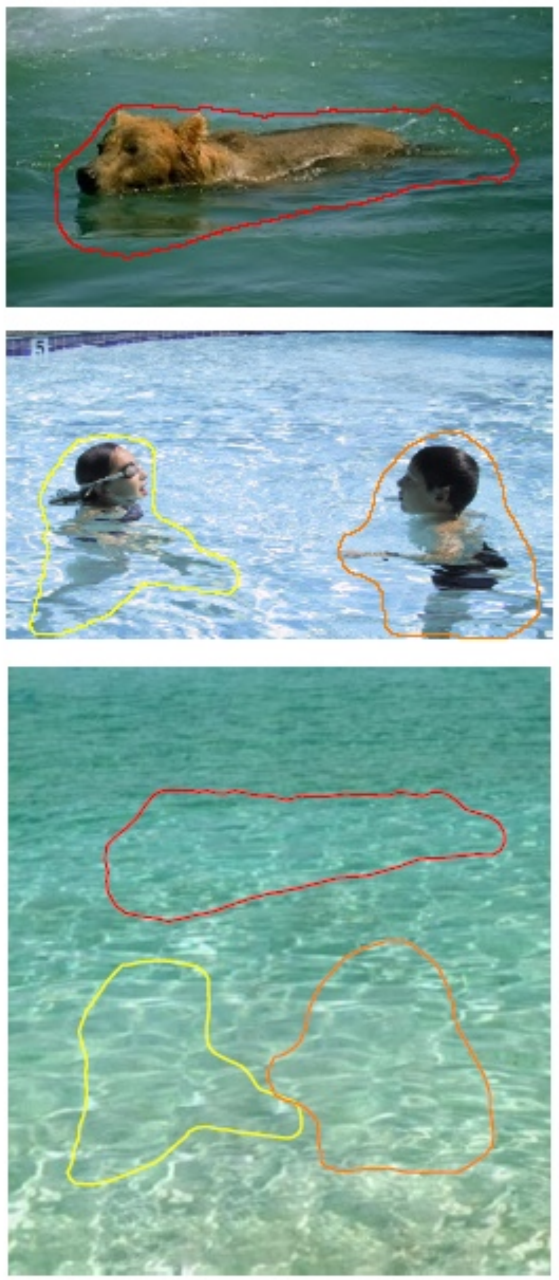
CS6640 Computational Photography

11. Gradient Domain Image Processing

Problems with direct copy/paste



sources/destinations

cloning

From Perez et al. 2003

Image gradient

- **Gradient: derivative of a function *R*^{n} → *R* (*n* = 2 for images)**

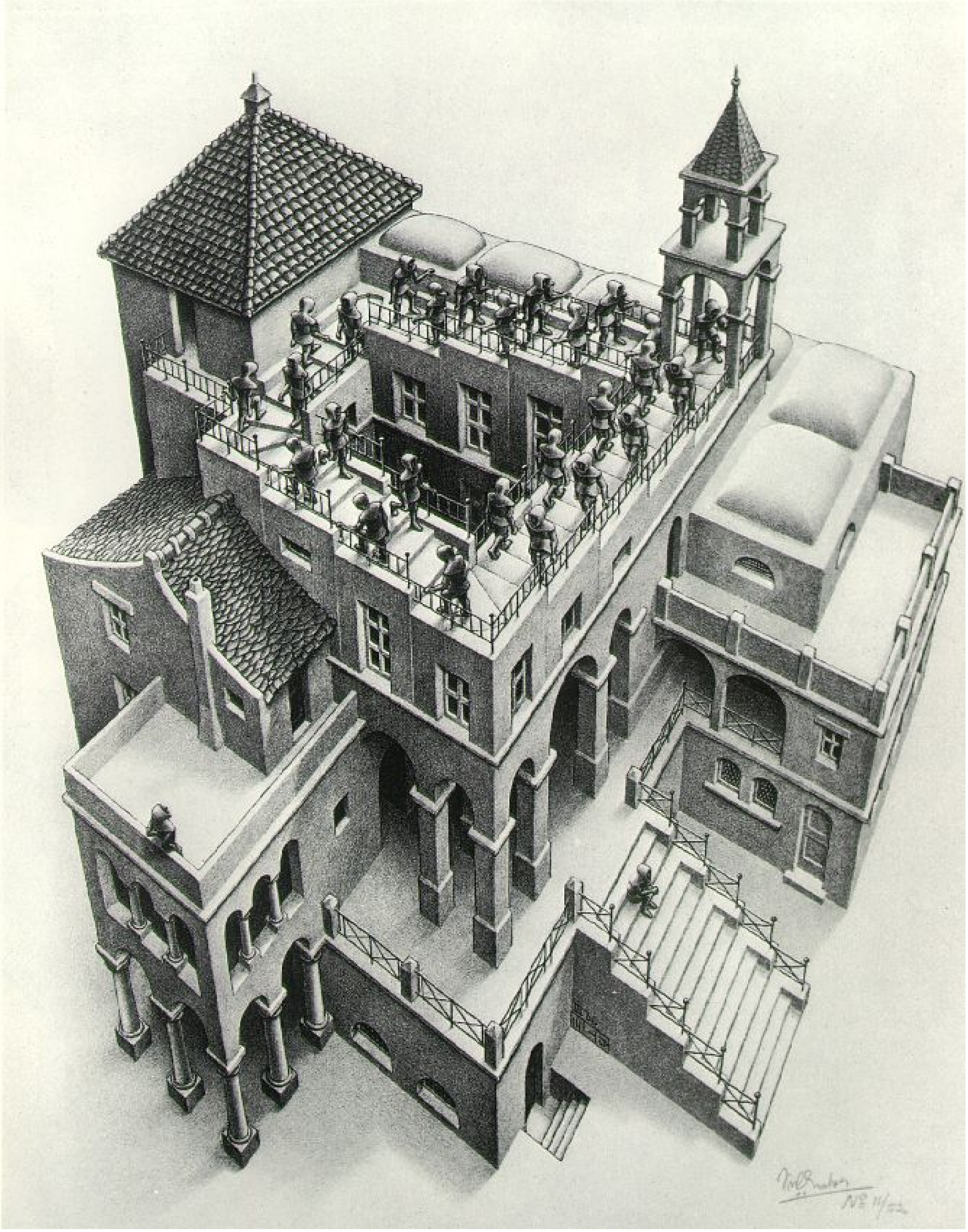
$$\nabla f = \begin{bmatrix} \frac{df}{dx} & \frac{df}{dy} \end{bmatrix} = \begin{bmatrix} f_x & f_y \end{bmatrix}$$

- **Note it turns a function *R*^{2} → *R* into a function *R*^{2} → *R*^{2}**
- **Most such functions are not the derivative of anything!**
- **How do you if some function *g* is the derivative of something?**

  in 2D, simple: mixed partials are equal (*g* is conservative)

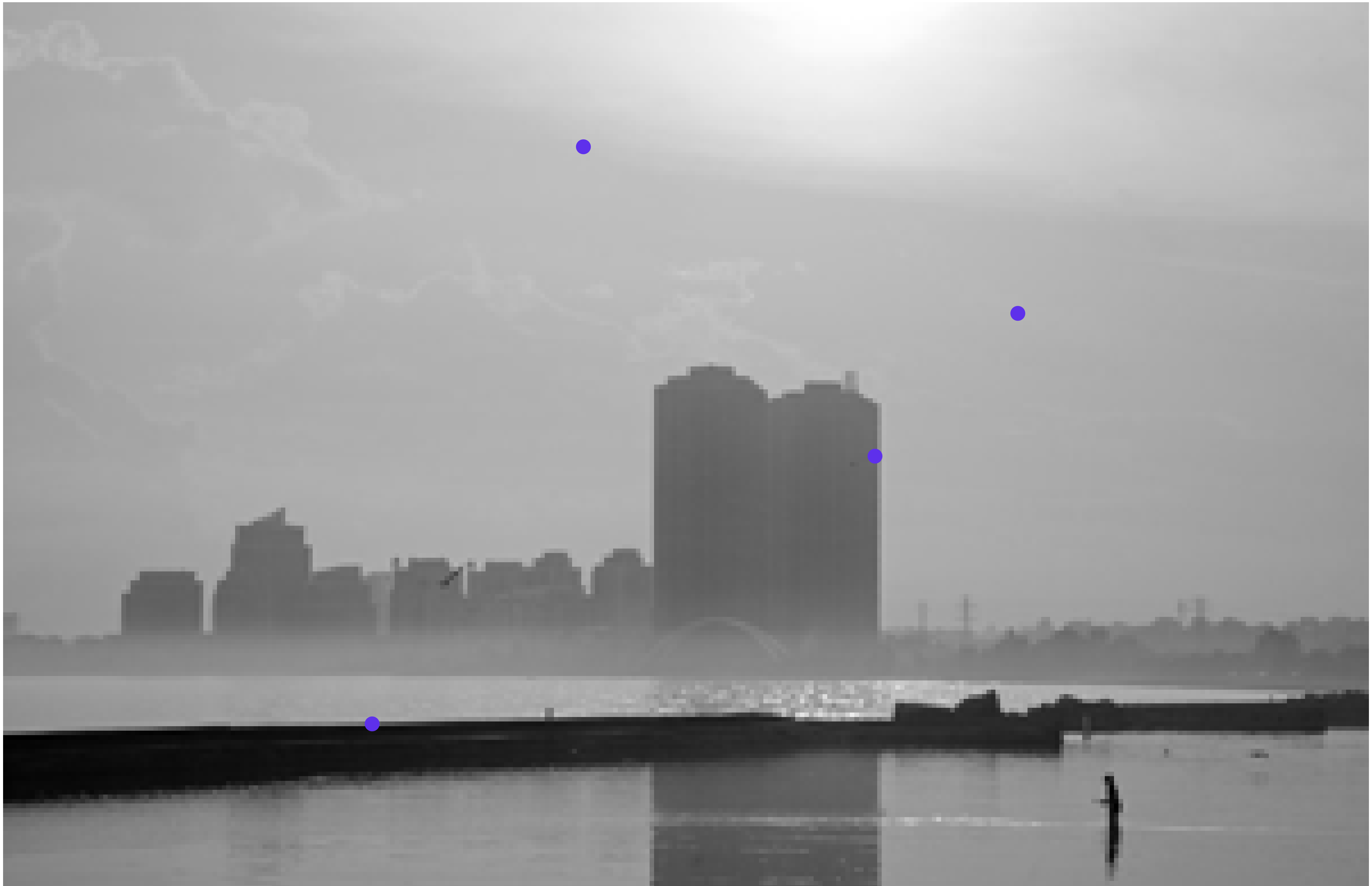
$g_x^y = g_y^x$ because $g = \nabla f$ and $f_{xy} = f_{yx}$

A nonconservative gradient?



M.C. Escher
Ascending and Descending
1960
Lithograph
35.5 x 28.5 cm

Gradient: intuition



slide by Frédo Durand, MIT

Gradient: intuition

slide by Frédo Durand, MIT

Gradient: intuition

slide by Frédo Durand, MIT

Gradient: intuition

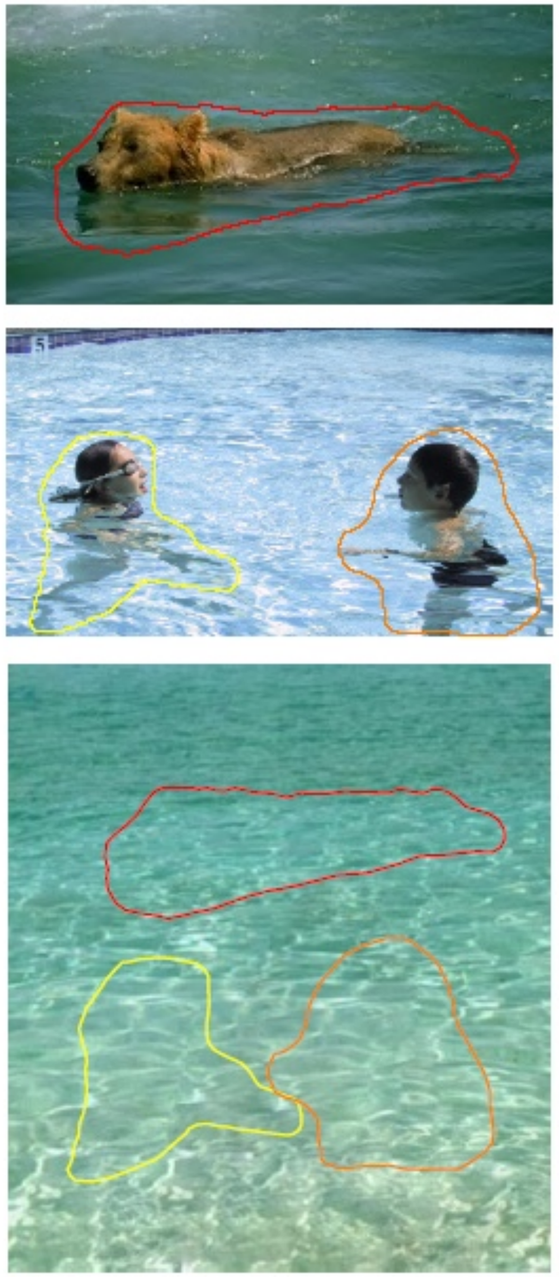
slide by Frédo Durand, MIT

Key gradient domain idea

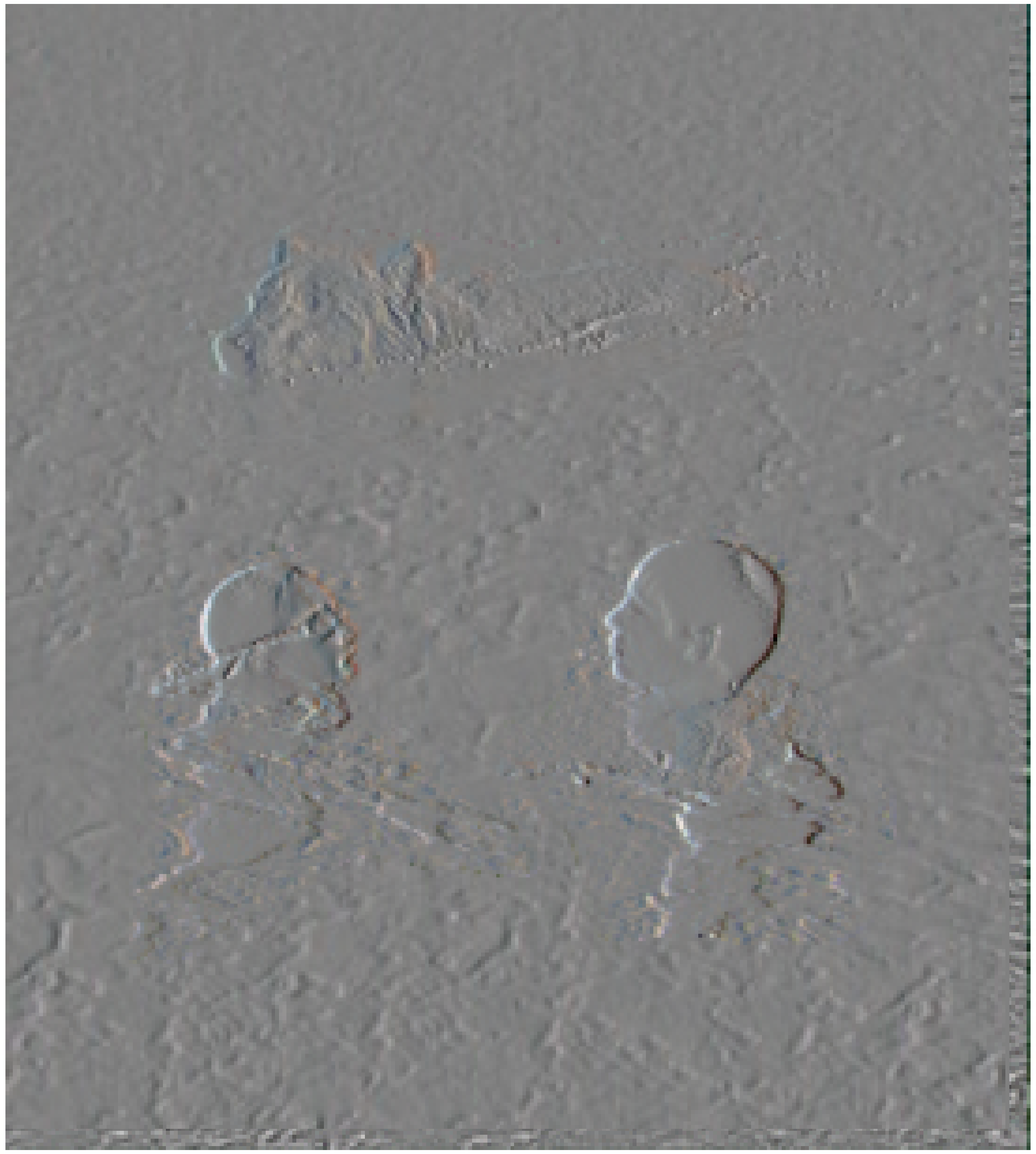
1. **Construct a vector field that we wish was the gradient of our output image**
2. **Look for an image that has that gradient**
3. **That won't work, so look for an image that has *approximately* the desired gradient**

Gradient domain image processing is all about clever choices for (1) and efficient algorithms for (3)

Solution: paste gradient



sources/destinations



hacky visualization of gradient

seamless cloning

Problem setup

Given desired gradient *g* on a domain *D*, and some constraints on a subset *B* of the domain

$$\vec{g} : D \to \mathbb{R}^2 \qquad B \subset D \qquad f^* : B \to \mathbb{R}$$

Find a function *f* on *D* that fits the constraints and has a gradient close to *g*

$$\min_f \|\nabla f - \vec{g}\|_2 \quad \text{subject to} \quad f\big|_B = f^*$$

Since the gradient is a linear operator, this is a (constrained) linear least squares problem.

Discretization

- **Of course images are made up of finitely many pixels**
- **Use discrete derivative [–1 1] to approximate gradient**
  there are other choices but this works fine here
- **Minimize sum-squared rather than integral-squared difference**
  sum is over edges joining neighboring pixels
- **Result is a matrix that maps *f* to its derivative**

$$\min_{\mathbf{f}} \|\mathbf{G}\mathbf{f} - \mathbf{g}\| \quad \text{subject to} \quad \Pi_B \mathbf{f} = \mathbf{f}^*$$

discrete gradient operator

pixels listed as vector

desired gradient

projection that selects pixels in B

constrained values

Handling constraints

- **To deal with constraints just leave out the constrained pixels**

$$f = \Pi_B^T \mathbf{f}^* + \Pi_{\bar{B}}^T \mathbf{f}'$$

$$\min_{\mathbf{f}'} \left\| G \left(\Pi_B^T \mathbf{f}^* + \Pi_{\bar{B}}^T \mathbf{f}' \right) - \mathbf{g} \right\|$$

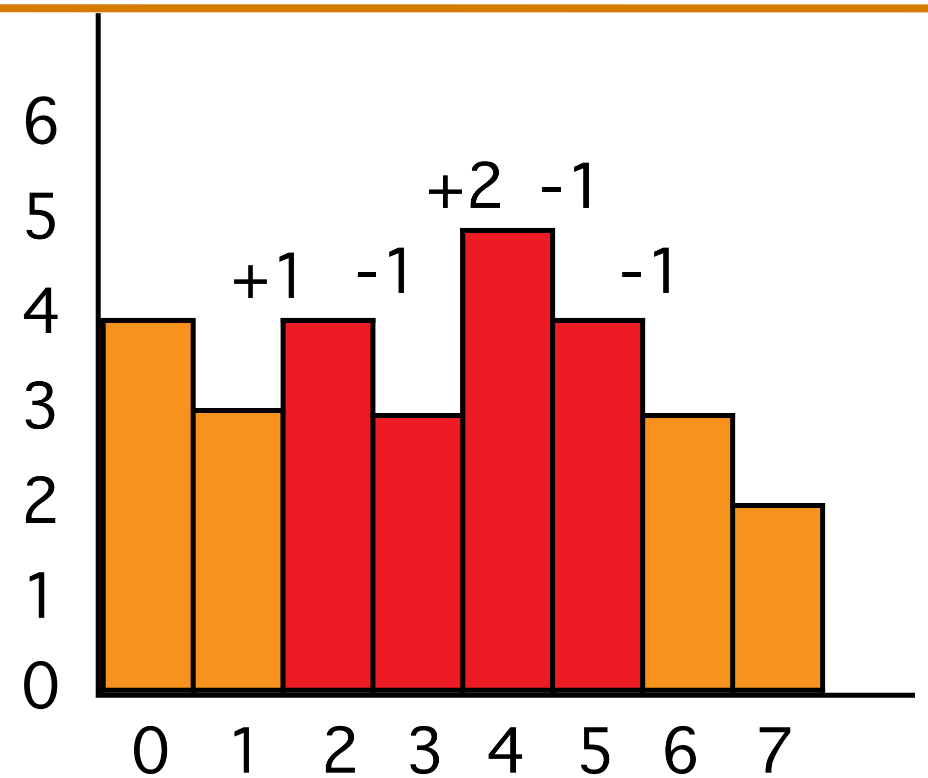
$$\min_{\mathbf{f}'} \left\| \underbrace{\left[G \Pi_{\bar{B}}^T \right]}_{\text{G without constrained columns}} \underbrace{\mathbf{f}'}_{\text{shorter vector of unknowns}} - \underbrace{\left[\mathbf{g} - \Pi_B^T \mathbf{f}^* \right]}_{\text{augmented right hand side}} \right\|$$

- **The result is an unconstrained problem to be solved for the unknown variables in f′**

