中国科学技术大学
University of Science and Technology of China

Gradient Image Processing

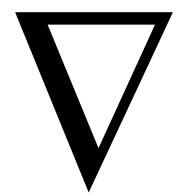
Juyong Zhang
School of Mathematics, USTC

Today: Gradient manipulation

Idea:

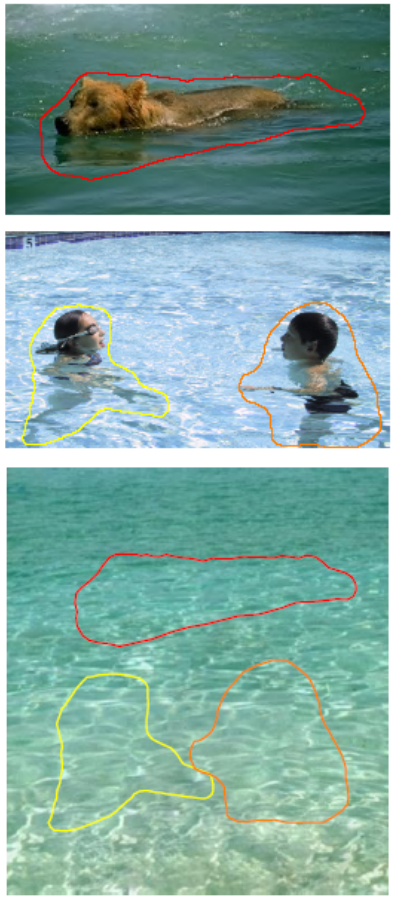
- Human visual system is very sensitive to gradient
- Gradient encode edges and local contrast quite well

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



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

Problems with direct cloning

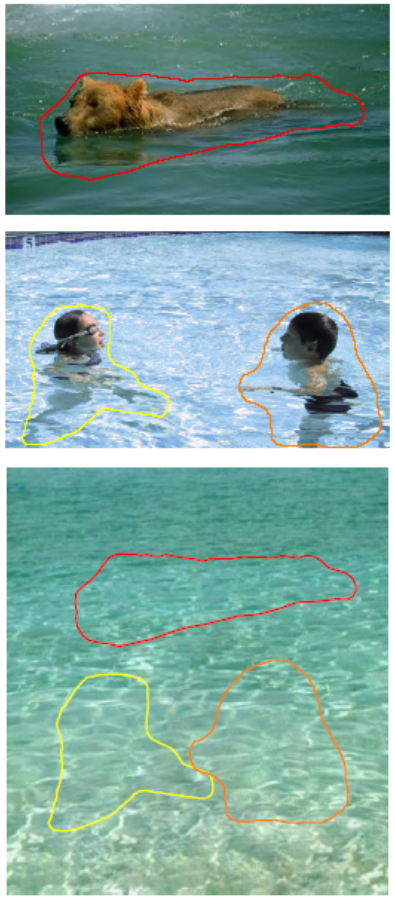


sources/destinations

cloning

From Perez et al. 2003

Solution: clone gradient



sources/destinations

seamless cloning

Gradients and grayscale images

- Grayscale image: $n \times n$ scalars
- Gradient: $n \times n$ 2D vectors
- Overcomplete!
- 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 it does not matter for us

Today message I

- Manipulating the gradient is powerful

Today message II

- Optimization is powerful
  - In particular least square

- Good Least square optimization reduces to a big linear system
- Linear algebra is your friend
  - Big sparse linear systems can be solved efficiently

Today message III

- Toy examples are good to further understanding
- 1D can however be overly simplifying, n-D is much more complicated

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

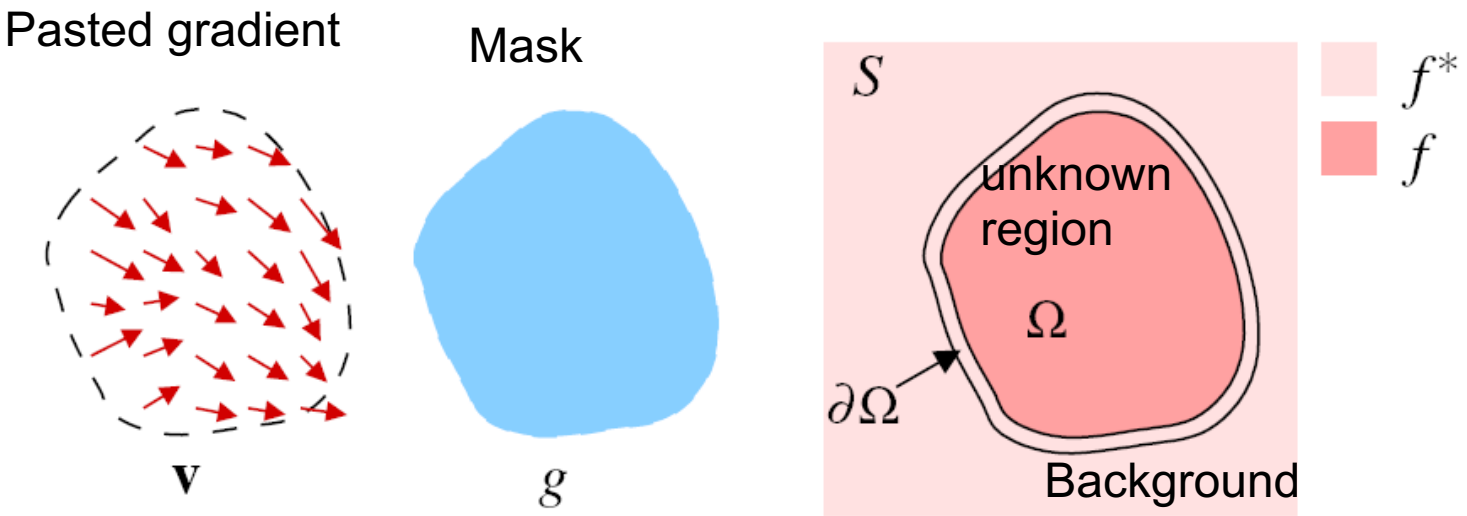
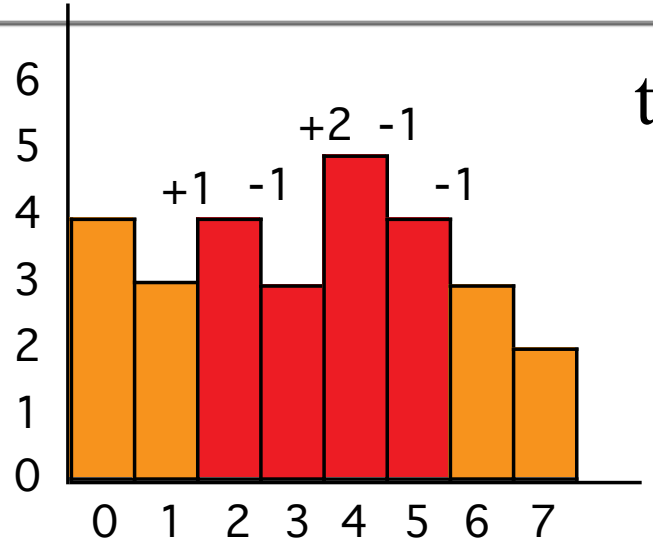


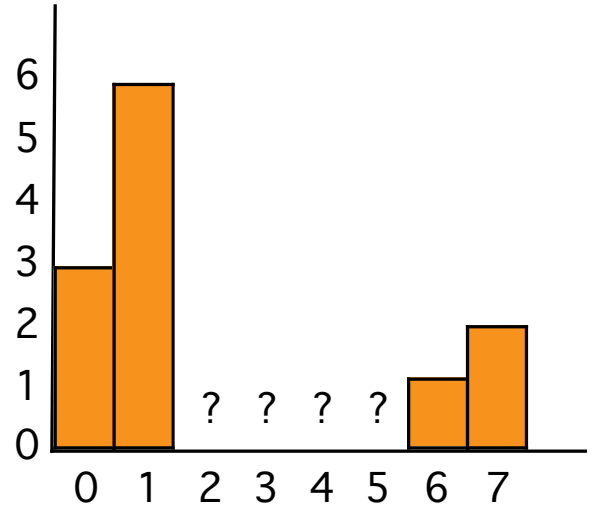
Figure 1: **Guided interpolation notations**. Unknown function f interpolates in domain Ω the destination function f^*, under guidance of vector field $\mathbf{v}$, which might be or not the gradient field of a source function g.