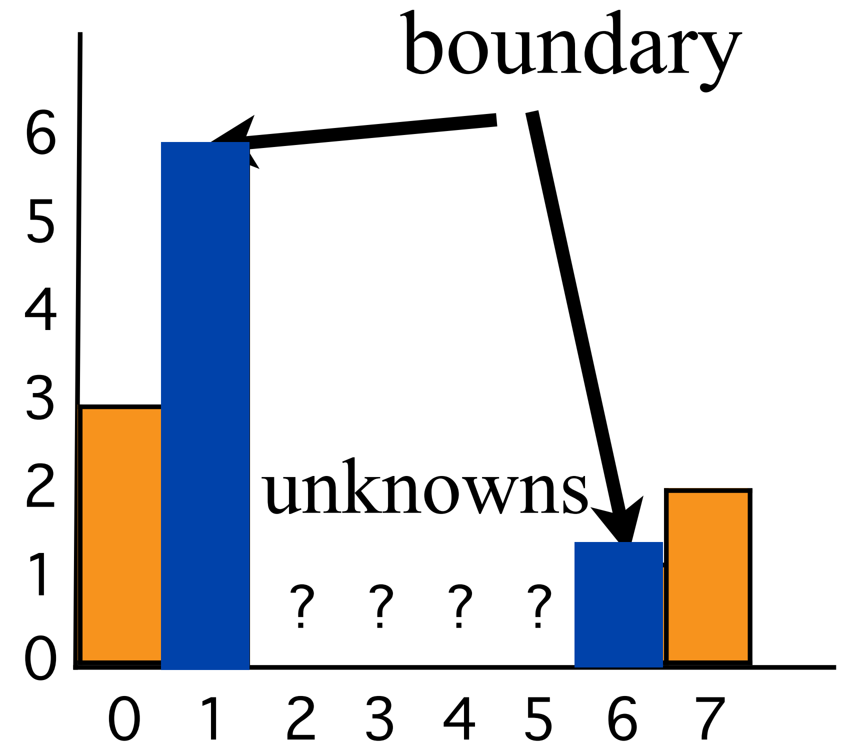
  one column per unknown pixel; one row per neighbor edge
  (any zero rows can be left out)

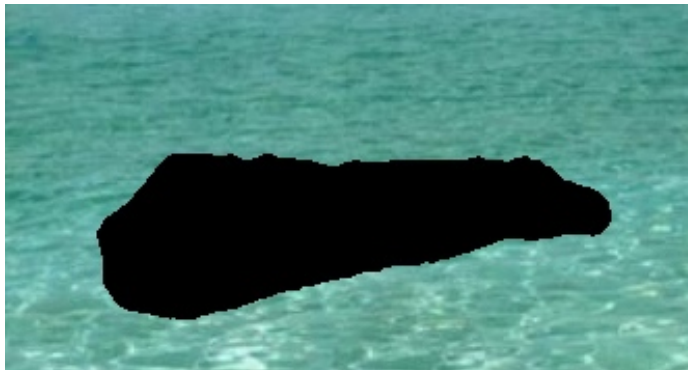
Discrete 1D example: minimization

- **Copy**



to



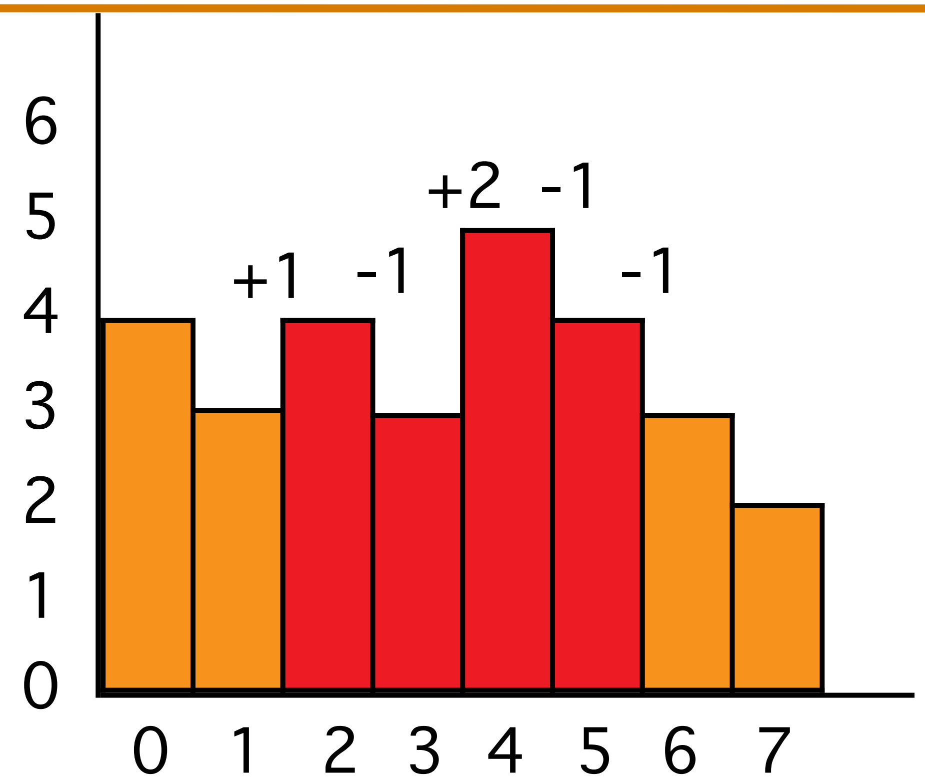


orange: pixel outside the mask
red: source pixel to be pasted
blue: boundary conditions (in background)

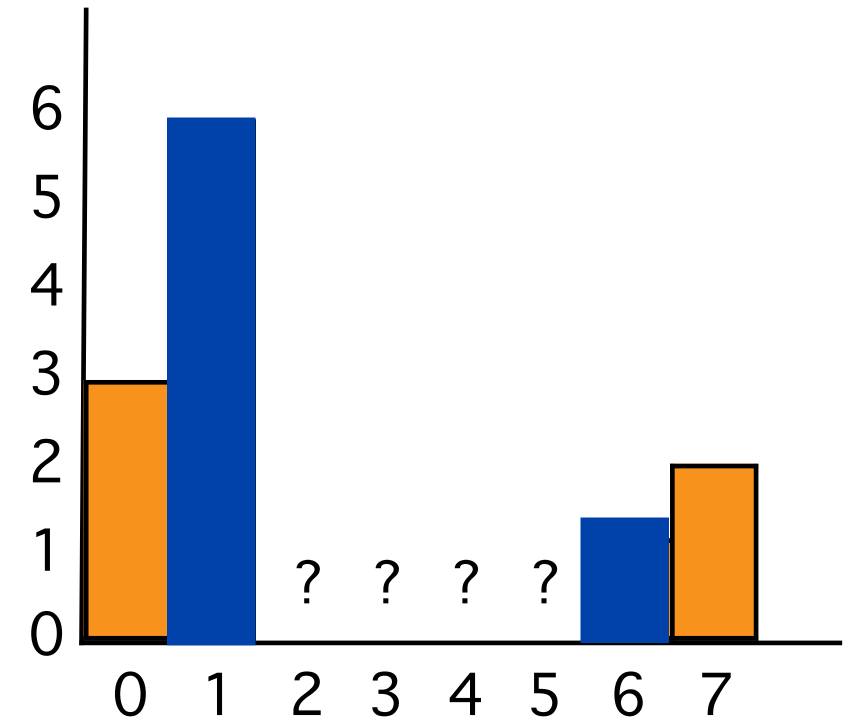
slide by Frédo Durand, MIT

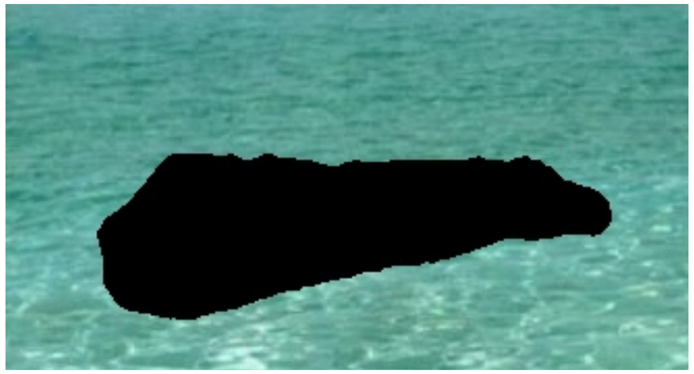
Discrete 1D example: minimization

- **Copy**



to

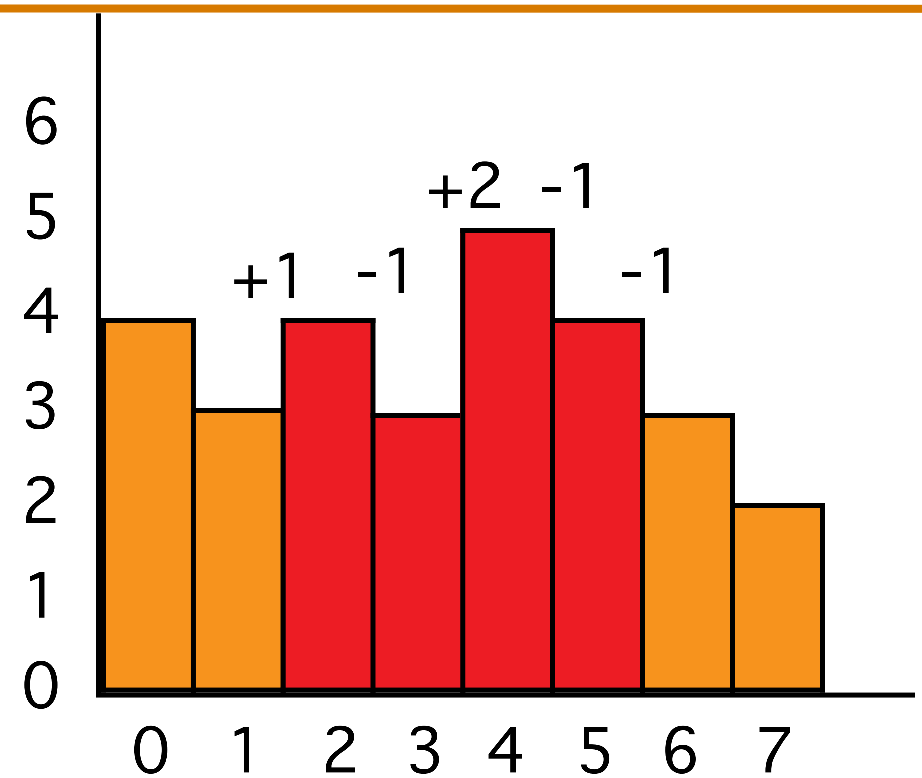




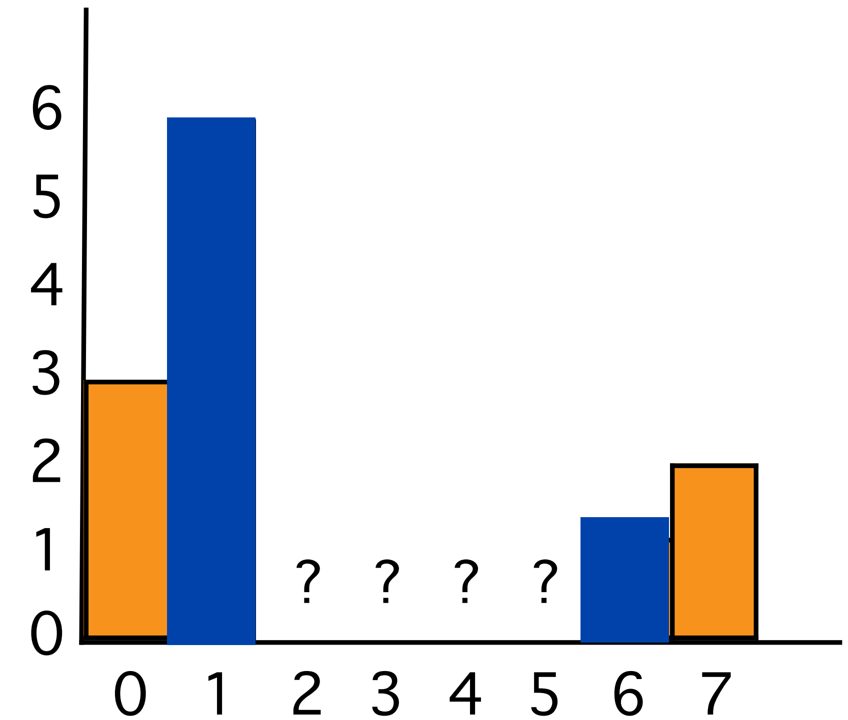
slide by Frédo Durand, MIT

Discrete 1D example: minimization

- **Copy**



to



Min $[(f_2-f_1)-1]^2$

$+ [(f_3-f_2)-(-1)]^2$

$+ [(f_4-f_3)-2]^2$

$+ [(f_5-f_4)-(-1)]^2$

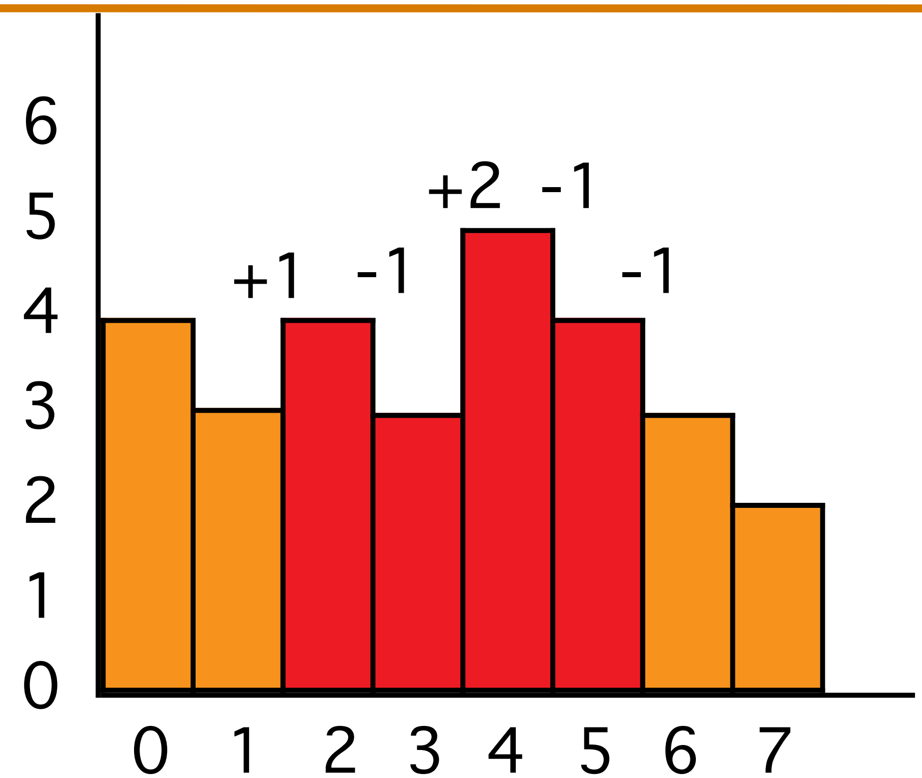
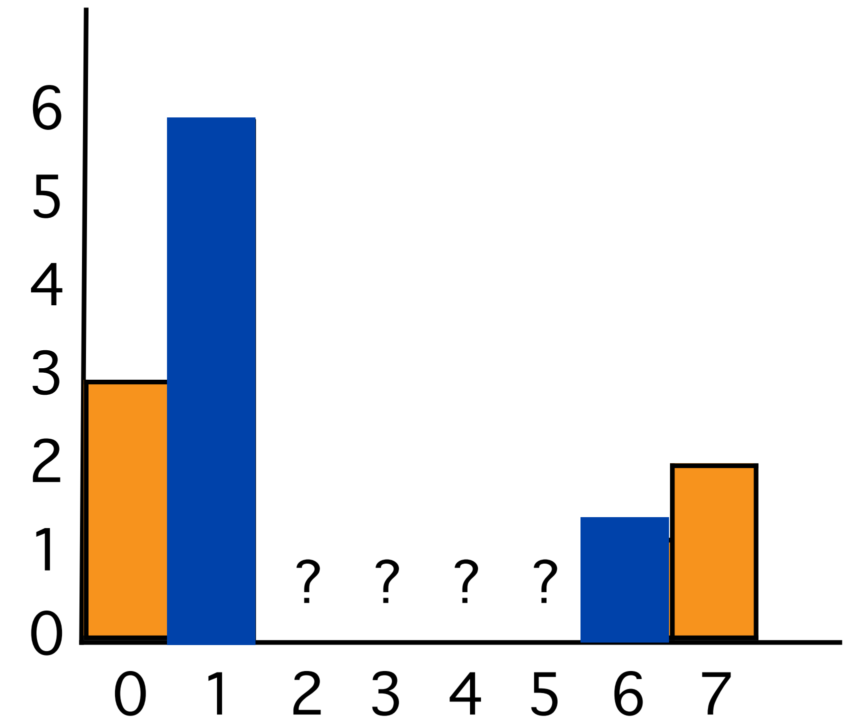
$+ [(f_6-f_5)-(-1)]^2$