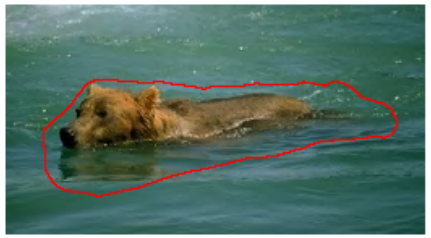
Discrete 1D example: minimization

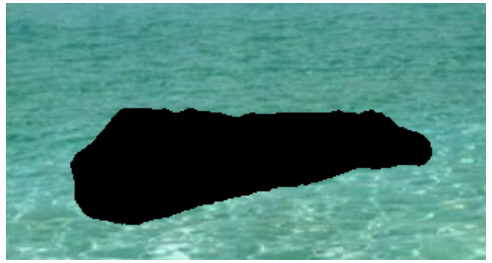
- Copy



to





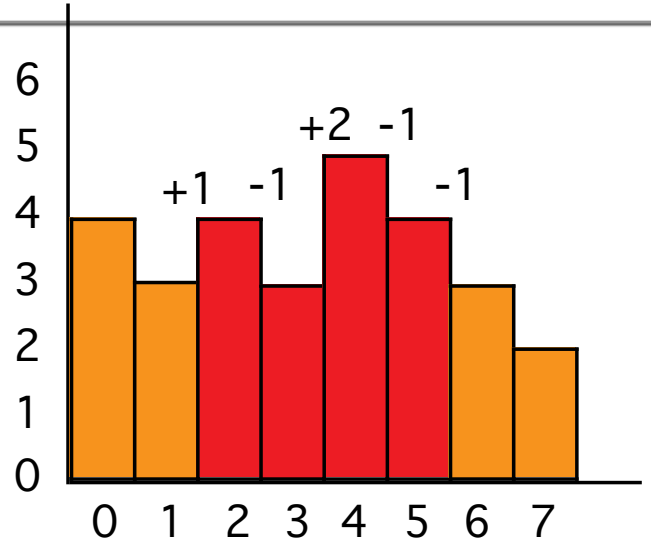


- Min $((f_2-f_1)-1)^2$
- Min $((f_3-f_2)-(-1))^2$
- Min $((f_4-f_3)-2)^2$
- Min $((f_5-f_4)-(-1))^2$
- Min $((f_6-f_5)-(-1))^2$

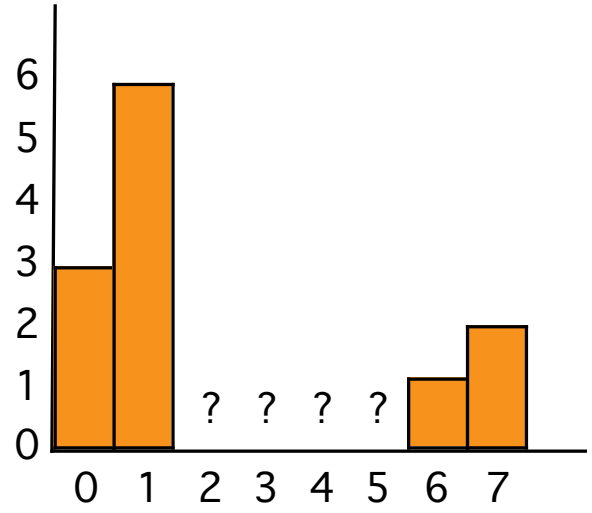
With
$f_1=6$
$f_6=1$

1D example: minimization

- Copy

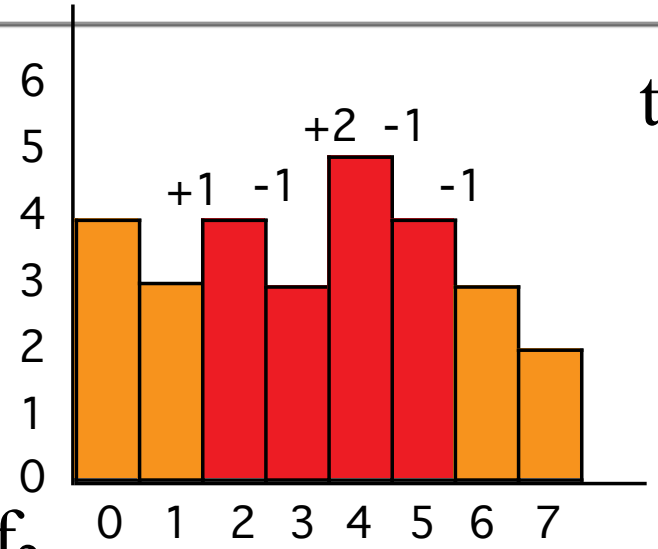


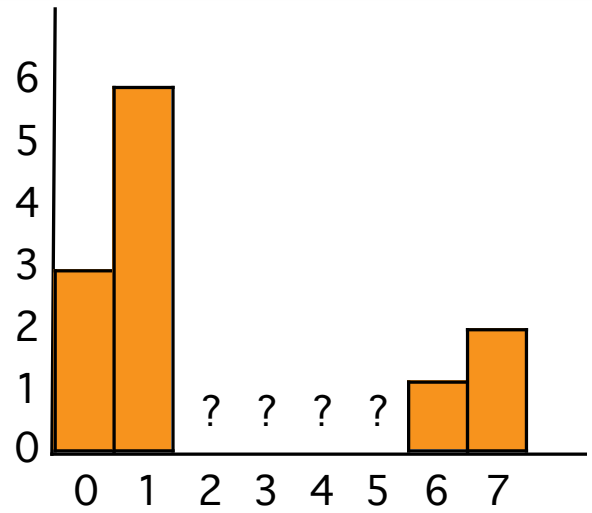
to



- Min $((f_2-6)-1)^2$ ==> $f_2^2+49-14f_2$
- Min $((f_3-f_2)-(-1))^2$ ==> $f_3^2+f_2^2+1-2f_3f_2+2f_3-2f_2$
- Min $((f_4-f_3)-2)^2$ ==> $f_4^2+f_3^2+4-2f_3f_4-4f_4+4f_3$
- Min $((f_5-f_4)-(-1))^2$ ==> $f_5^2+f_4^2+1-2f_5f_4+2f_5-2f_4$
- Min $((1-f_5)-(-1))^2$ ==> $f_5^2+4-4f_5$

1D example: big quadratic

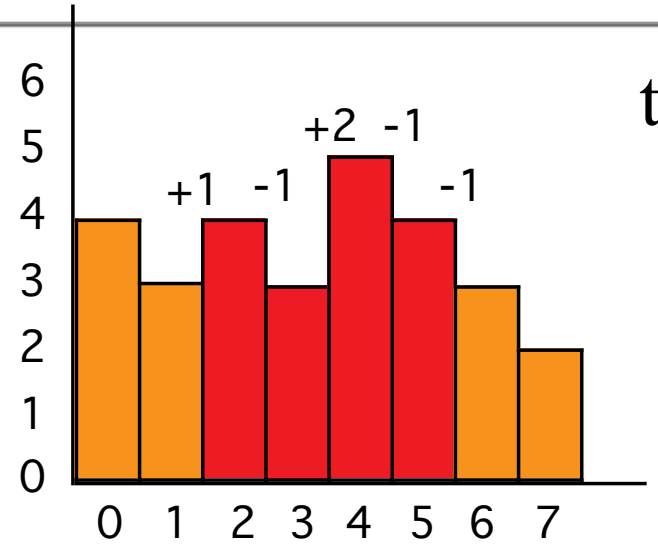
- Copy 

to 

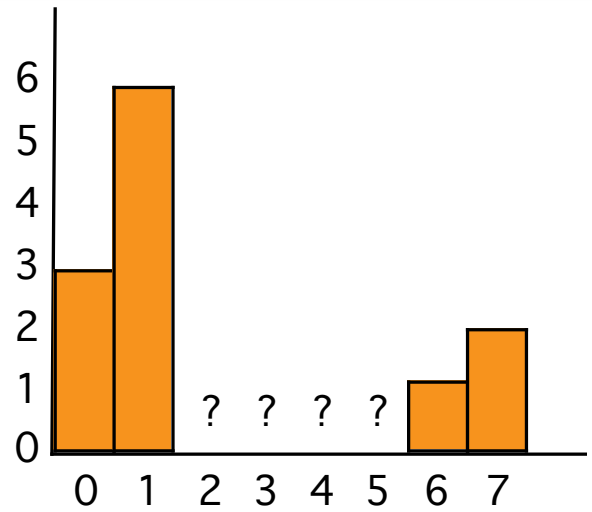
- Min (f_2^2+49-14f_2
  + f_3^2+f_2^2+1-2f_3f_2 +2f_3-2f_2
  + f_4^2+f_3^2+4-2f_3f_4 -4f_4+4f_3
  + f_5^2+f_4^2+1-2f_5f_4 +2f_5-2f_4
  + f_5^2+4-4f_5)
  Denote it Q