With

$f_1=6$

$f_6=1$

1D example: minimization

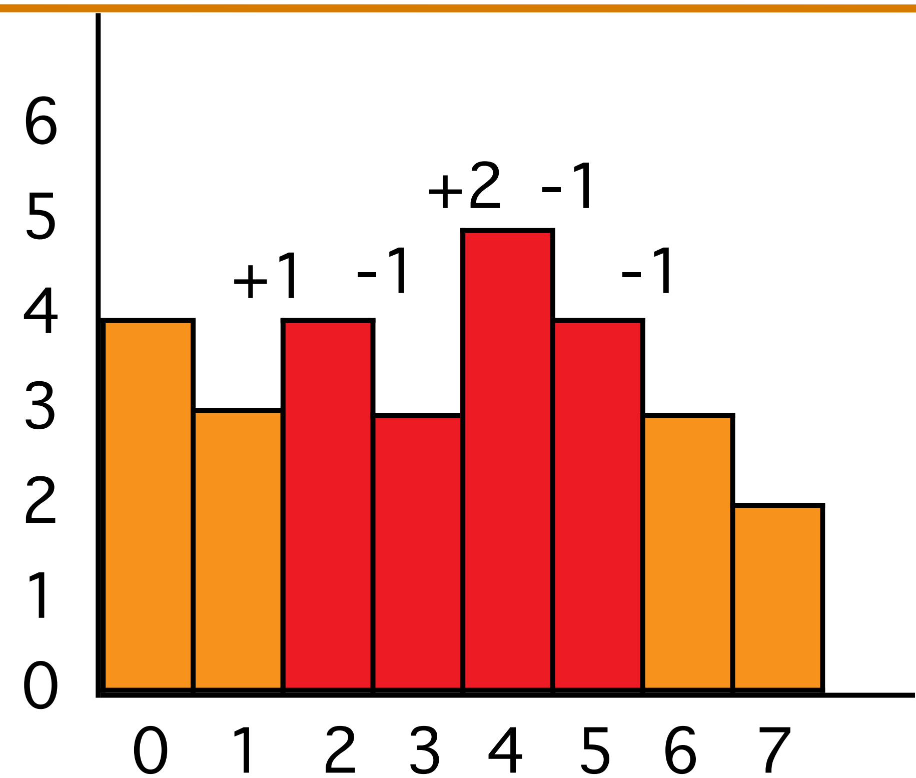
- **Copy**  **to** 

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

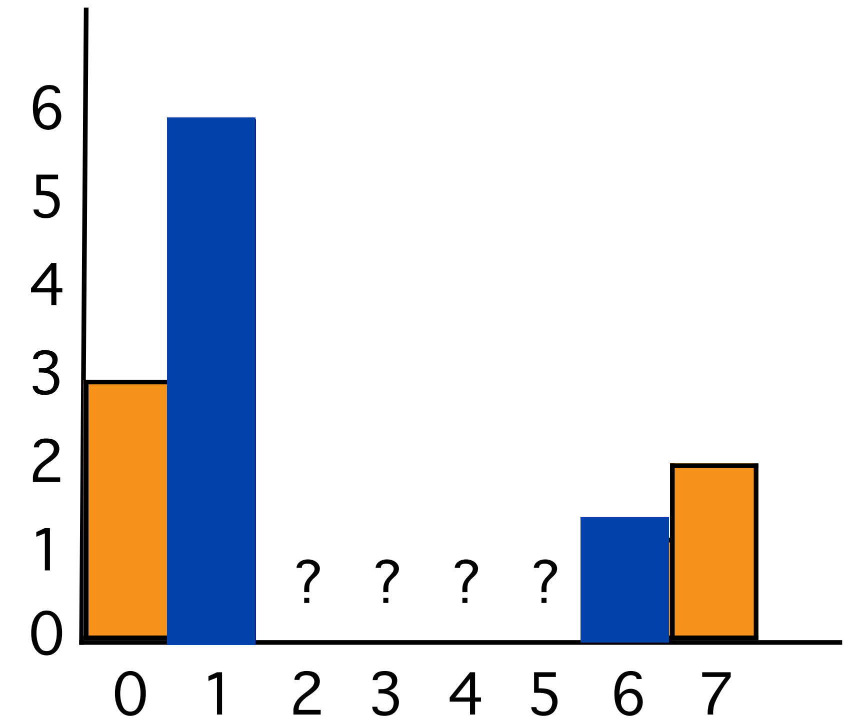
slide by Frédo Durand, MIT

1D example: minimization

- Copy



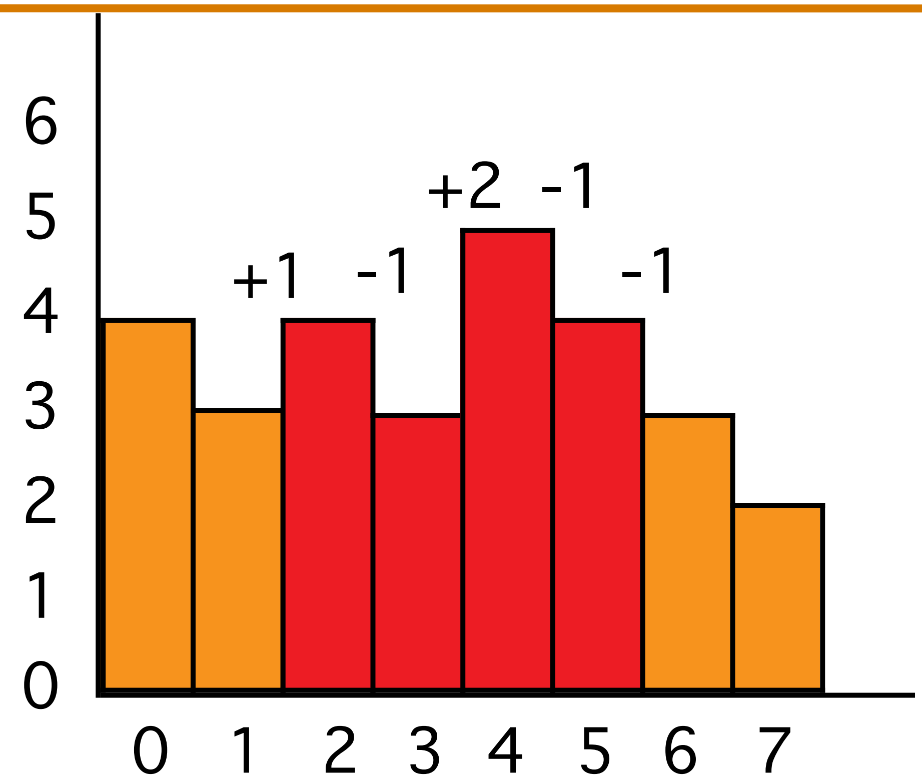
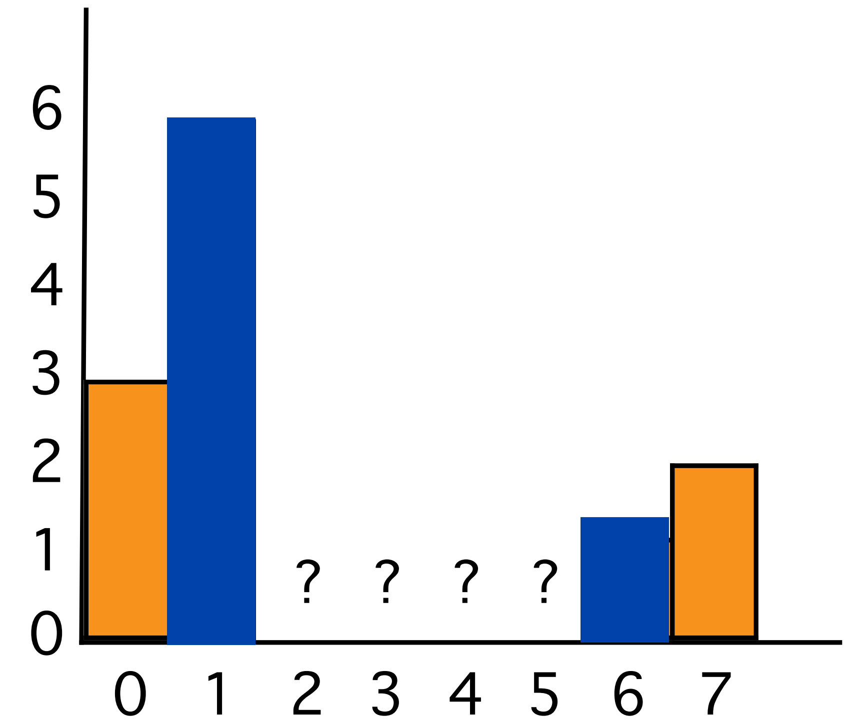
to



$$
\begin{aligned}
&\text{Min } [(f_2-f_1)-1]^2 && \Longrightarrow f_2^2+49-14f_2 \\
&+ [(f_3-f_2)-(-1)]^2 && \Longrightarrow f_3^2+f_2^2+1-2f_3f_2 +2f_3-2f_2 \\
&+ [(f_4-f_3)-2]^2 && \Longrightarrow f_4^2+f_3^2+4-2f_3f_4 -4f_4+4f_3 \\
&+ [(f_5-f_4)-(-1)]^2 && \Longrightarrow f_5^2+f_4^2+1-2f_5f_4 +2f_5-2f_4 \\
&+ [(f_6-f_5)-(-1)]^2 && \Longrightarrow f_5^2+4-4f_5
\end{aligned}
$$

slide by Frédo Durand, MIT

1D example: big quadratic

- Copy  to 

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

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

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

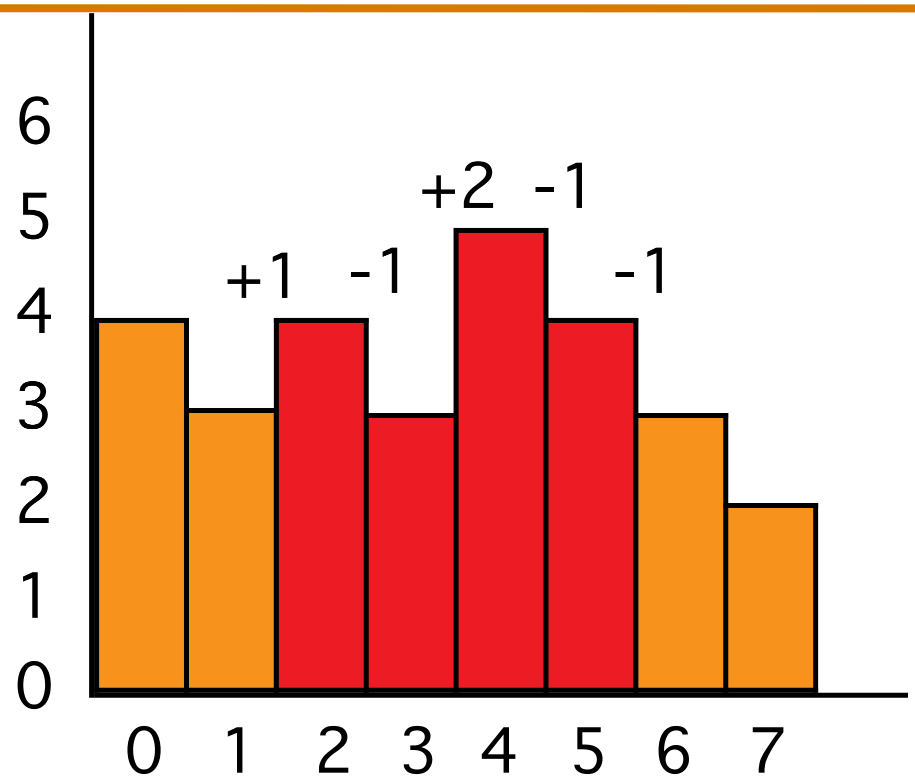
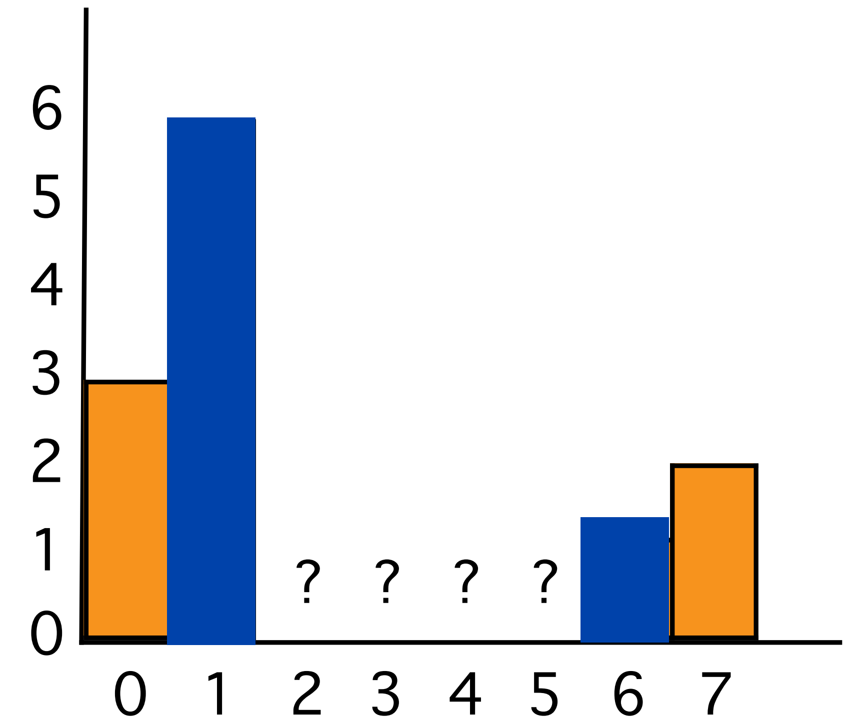
$+ f_5^2+f_4^2+1-2f_5f_4 +2f_5-2f_4$

$+ f_5^2+4-4f_5)$

Denote it Q

slide by Frédo Durand, MIT

1D example: derivatives

- Copy  to 

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

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

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

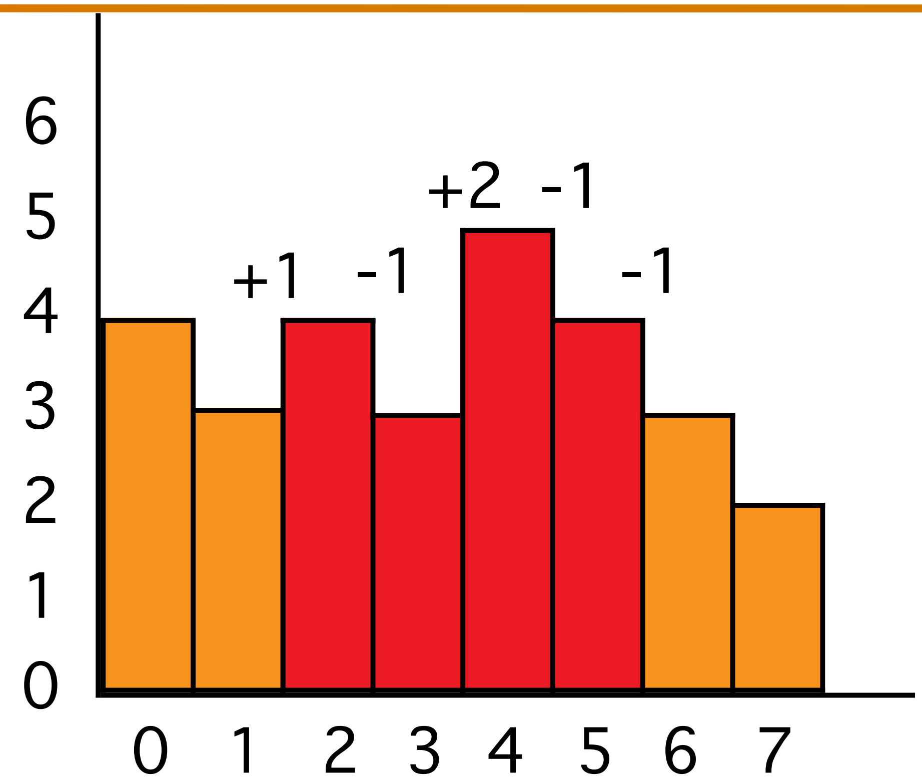
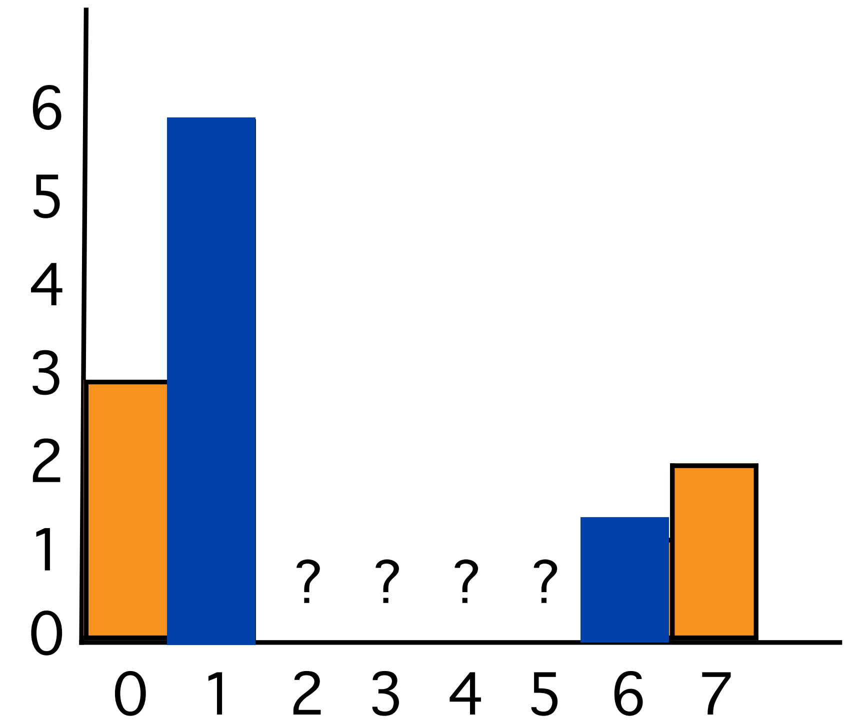
$+ f_5^2+f_4^2+1-2f_5f_4 +2f_5-2f_4$

$+ f_5^2+4-4f_5)$

Denote it Q

slide by Frédo Durand, MIT

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

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

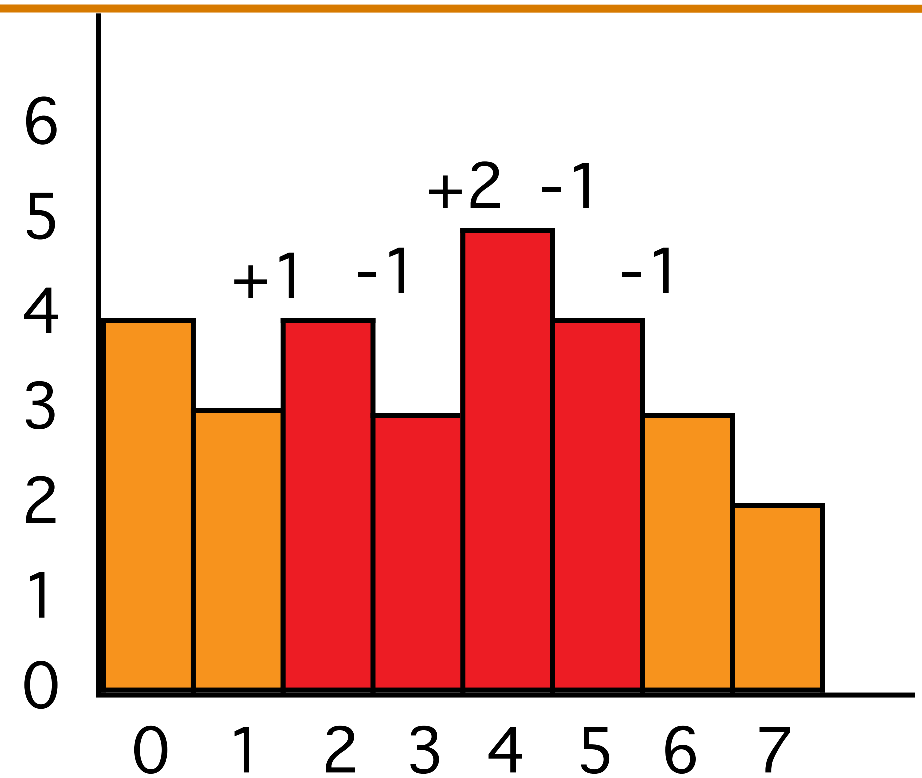
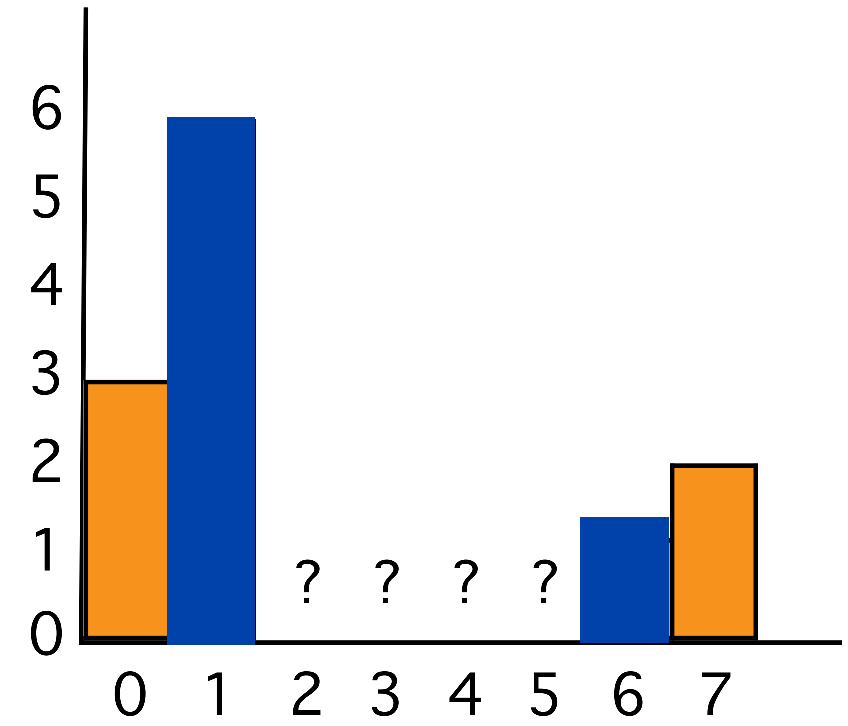
$$\frac{dQ}{df_3} = 2f_3 - 2f_2 + 2 + 2f_3 - 2f_4 + 4$$

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

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

slide by Frédo Durand, MIT

1D example: set derivatives to zero

- **Copy**  **to** 

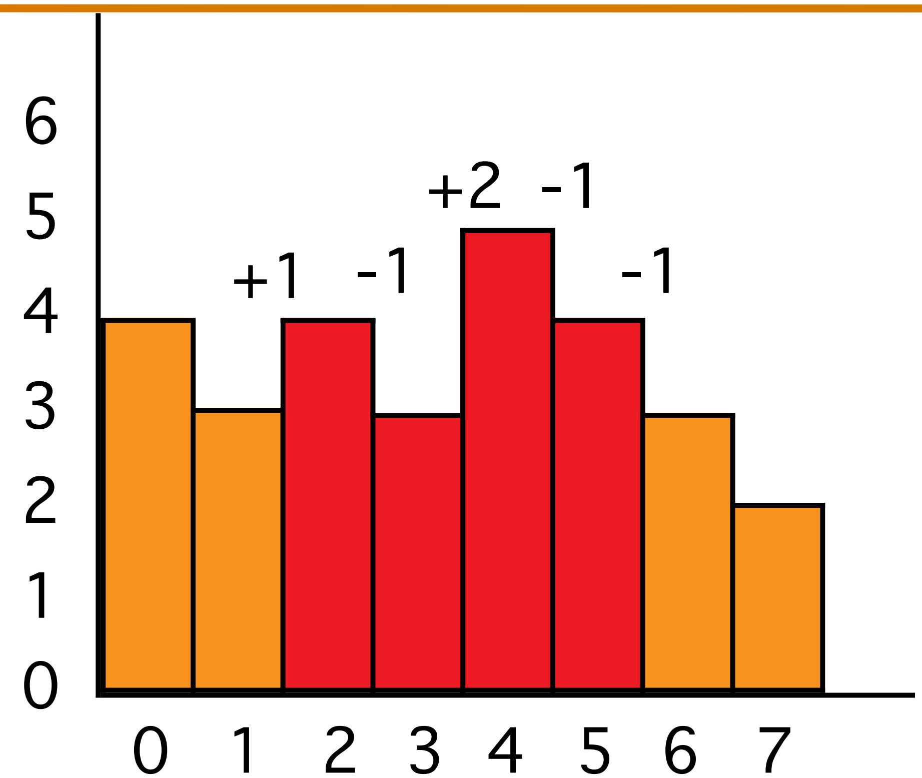
$$\frac{dQ}{df_2} = 2f_2 + 2f_2 - 2f_3 - 16$$

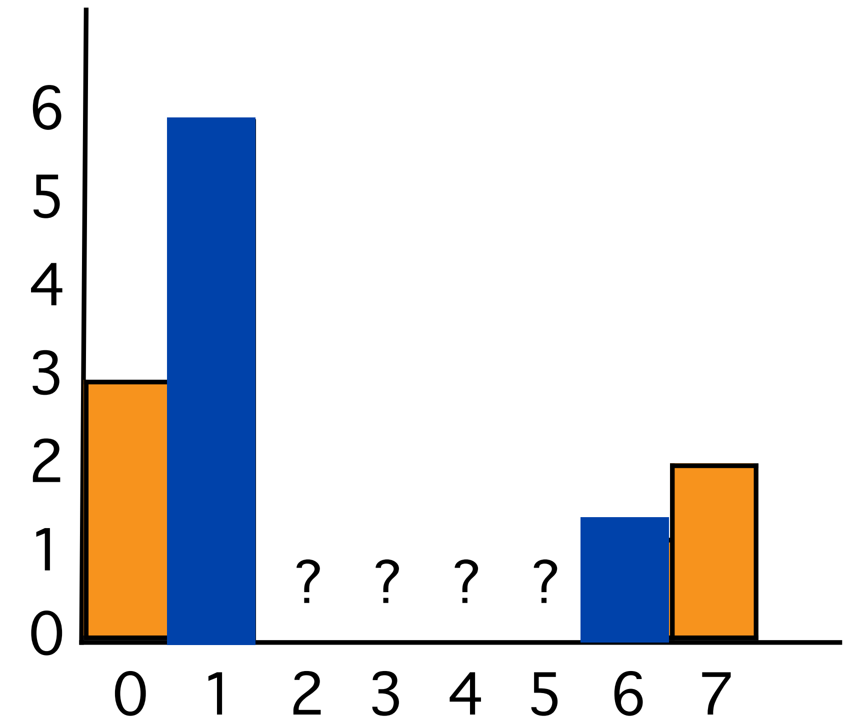
$$\frac{dQ}{df_3} = 2f_3 - 2f_2 + 2 + 2f_3 - 2f_4 + 4$$