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

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

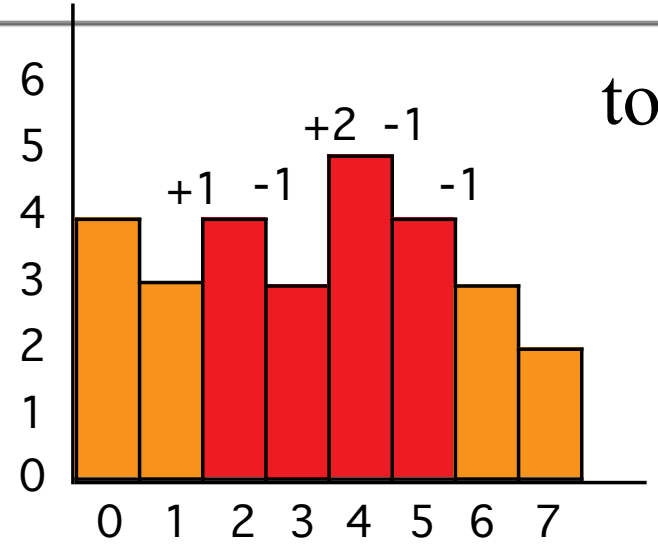
$$\frac{dQ}{df_3} = 2f_3 - 2f_2 + 2 + 2f_3 - 2f_4 + 4$$

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

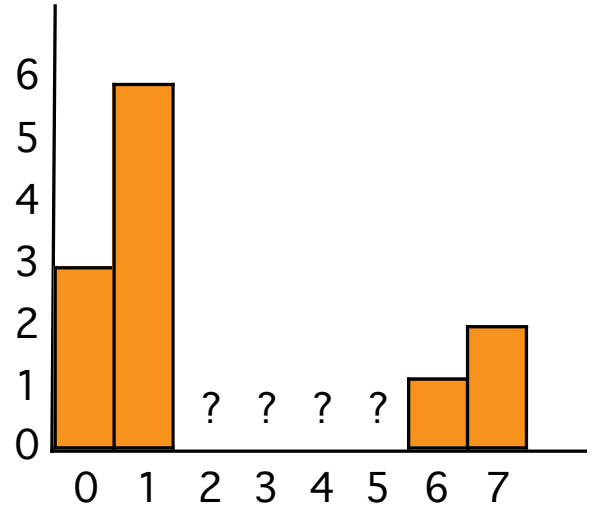
$$\frac{dQ}{df_5} = 2f_5 - 2f_4 + 2 + 2f_5 - 4$$

1D example: set derivatives to zero

- Copy



to



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

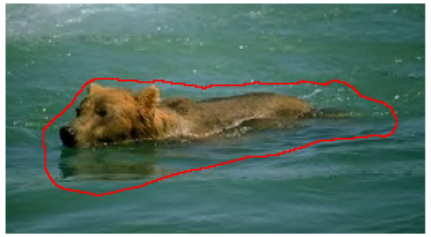
$$\frac{dQ}{df_3} = 2f_3 - 2f_2 + 2 + 2f_3 - 2f_4 + 4$$

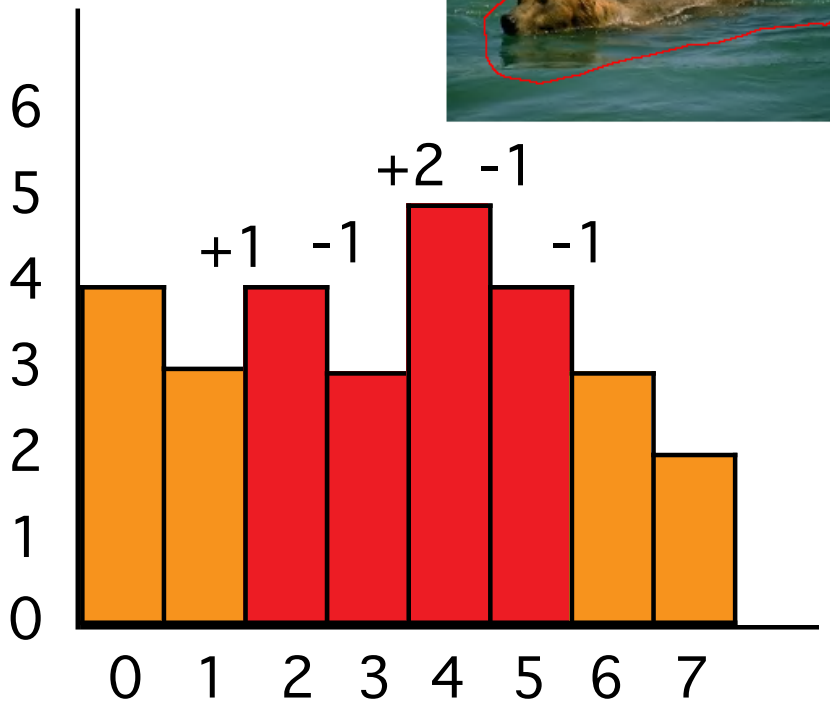
$$\frac{dQ}{df_4} = 2f_4 - 2f_3 - 4 + 2f_4 - 2f_5 - 2$$

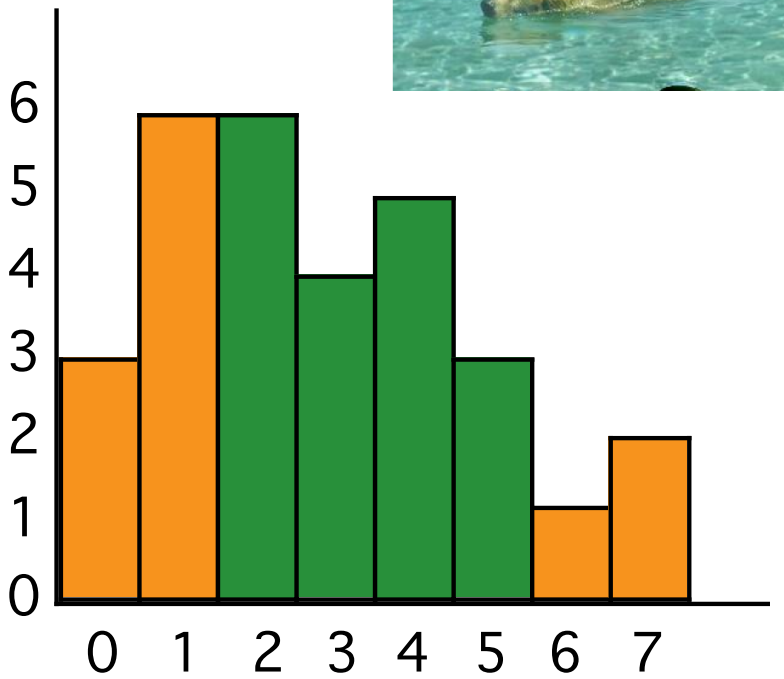
$$\frac{dQ}{df_5} = 2f_5 - 2f_4 + 2 + 2f_5 - 4$$

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

1D example

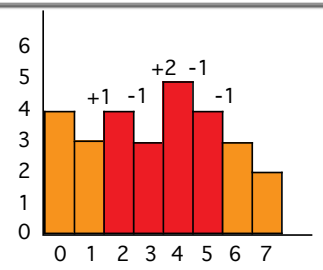
- Copy  to 

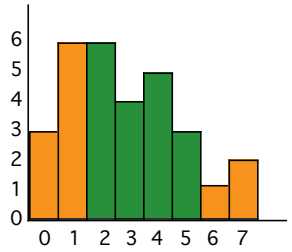




$$\begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 16 \\ -6 \\ 6 \\ 2 \end{pmatrix} \qquad \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \\ 5 \\ 3 \end{pmatrix}$$

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
- Matrix is symmetric
- Everything is a multiple of 2
  - because square and derivative of square
- Matrix is a convolution (kernel -2 4 -2)
- Matrix is independent of gradient field. Only RHS is
- Matrix is a second derivative