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

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

1D example: set derivatives to zero

- **Copy** 

to 

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

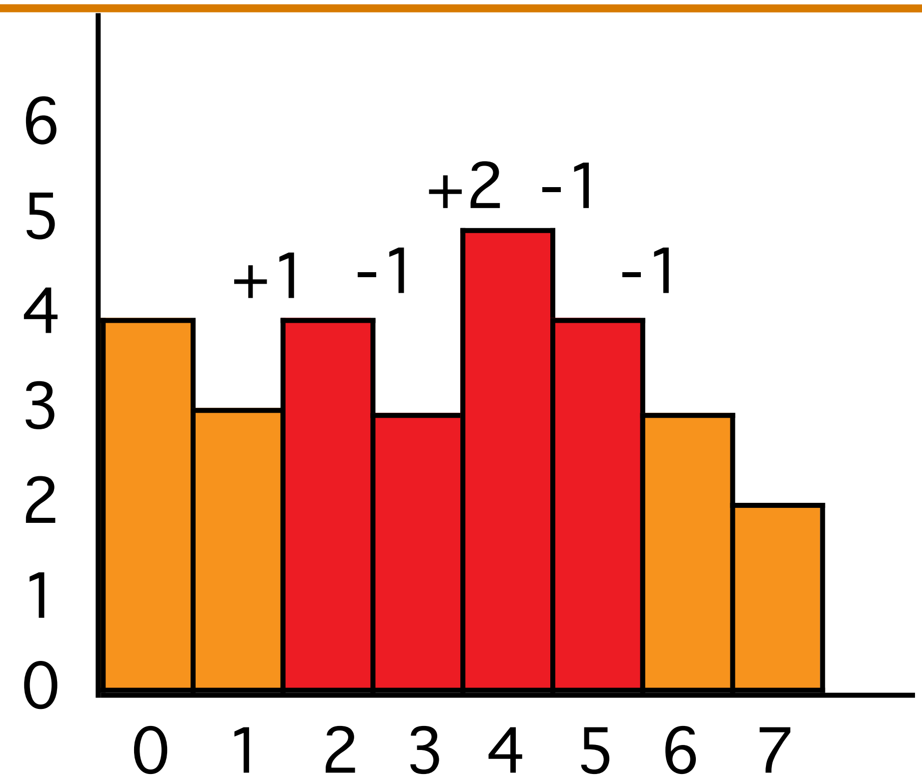
$$\frac{dQ}{df_3} = 2f_3 - 2f_2 + 2 + 2f_3 - 2f_4 + 4 = 0$$

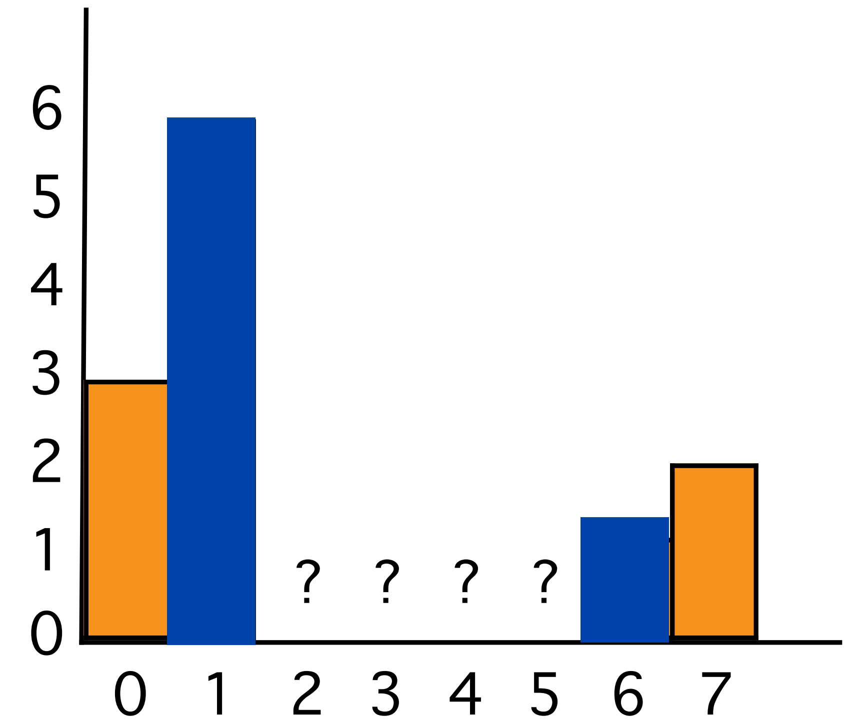
$$\frac{dQ}{df_4} = 2f_4 - 2f_3 - 4 + 2f_4 - 2f_5 - 2 = 0$$

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

slide by Frédo Durand, MIT

1D example: set derivatives to zero

- Copy 

to 

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

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

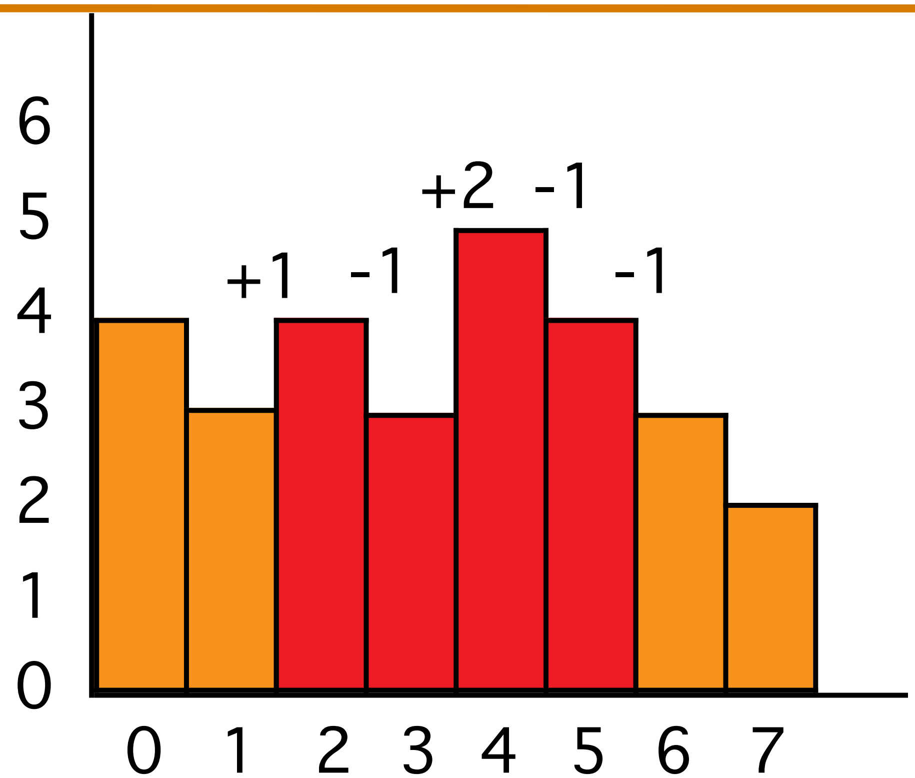
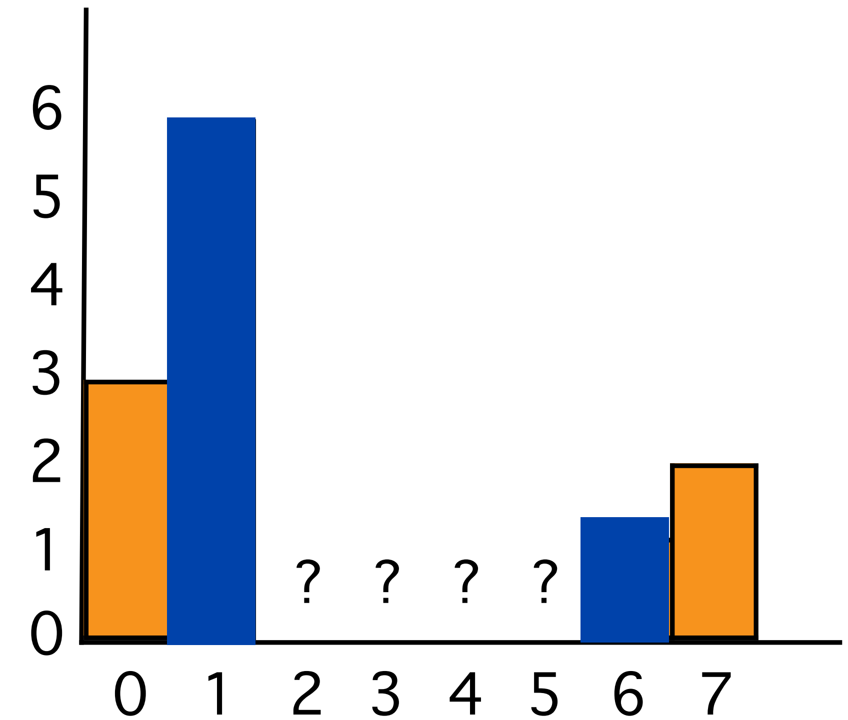
$$\frac{dQ}{df_4} = 2f_4 - 2f_3 - 4 + 2f_4 - 2f_5 - 2 \quad =0$$

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

==>

slide by Frédo Durand, MIT

1D example: set derivatives to zero

- **Copy**  **to** 

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

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

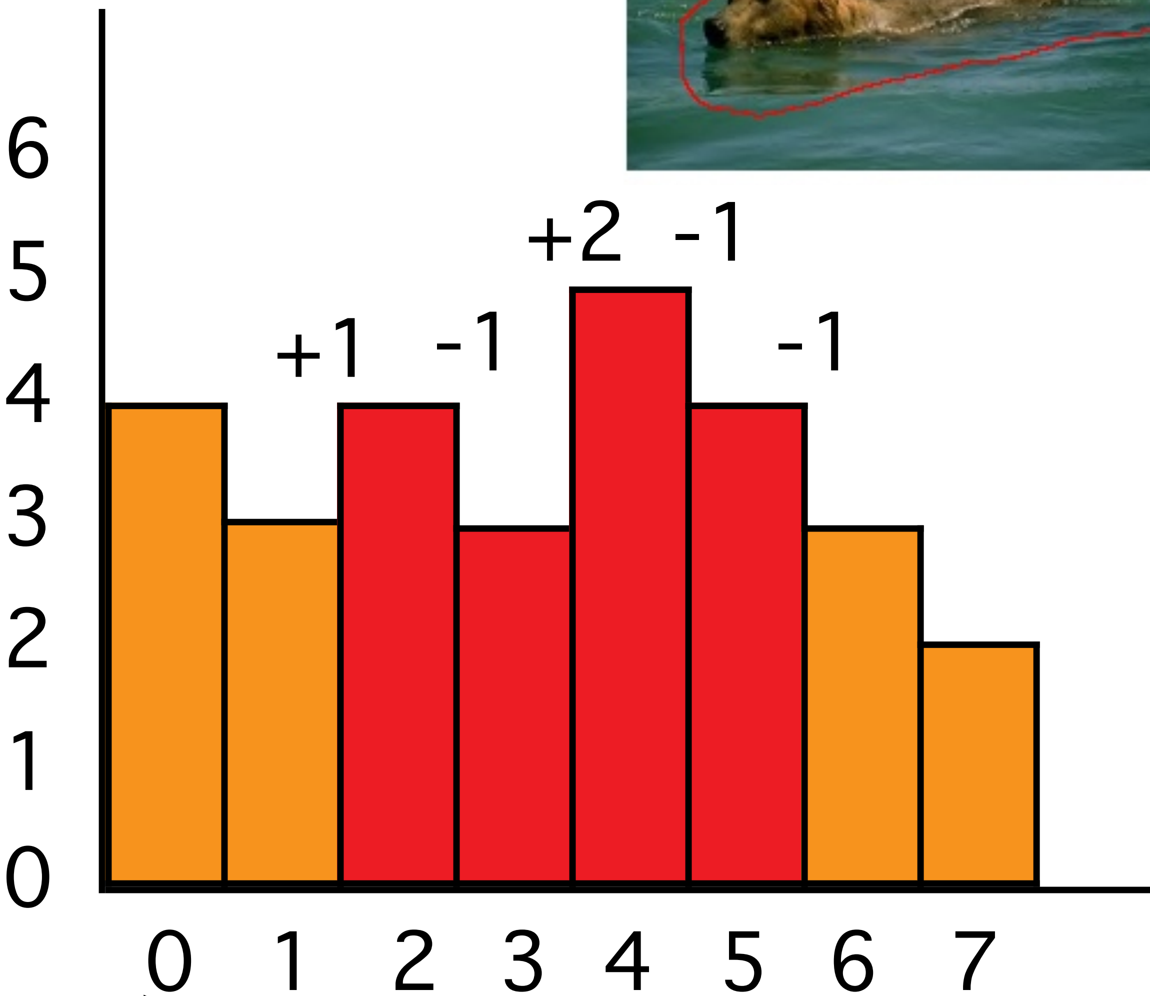
$$\frac{dQ}{df_4} = 2f_4 - 2f_3 - 4 + 2f_4 - 2f_5 - 2 = 0$$

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

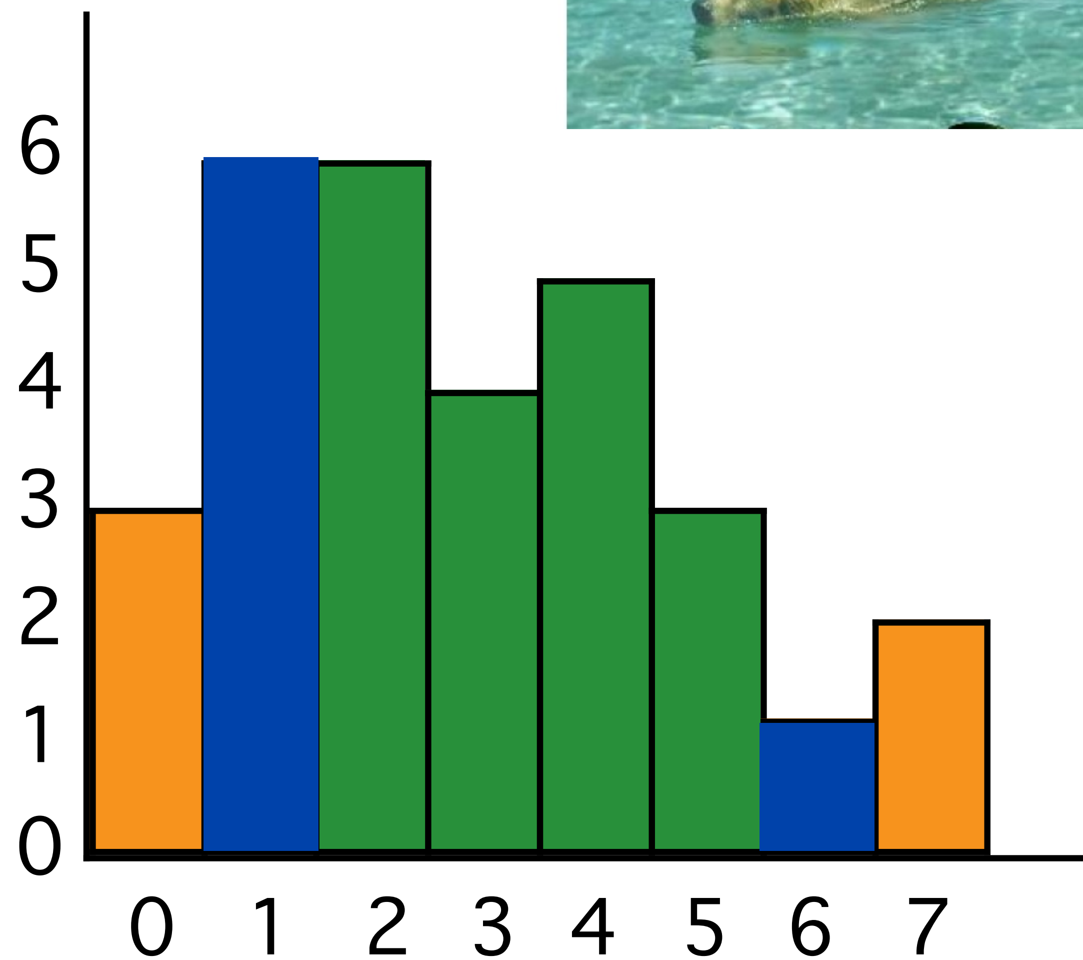
$$\Longrightarrow \begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 16 \\ -6 \\ 6 \\ 2 \end{pmatrix}$$

1D example recap

- **Copy**



to



==>

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

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

Matrix structure

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

- **That matrix is $G^T G$; least squares system reads**

  and the solution to $(G^T G)f = G^T b$ is the minimizer. (This system is the normal equations for the LLS problem.)

- **Interesting that it looks like a second derivative…**

Matrix structure

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

- **That matrix is G^TG; least squares system reads**

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 7 \\ -1 \\ 2 \\ -1 \\ -2 \end{pmatrix}$$

  and the solution to $(G^TG)f = G^Tb$ is the minimizer. (This system is the normal equations for the LLS problem.)

- **Interesting that it looks like a second derivative…**

Matrix structure in 2D

- **The matrix *G* has:**

  one column for each pixel (one per unknown pixel after projection)

  one row for each neighbor-edge joining two pixels

  a 1 and a –1 in each row (some with just 1 or zero after projection)

- **The matrix $A = G^T G$ has:**

  one row and column for each (unknown) pixel

- **Away from constraints, $G^T G$ implements a convolution with a discrete Laplacian filter**

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

no surprise this is a second derivative: applied derivative twice

Euler-Lagrange

- **Analogous conversion to square system in 2D continuous case**

$$\min_f \|\nabla f - \vec{g}\|_2 \quad \text{subject to} \quad f|_B = f^*$$

- **Euler-Lagrange equations give a solution to this variational problem; in this case they work out to**

$$\nabla^2 f = \nabla \cdot \vec{g} \quad \text{subject to} \quad f|_B = f^*$$

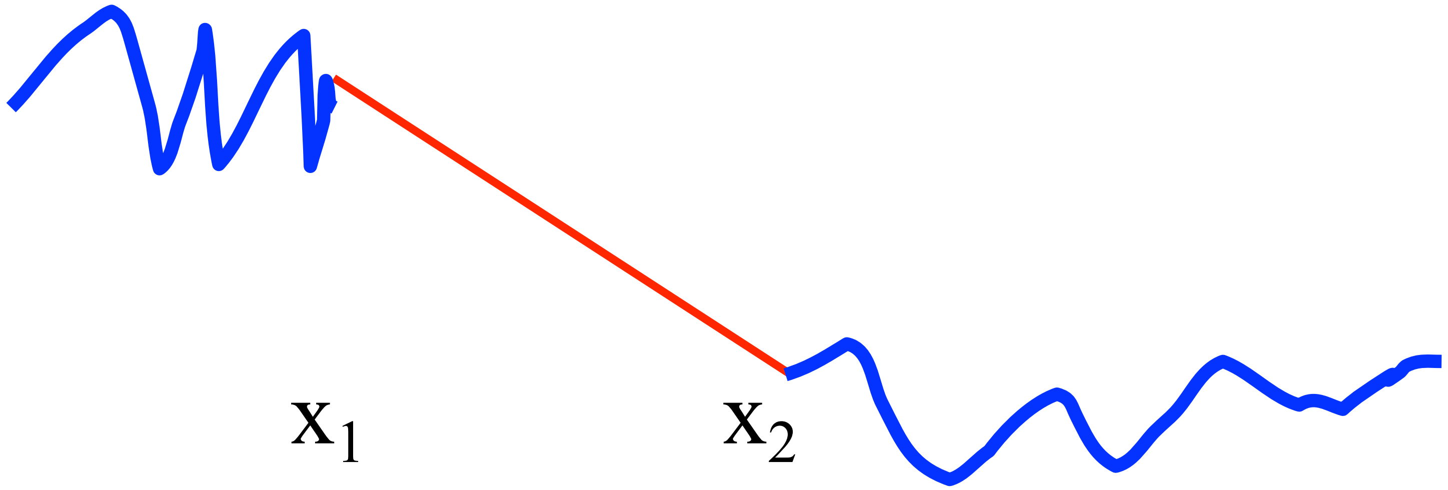
  reads “laplacian f equals divergence g”

- **This is Poisson’s equation, which explains the use of the word “Poisson” to describe this class of methods**

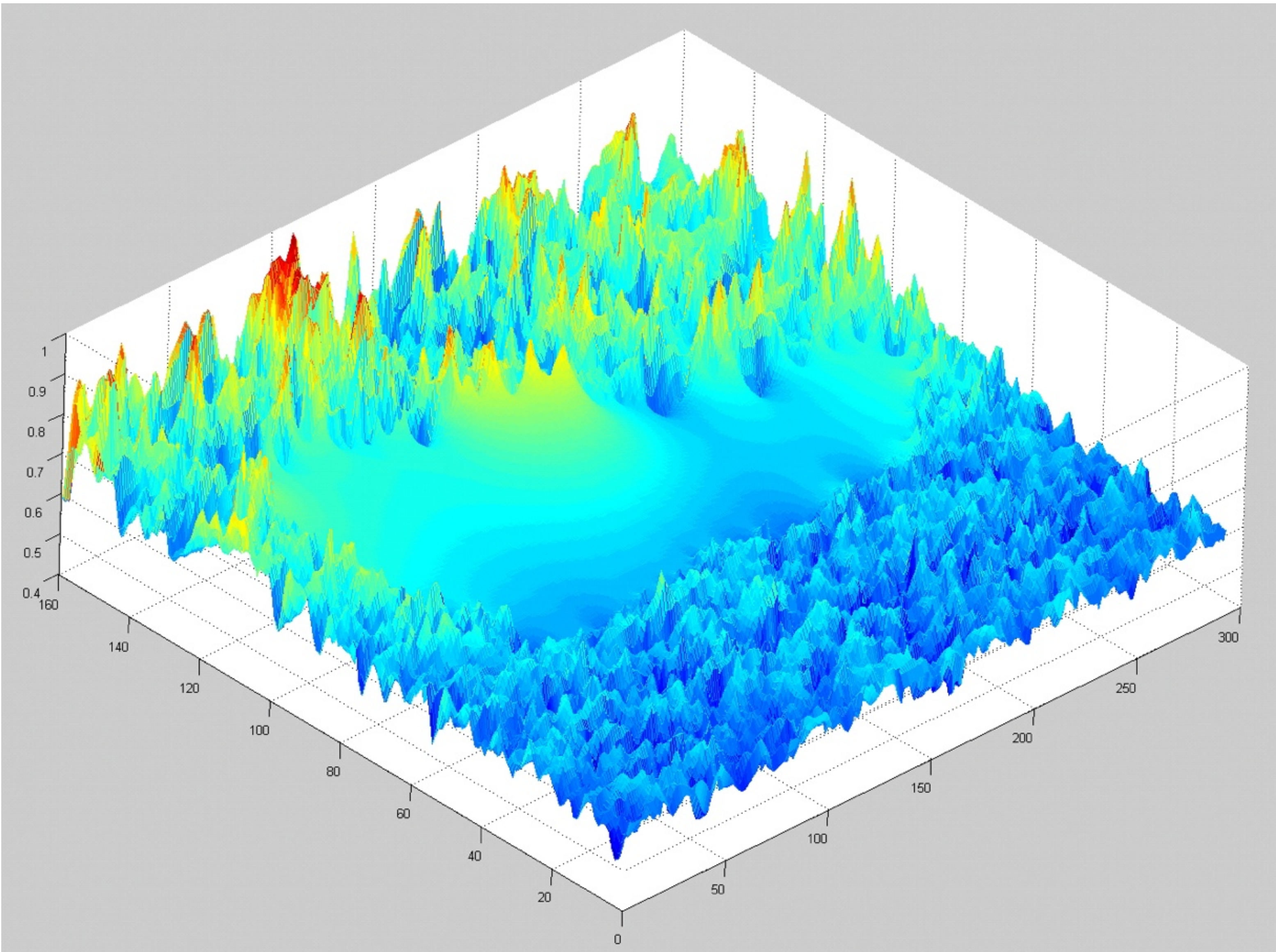
  don’t need this, computationally; just solve the discrete least squares system, which is easier than discretizing the Poisson equation.

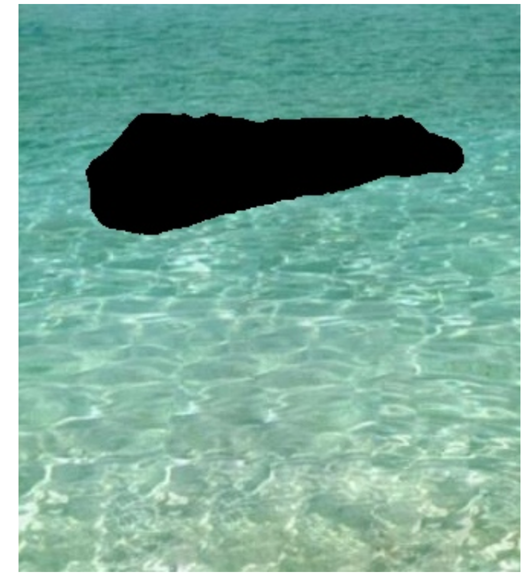
Intuition

- 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

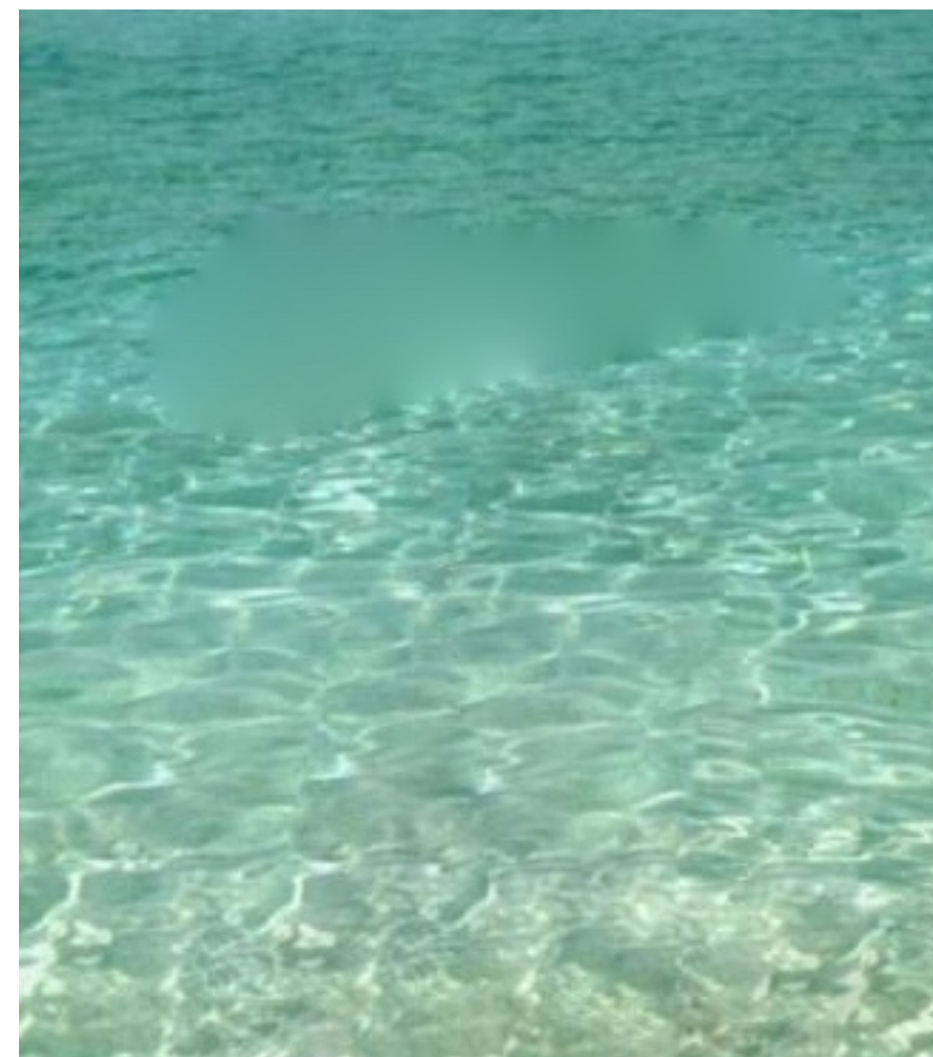


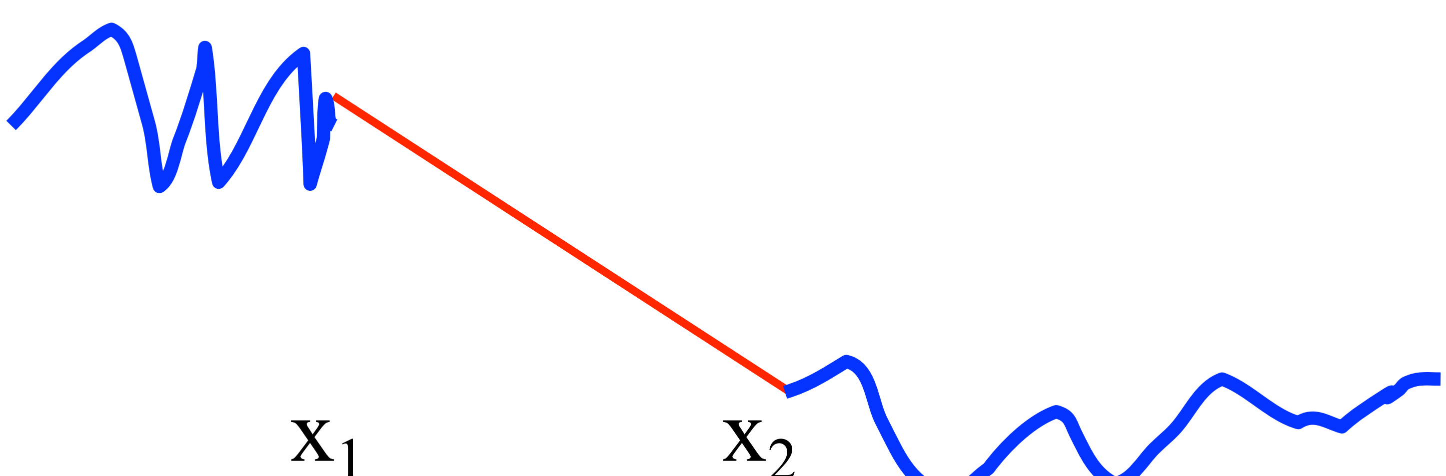
In 2D: membrane interpolation





Not as simple





Solution methods

- **The matrix *A* is square, sparse, and positive definite**
- **Direct solve**

  just form the matrix and solve it—fine for smaller problems
- **Steepest descent**

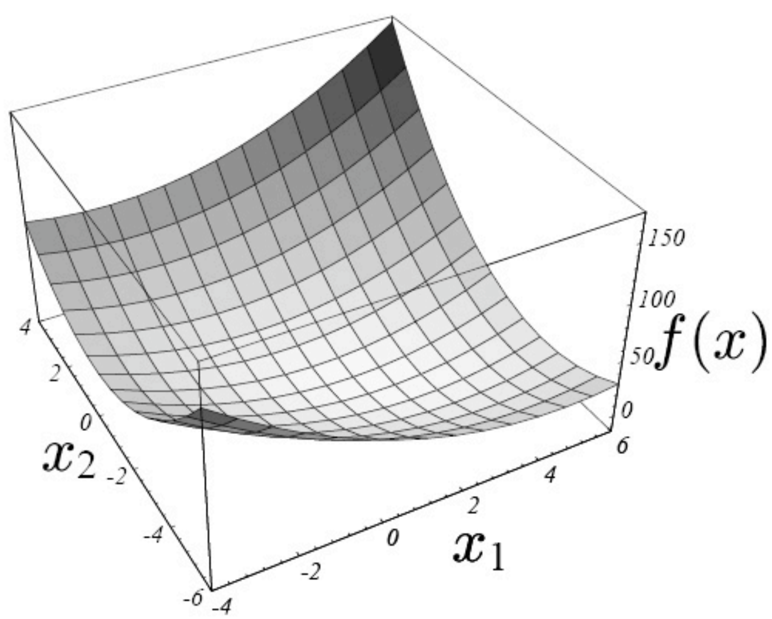
  a simple-minded iterative method
- **Conjugate gradients**

  a cleverer and much faster iterative method
- **Preconditioned conjugate gradients**

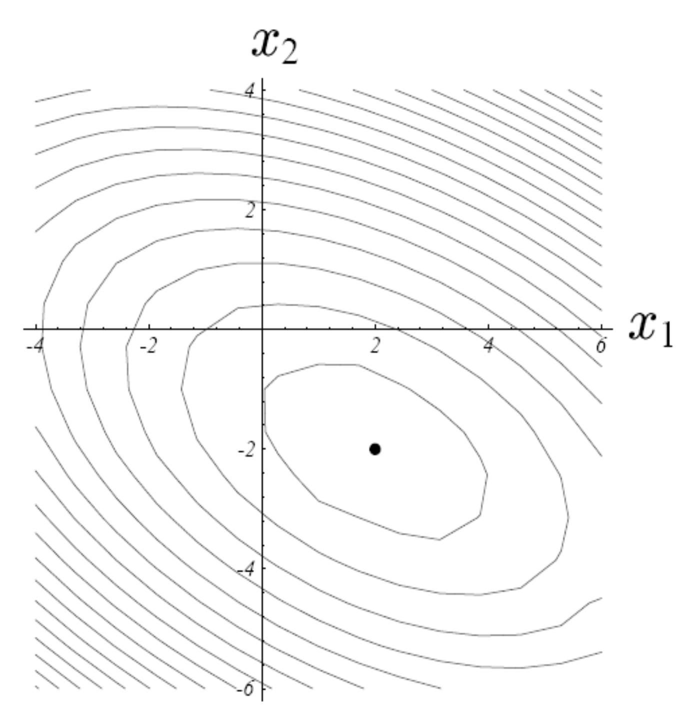
  CG can be greatly sped up for larger problems

Turn Ax=b into a minimization problem

- **Minimization is more logical to analyze iteration (gradient ascent/descent)**
- **Quadratic form** $f(x) = \frac{1}{2}x^T Ax - b^T x + c$
  - c can be ignored because we want to minimize
- **Intuition:**
  - the solution of a linear system is always the intersection of n hyperplanes
  - Take the square distance to them
  - A needs to be positive-definite so that we have a nice parabola with a minimum, not maximum



Graph of quadratic form $f(x) = \frac{1}{2}x^T Ax - b^T x + c$. The minimum point of this surface is the solution to $Ax = b$.



Contours of the quadratic form. Each ellipsoidal curve has constant $f(x)$.

Gradient of the quadratic form

$$f'(x) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(x) \\ \frac{\partial}{\partial x_2} f(x) \\ \vdots \\ \frac{\partial}{\partial x_n} f(x) \end{bmatrix}$$

– Not our image gradient!

– Multidimensional gradient (as many dim as rows in matrix)

since

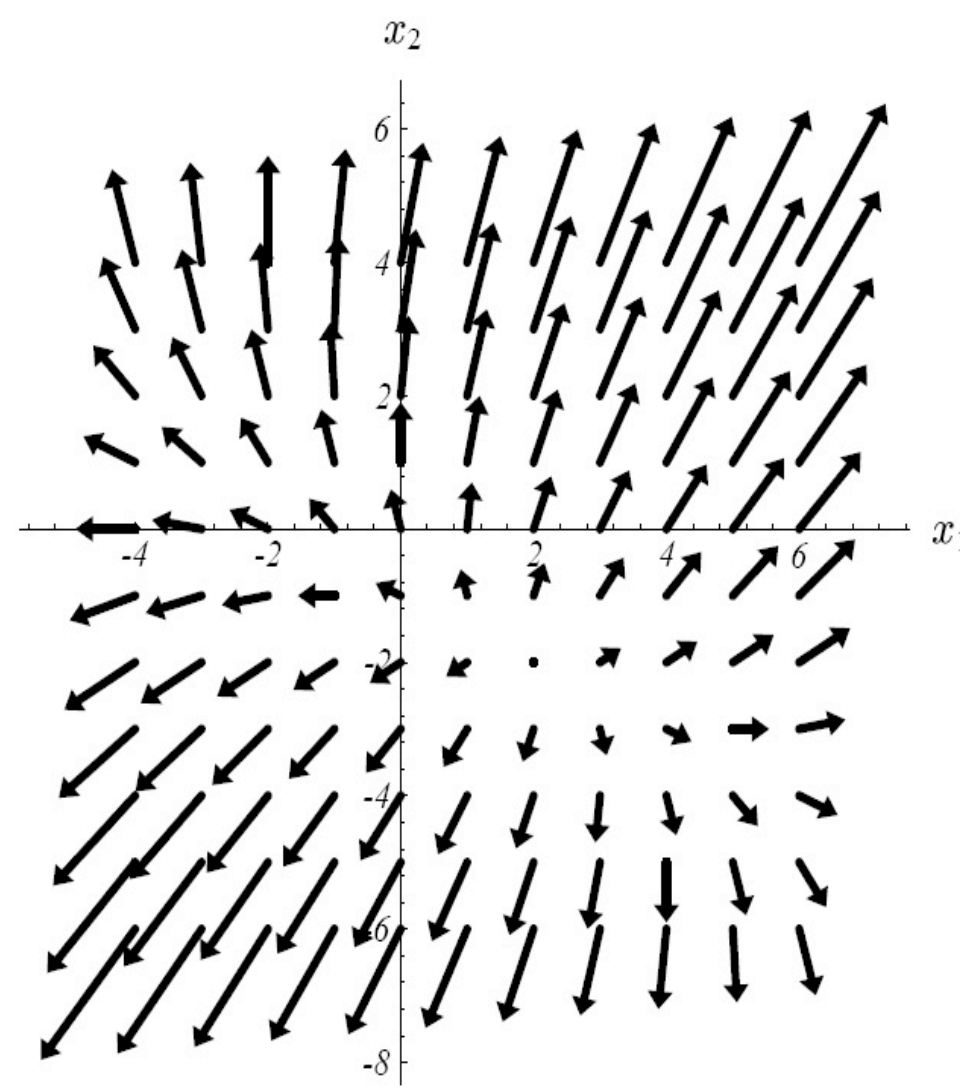
$$f(x) = \frac{1}{2} x^T A x - b^T x + c$$

$$f'(x) = \frac{1}{2} A^T x + \frac{1}{2} A x - b.$$

And since A is symmetric

$$f'(x) = Ax - b$$

Not surprising: we turned Ax=b into the quadratic minimization & vice versa



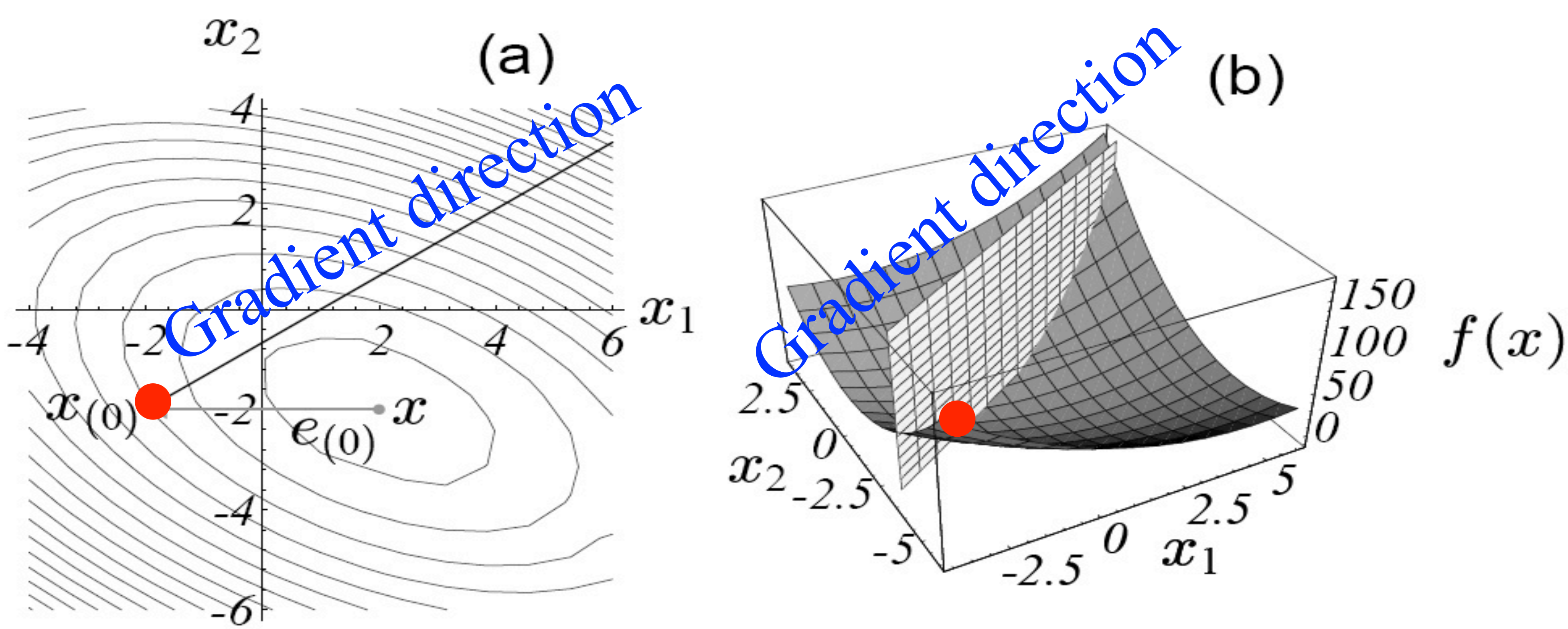
Gradient $f'(x)$ of the quadratic form. For every x, the gradient points in the direction of steepest increase of $f(x)$, and is orthogonal to the contour lines.