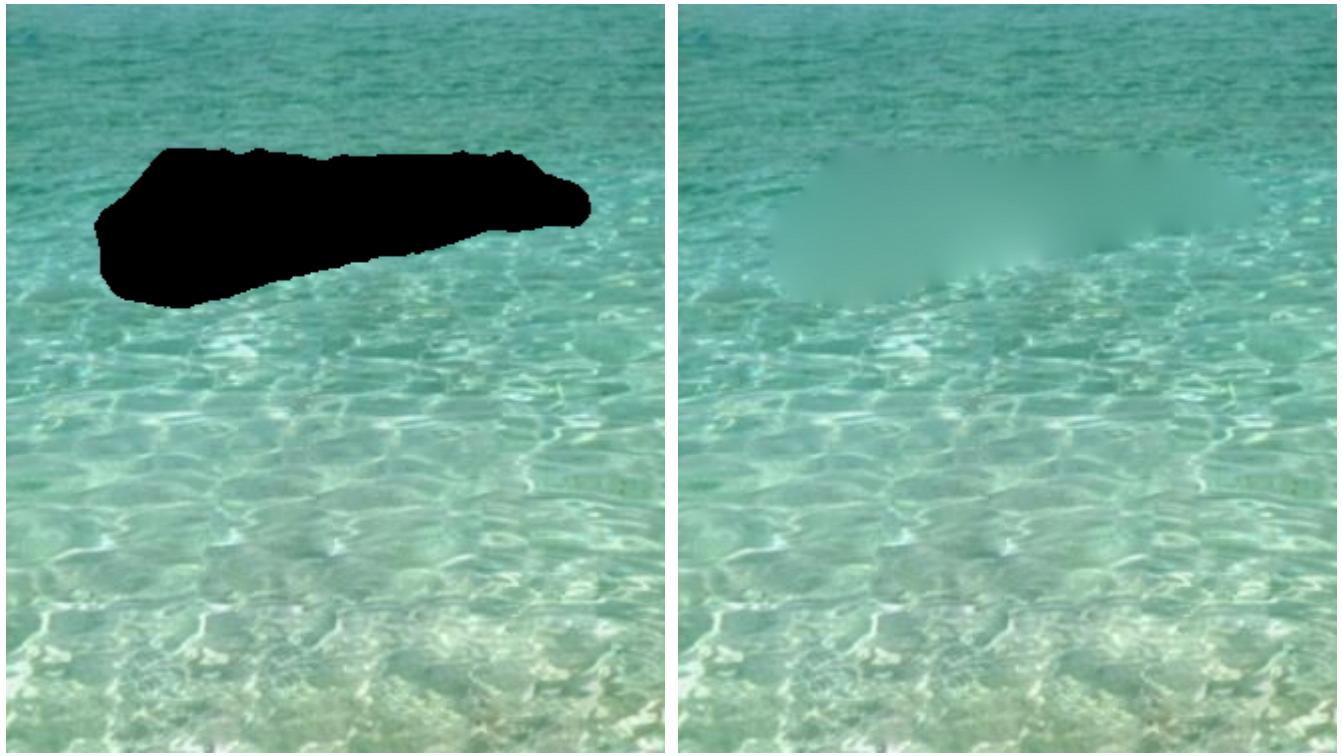
Let's try to further analyze

- What is a simple case?

Membrane interpolation

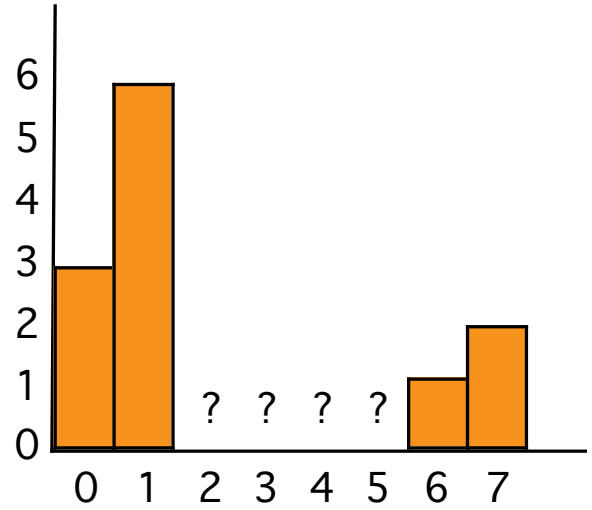
- What if v is null?
- Laplace equation (a.k.a. membrane equation)