(if A is not symmetric, conjugate gradient finds solution for $\frac{1}{2}(A^T + A)x = b.$

slide by Frédo Durand, MIT

Steepest descent/ascent

- **Pick residual (negative gradient) direction**
  – $Ax_{(i)}$-b

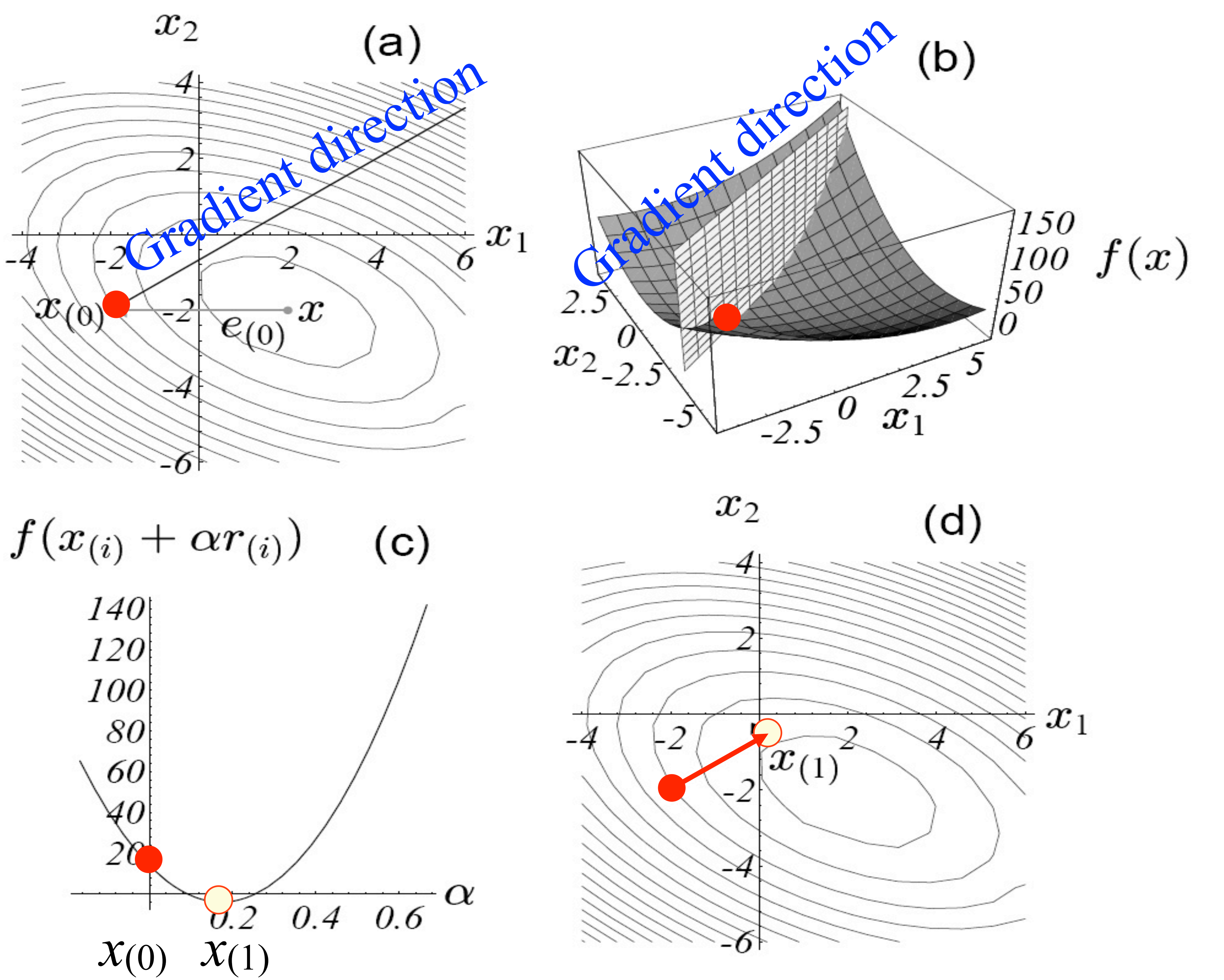


Steepest descent/ascent

- **Pick residual (negative gradient) direction**

  $-Ax_{(i)}$-b

- **Find optimum in this direction**

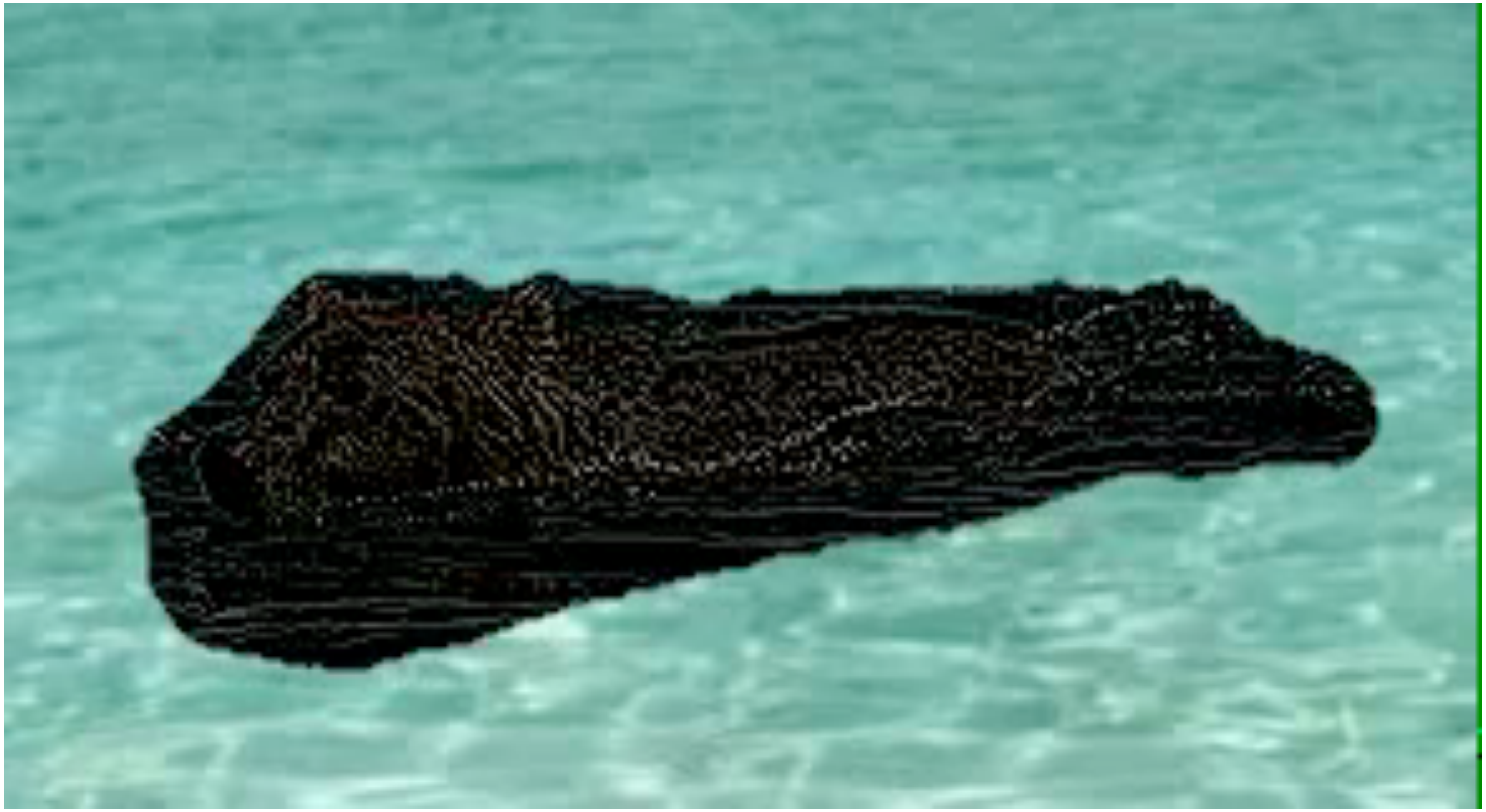


Energy along the gradient direction

slide by Frédo Durand, MIT

Convergence

- **A little slow: not fully there yet after 1000 iterations**

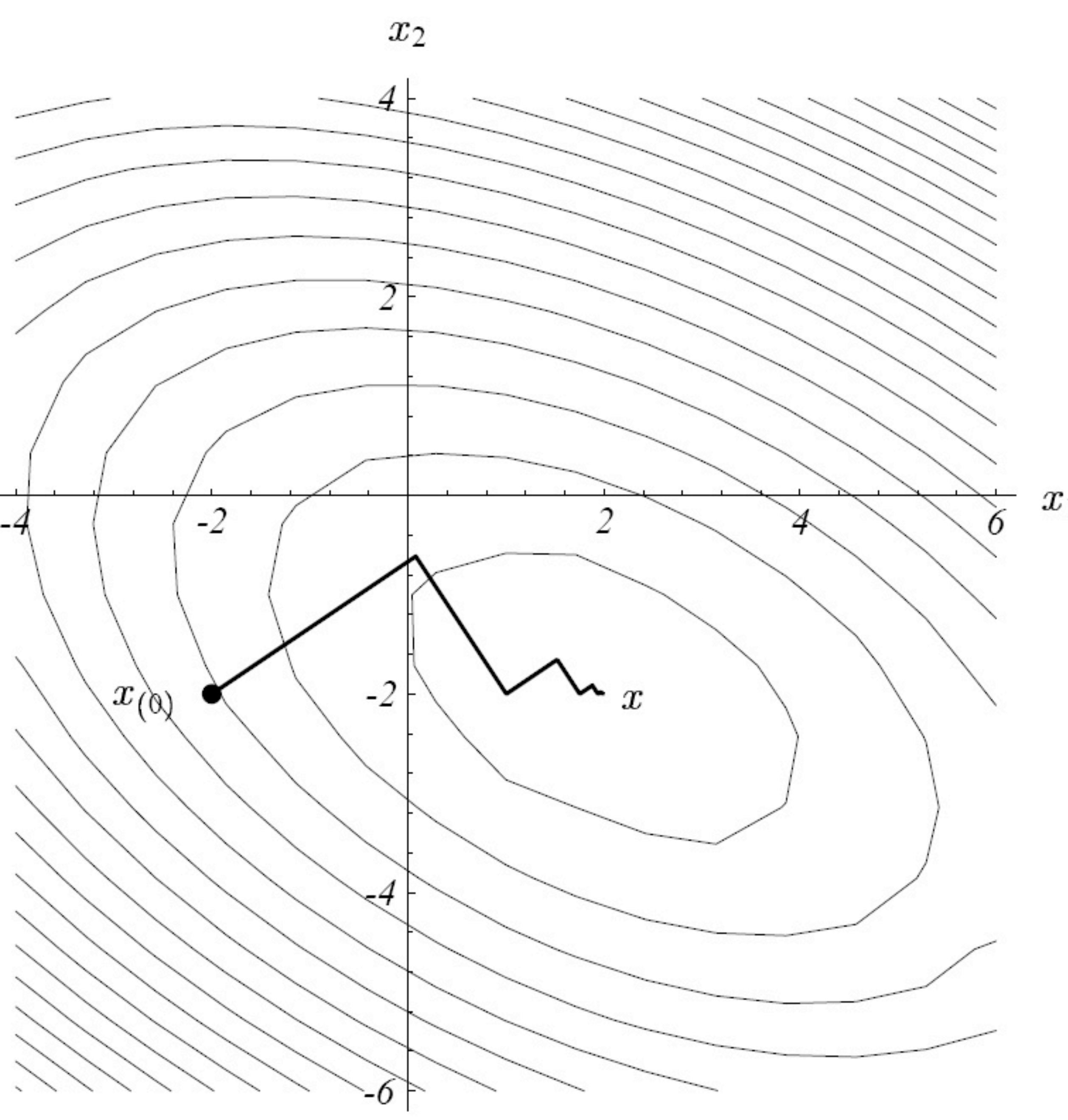


slide by Frédo Durand, MIT

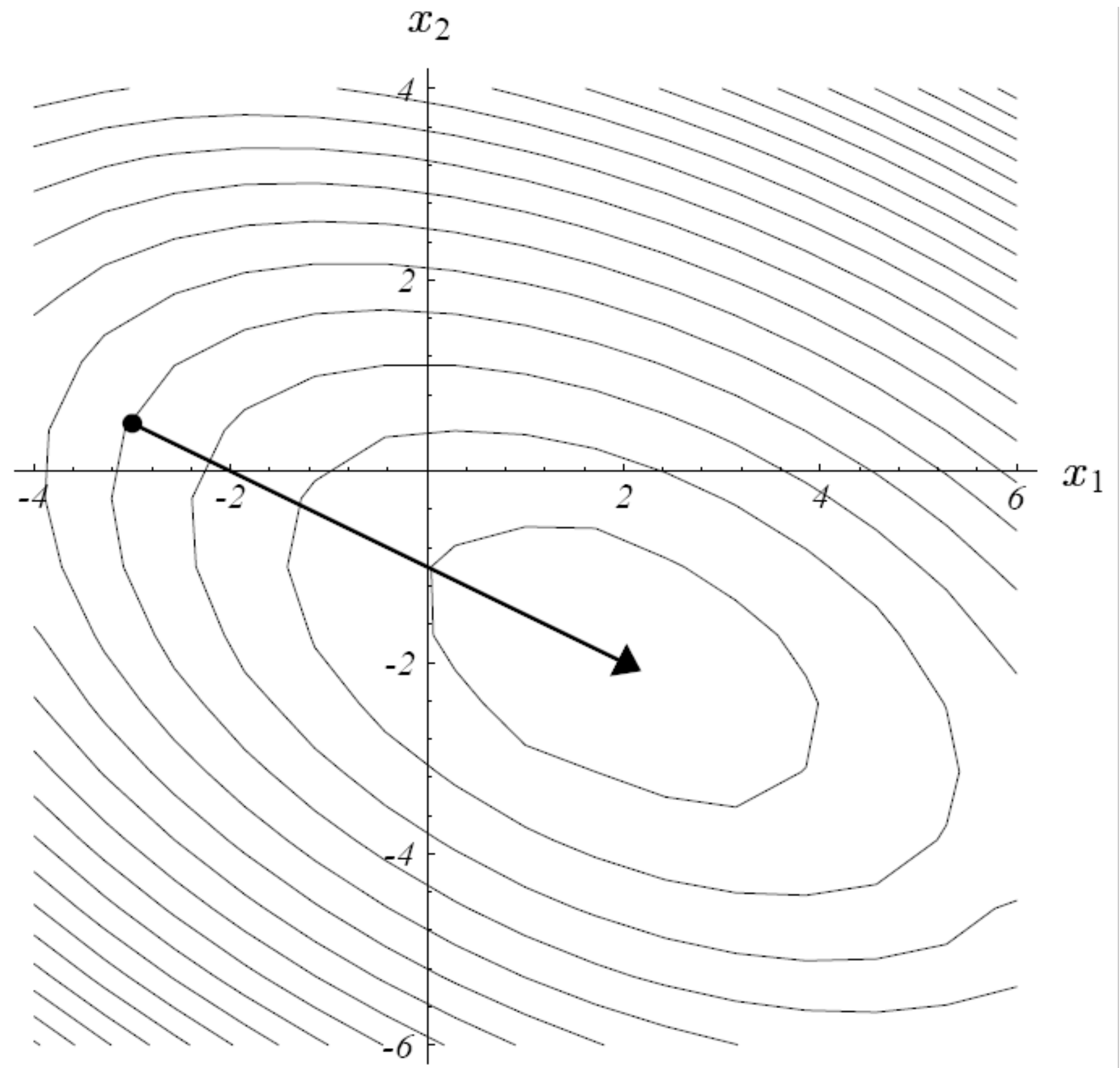
Behavior of gradient descent

- **Zigzag or goes straight depending if we're lucky**
  - Ends up doing multiple steps in the same direction

Unlucky

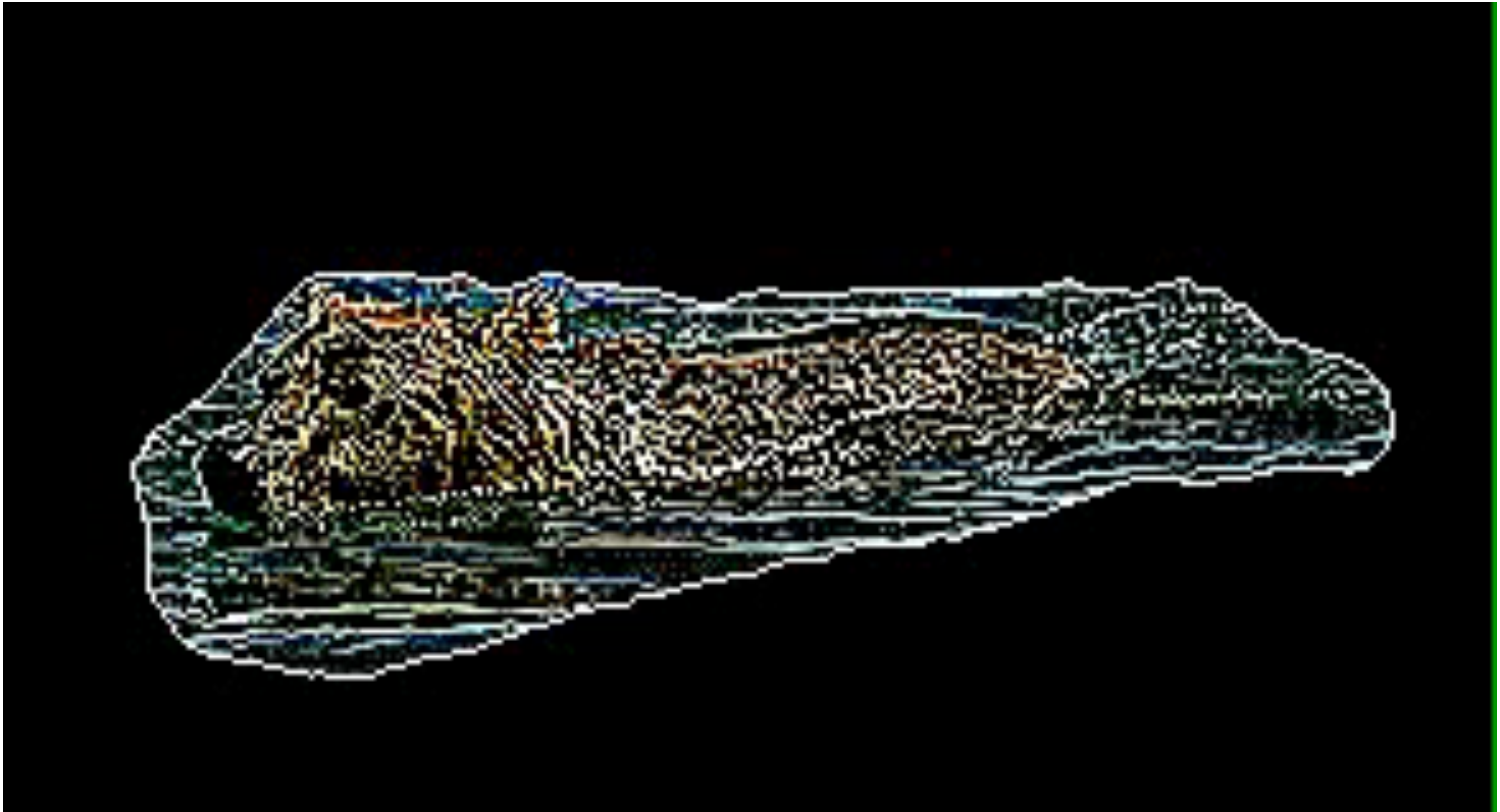


Lucky



Our residuals

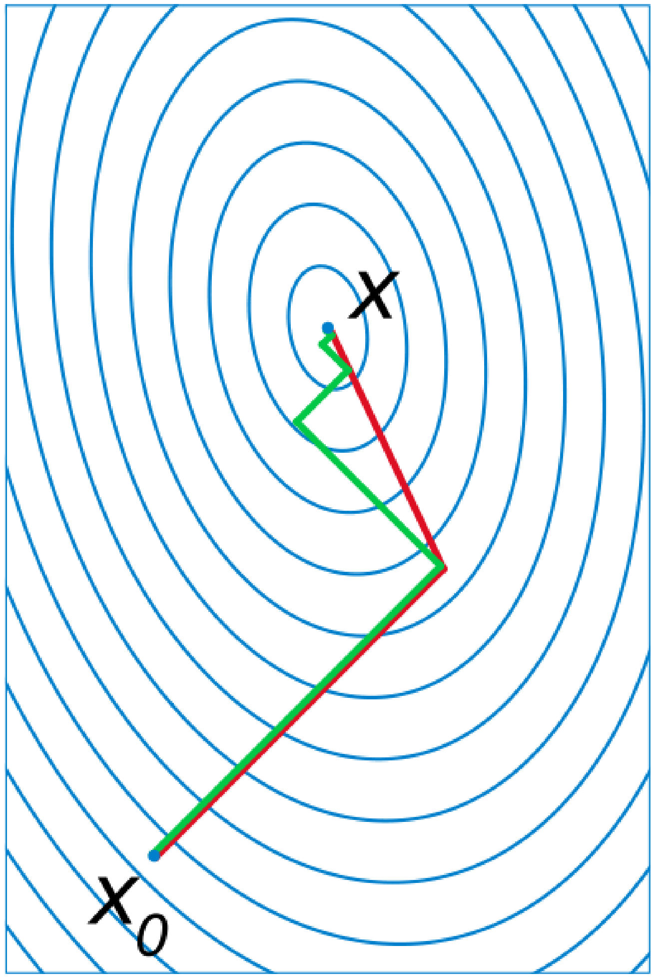
- times 10
- We zigzag between the two same checkerboard patterns



slide by Fredo Durand, MIT

Conjugate Gradient method

- **Naive iterative solver: Zigzag**
  - Ends up doing multiple steps in the same direction
- **Conjugate gradient: make sure never go twice in the same direction**
  - Don't go exactly along gradient direction



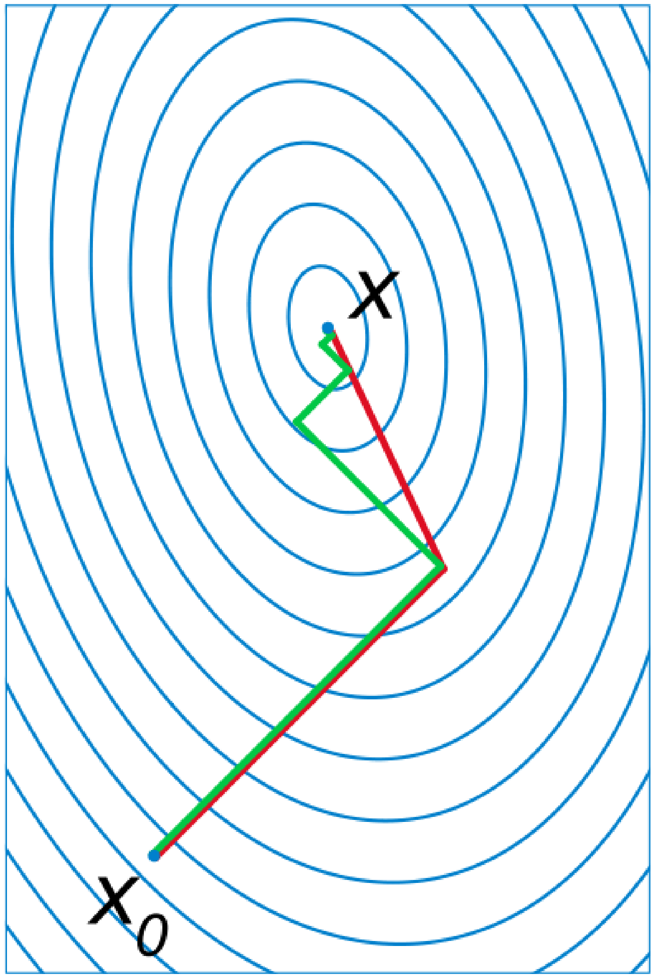
Green:
standard iterations
Red:
conjugate gradient

http://en.wikipedia.org/wiki/Image:Conjugate_gradient_illustration.svg

slide by Frédo Durand, MIT

Conjugate Gradient method

- **Naive iterative solver: Zigzag**
  - Ends up doing multiple steps in the same direction
- **Conjugate gradient: make sure never go twice in the same direction**
  - Don't go exactly along gradient direction



Green:
standard
iterations
Red:
conjugate
gradient

Good news: the code is simple

```
function [x] = conjgrad(A,b,x0)
    r = b - A*x0;
    w = -r;
    z = A*w;
    a = (r'*w)/(w'*z);
    x = x0 + a*w;
    B = 0;
    for i = 1:size(A);
        r = r - a*z;
        if( norm(r) < 1e-10 )
            break;
        B = (r'*z)/(w'*z);
        w = -r + B*w;
        z = A*w;
        a = (r'*w)/(w'*z);
        x = x + a*w;
```

http://en.wikipedia.org/wiki/Image:Conjugate_gradient_illustration.svg

Conjugate gradient

- **Smarter choice of direction**
  - Ideally, step directions should be orthogonal to one another (no redundancy)
  - But tough to achieve
  - Next best thing: make them A-orthogonal (conjugate)
    That is, orthogonal when transformed by √A
    - Turn the ellipses into circles

$$d_{(i)}^T A d_{(j)} = 0$$

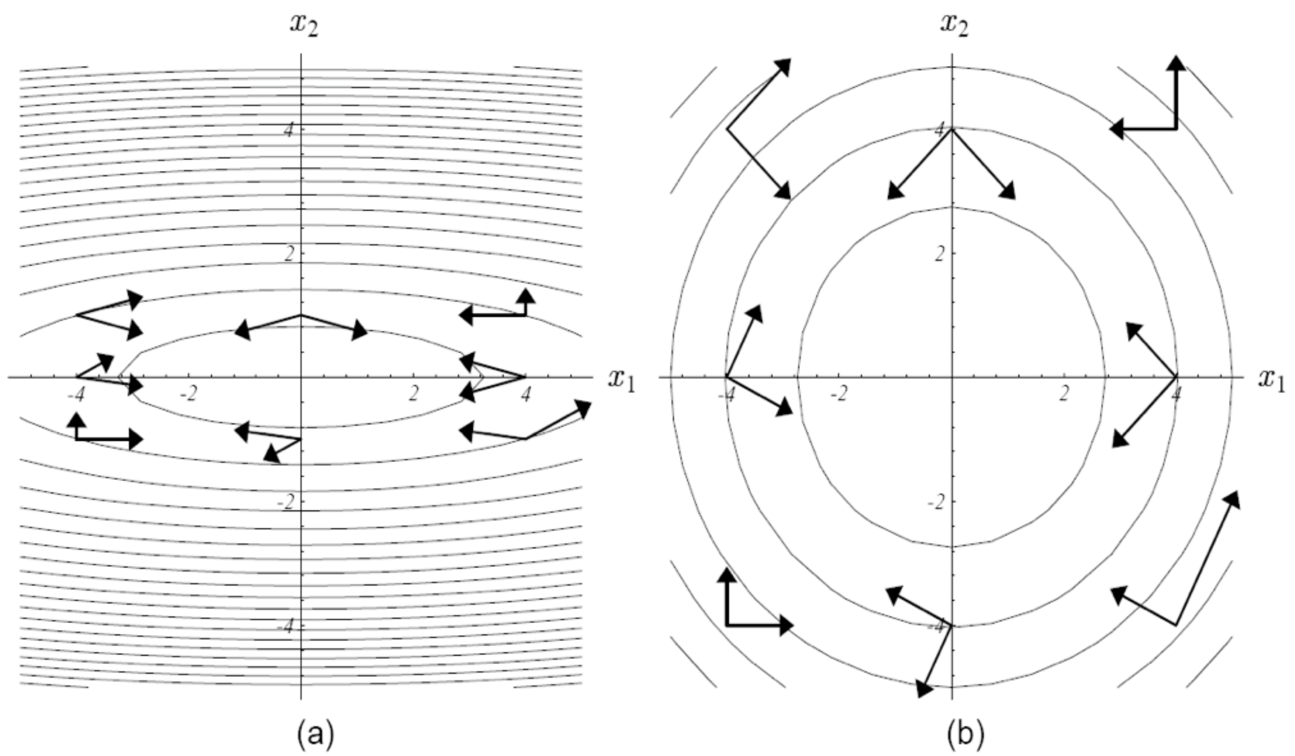
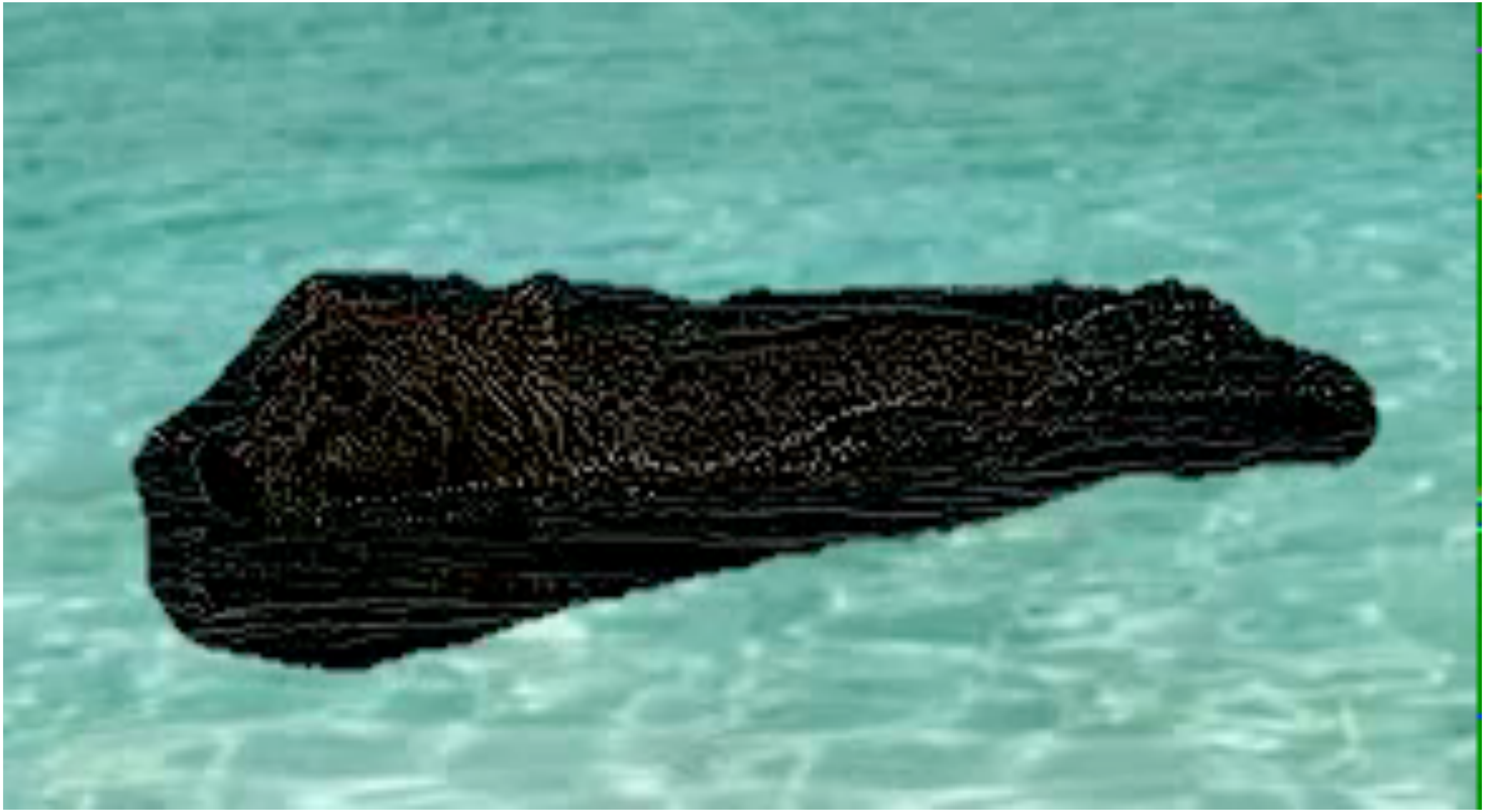


Figure 22: These pairs of vectors are A-orthogonal . . . because these pairs of vectors are orthogonal.

slide by Frédo Durand, MIT

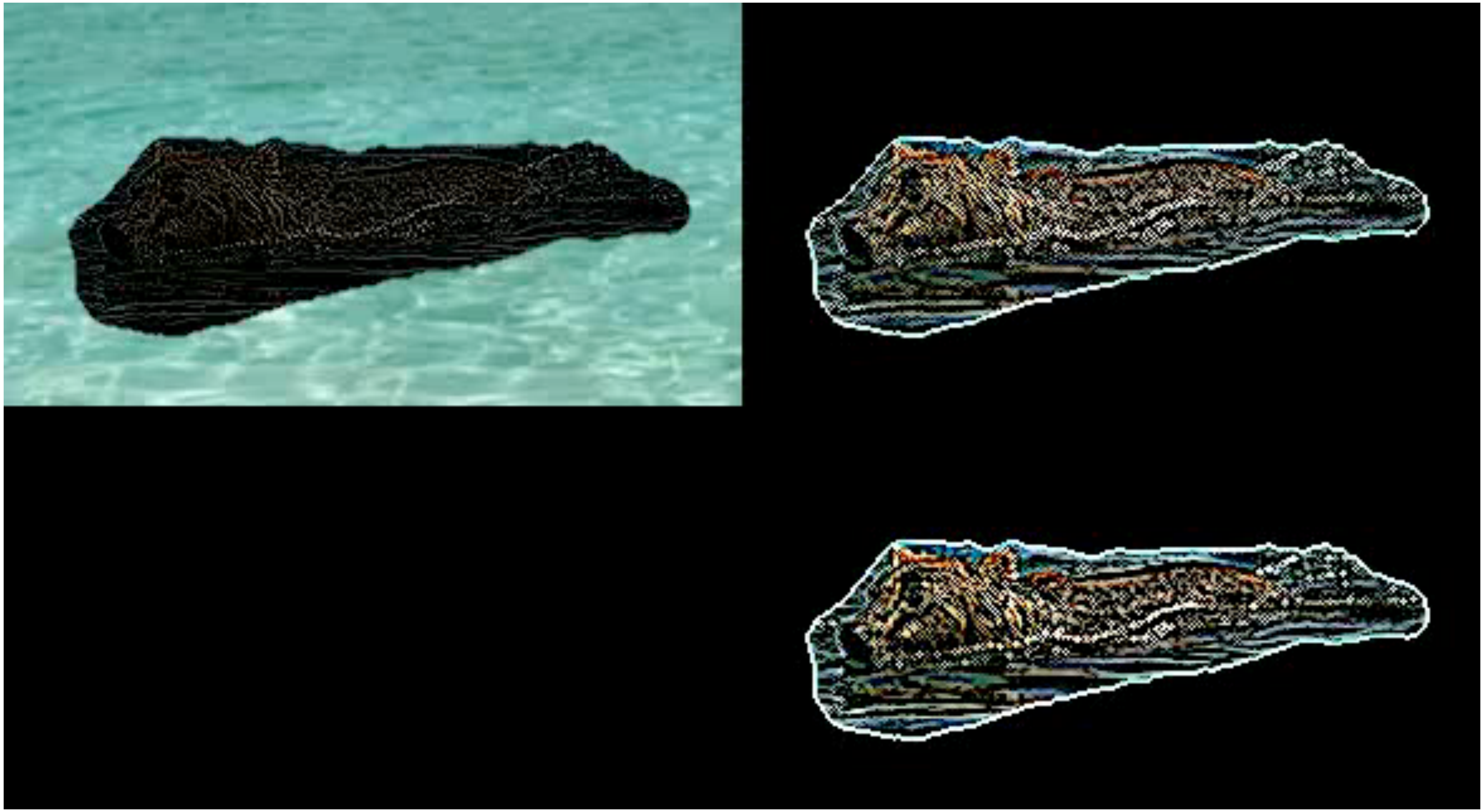
Convergence of CG



slide by Frédo Durand, MIT

Residuals and direction

- times 10, displayed at 10fps

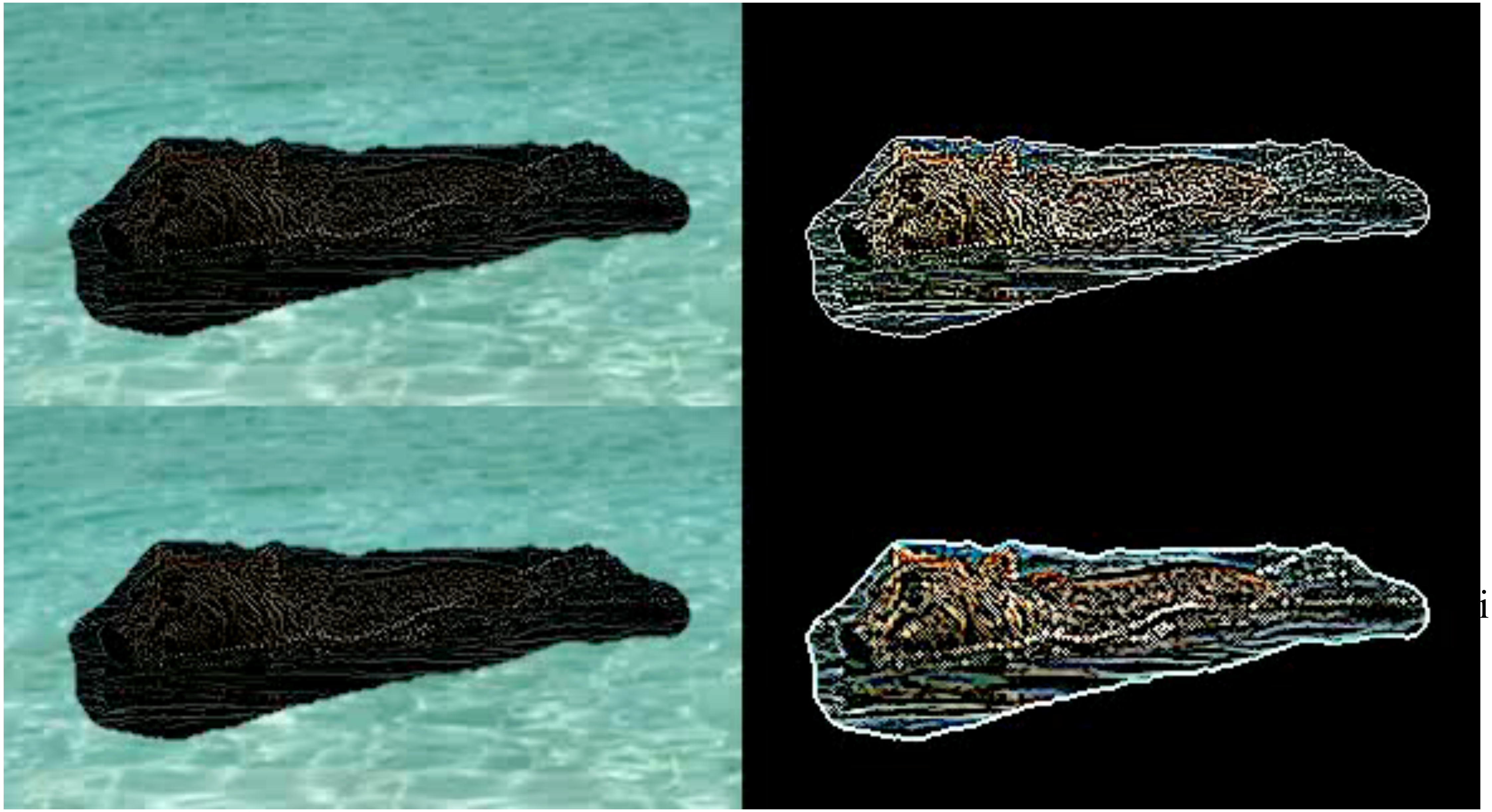


r_i

d_i

Compared to gradient descent

gradient
descent

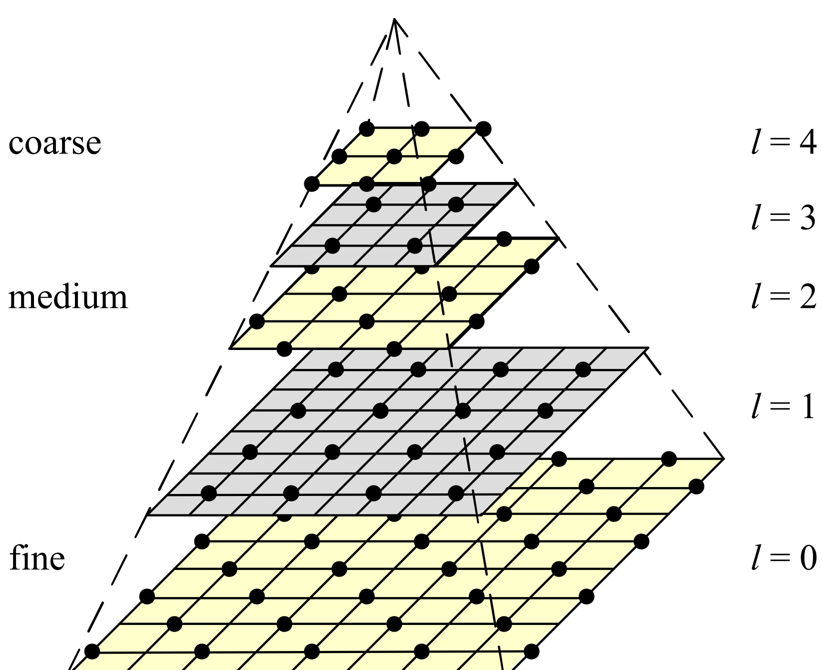


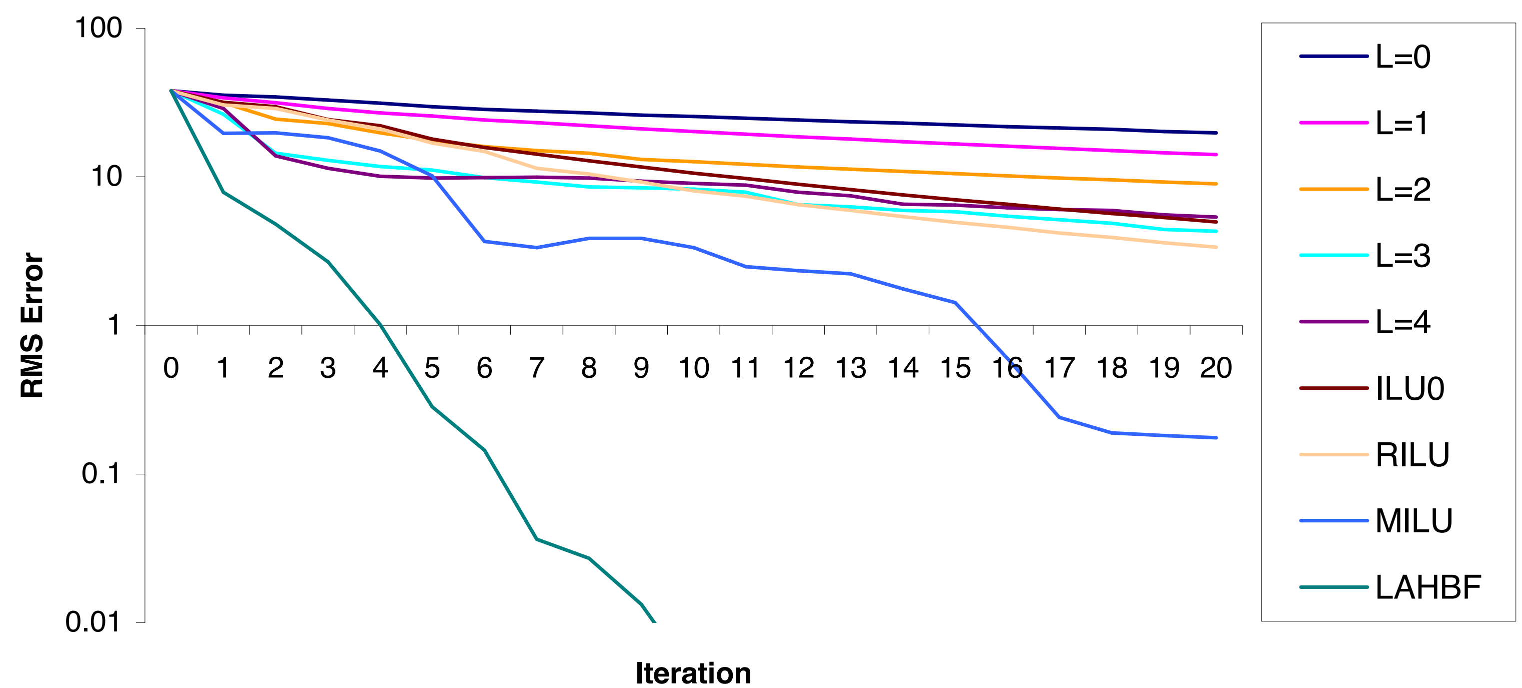
conjugate
gradient

slide by Frédo Durand, MIT

Preconditioners

- **When solving $Ax = b$ it's equivalent to solve $MAx = Mb$**
- **If $M = A^{-1}$ the problem becomes a lot easier**
- **If M at least converts A into a better conditioned matrix, it can greatly accelerate CG convergence**
- **Need a matrix we can efficiently solve systems with**
- **For Poisson problems on images, hierarchical preconditioners work well, particularly ones adapted to the problem**

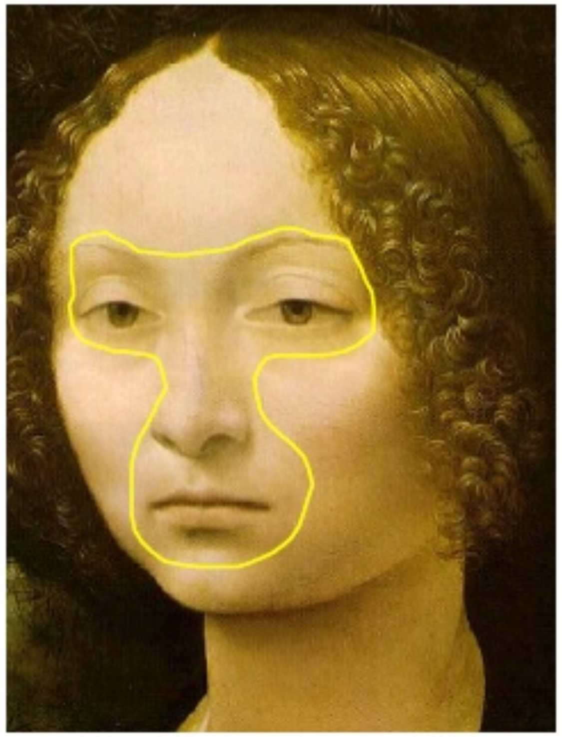


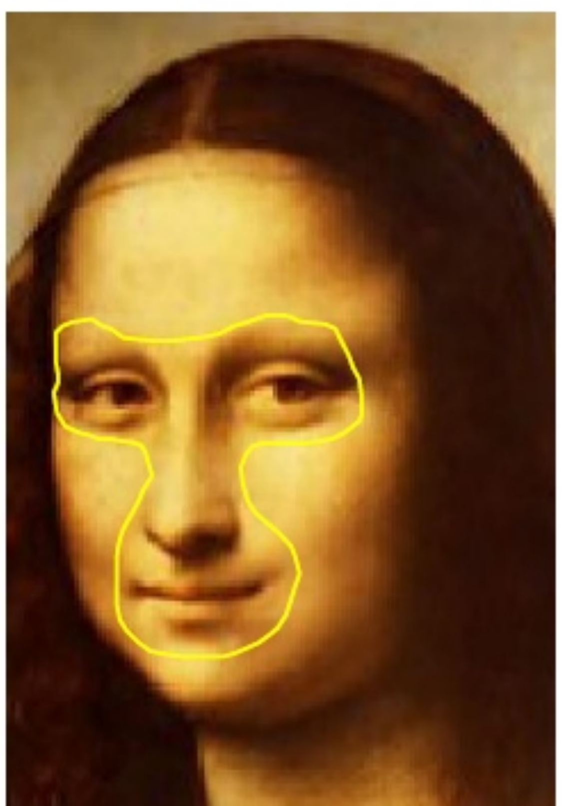


[Szeliski 2006]

Applications

Result (eye candy)





source/destination

cloning

[Pérez et al. 2003]

seamless cloning

sources destinations cloning seamless cloning

[Pérez et al. 2003]

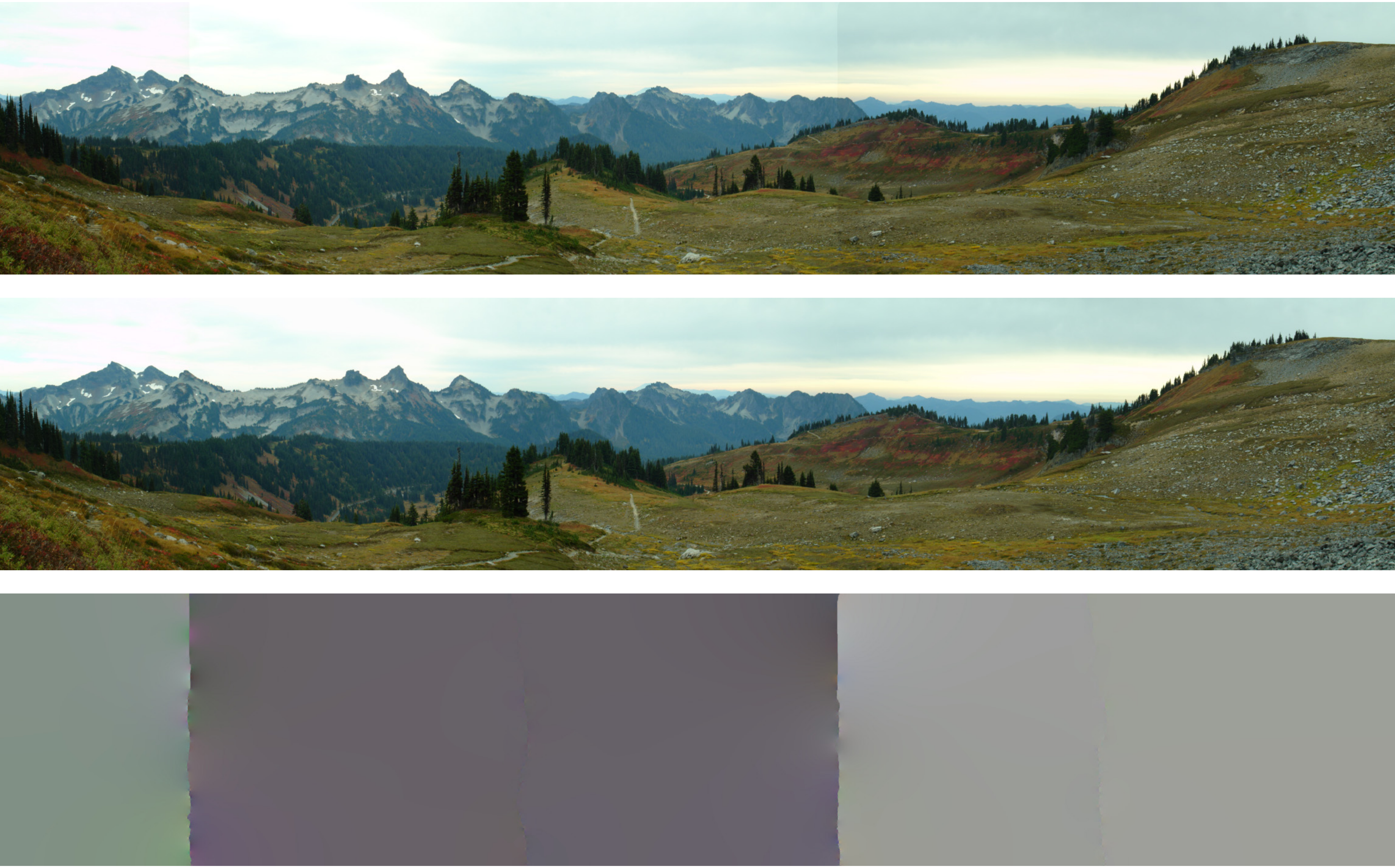
Figure 2: **Concealment**. By importing seamlessly a piece of the background, complete objects, parts of objects, and undesirable artifacts can easily be hidden. In both examples, multiple strokes (not shown) were used.

[Pérez et al. 2003]

Gradient domain HDR tone mapping

[Fattal 2002]

Gradient-domain mosaic assembly



[Agarwala 2007]

Mixed seamless cloning

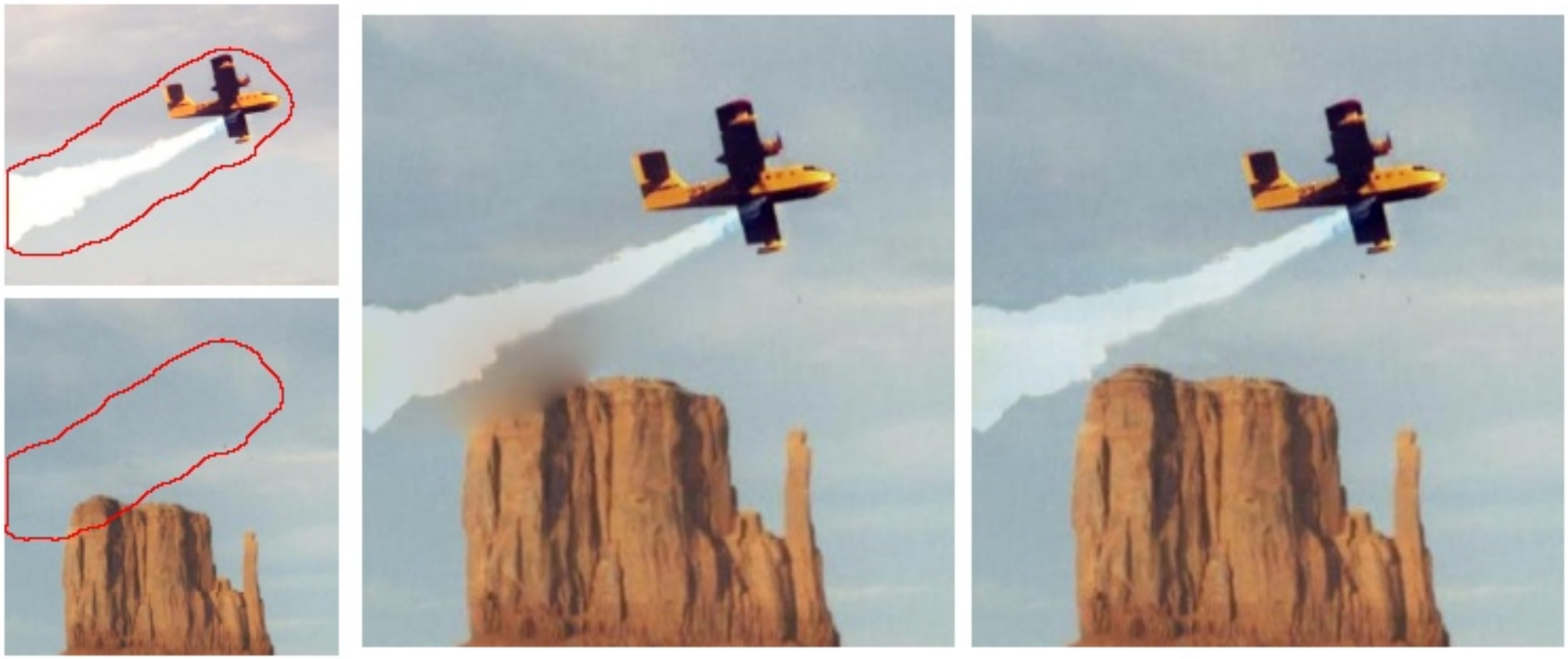
- **Rather than replacing the gradient entirely, blend the gradients using a max-like operation**

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

[Pérez et al. 2003]

Manipulate the gradient

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

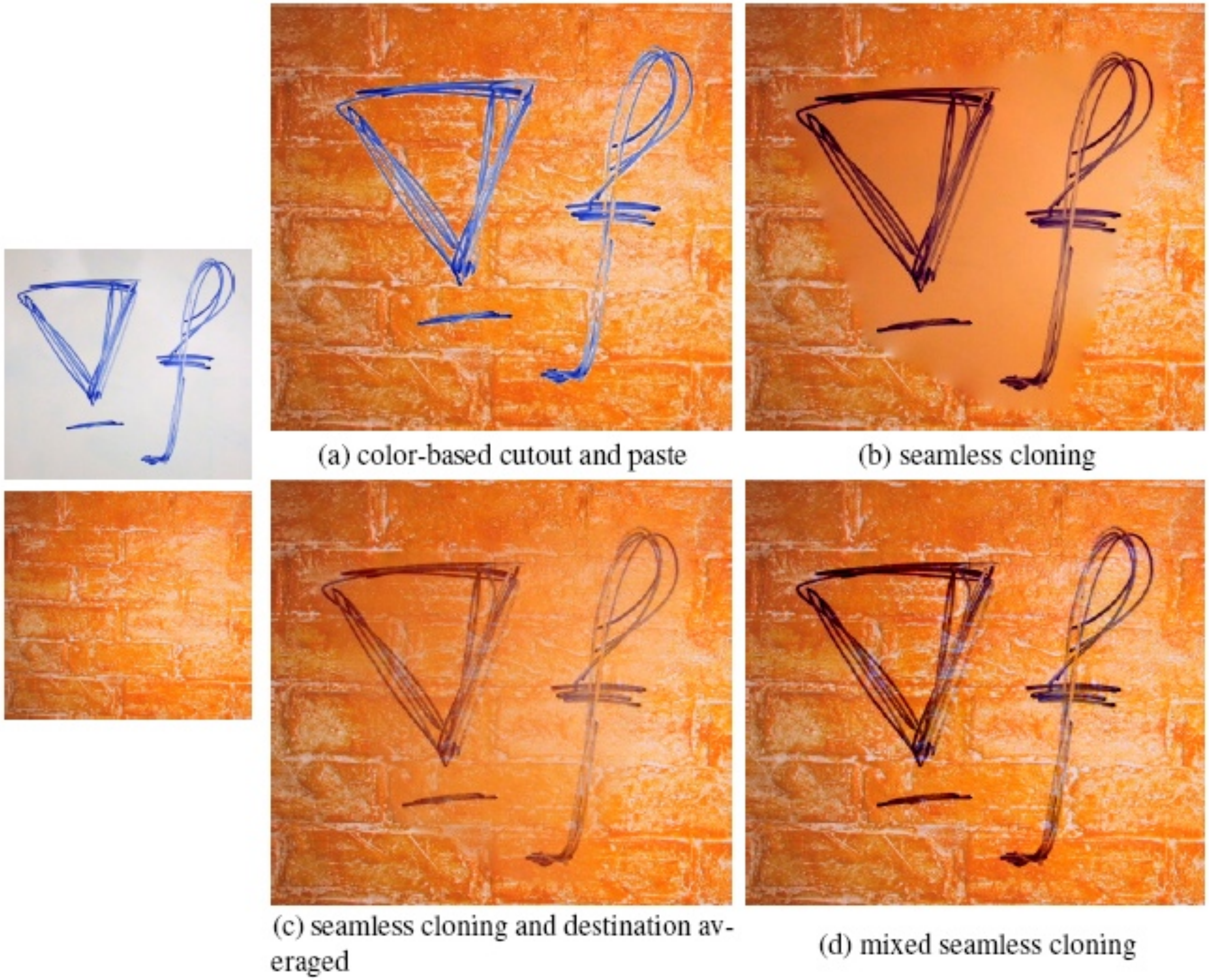


source/destination seamless cloning mixed seamless cloning

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.

[Pérez et al. 2003]

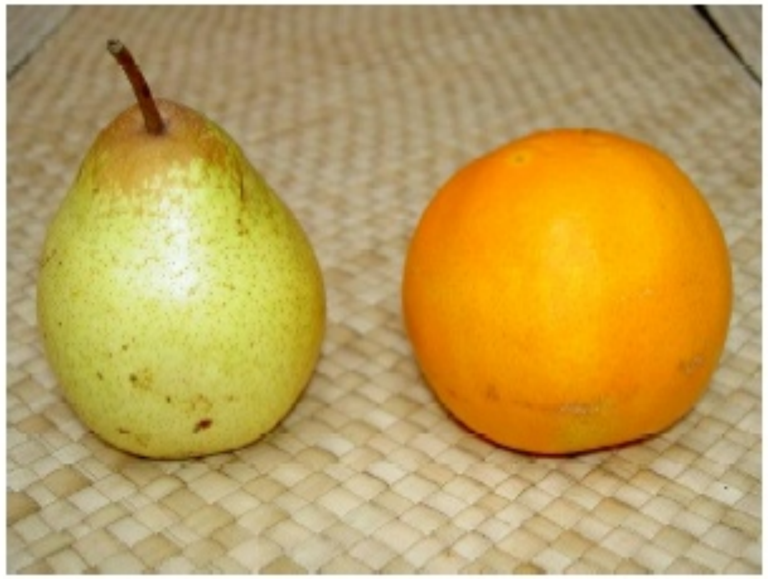
slide by Frédo Durand, MIT



[Pérez et al. 2003]

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.

slide by Frédo Durand, MIT



[Pérez et al. 2003]

swapped textures

source

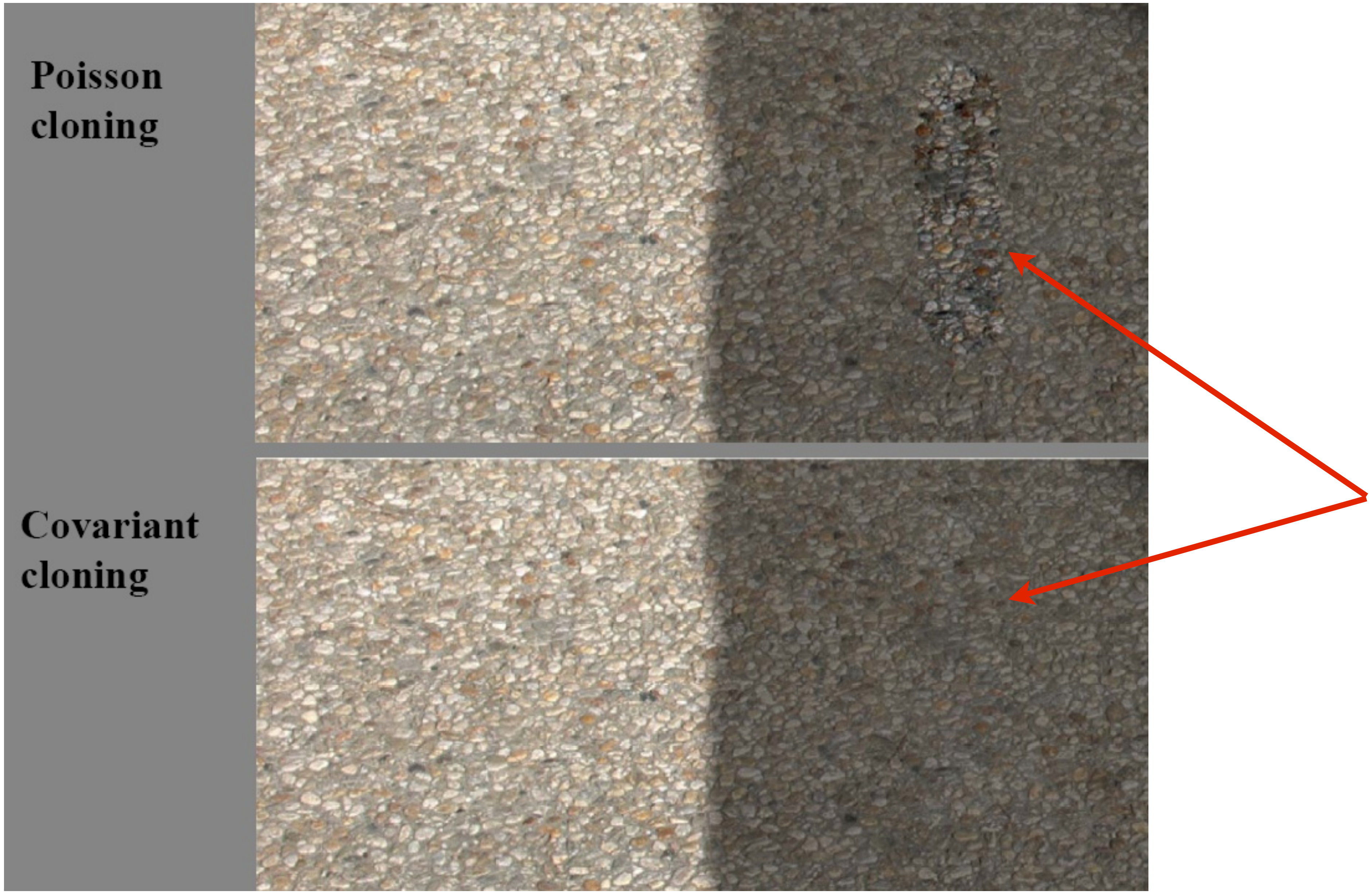
destination

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.

slide by Frédo Durand, MIT

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

slide by Frédo Durand, MIT