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



1D example: minimization

- Minimize derivatives to interpolate

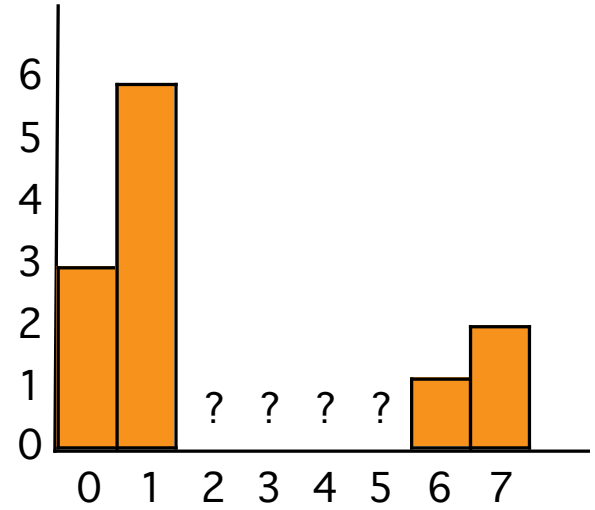


- Min $(f_2-f_1)^2$
- Min $(f_3-f_2)^2$
- Min $(f_4-f_3)^2$
- Min $(f_5-f_4)^2$
- Min $(f_6-f_5)^2$

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

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

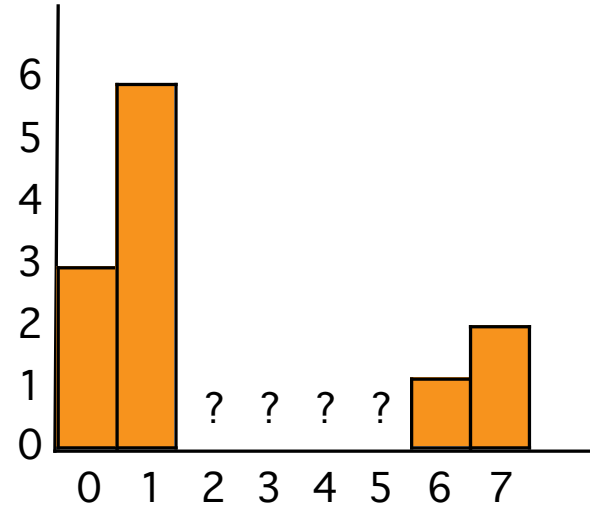
$$\frac{dQ}{df_3} = 2f_3 - 2f_2 + 2f_3 - 2f_4$$

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

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

1D example: set derivatives to zero

- Minimize derivatives to interpolate



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

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

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

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

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

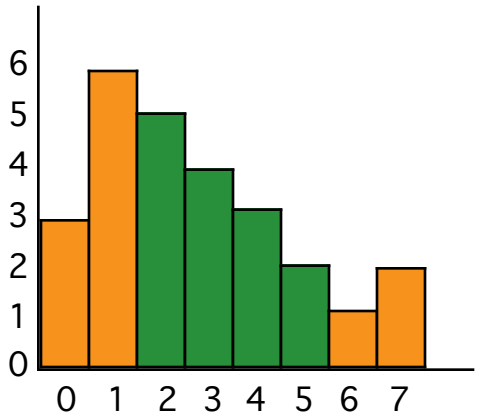
1D example

- Minimize derivatives to interpolate

- Pretty much says that second derivative should be zero

  (-1 2 -1)

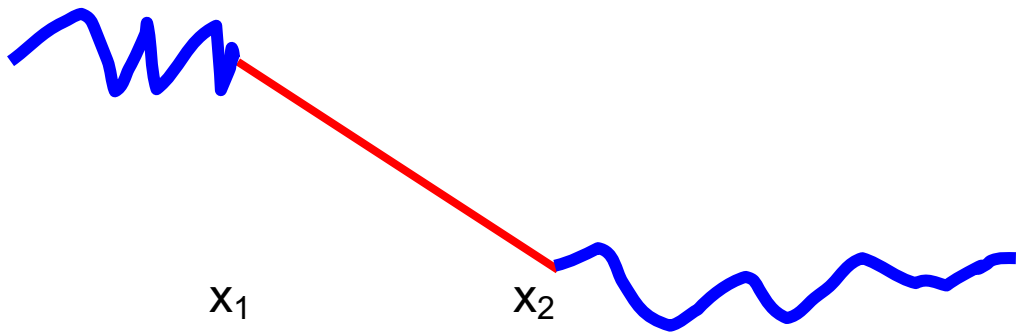
  is a second derivative filter



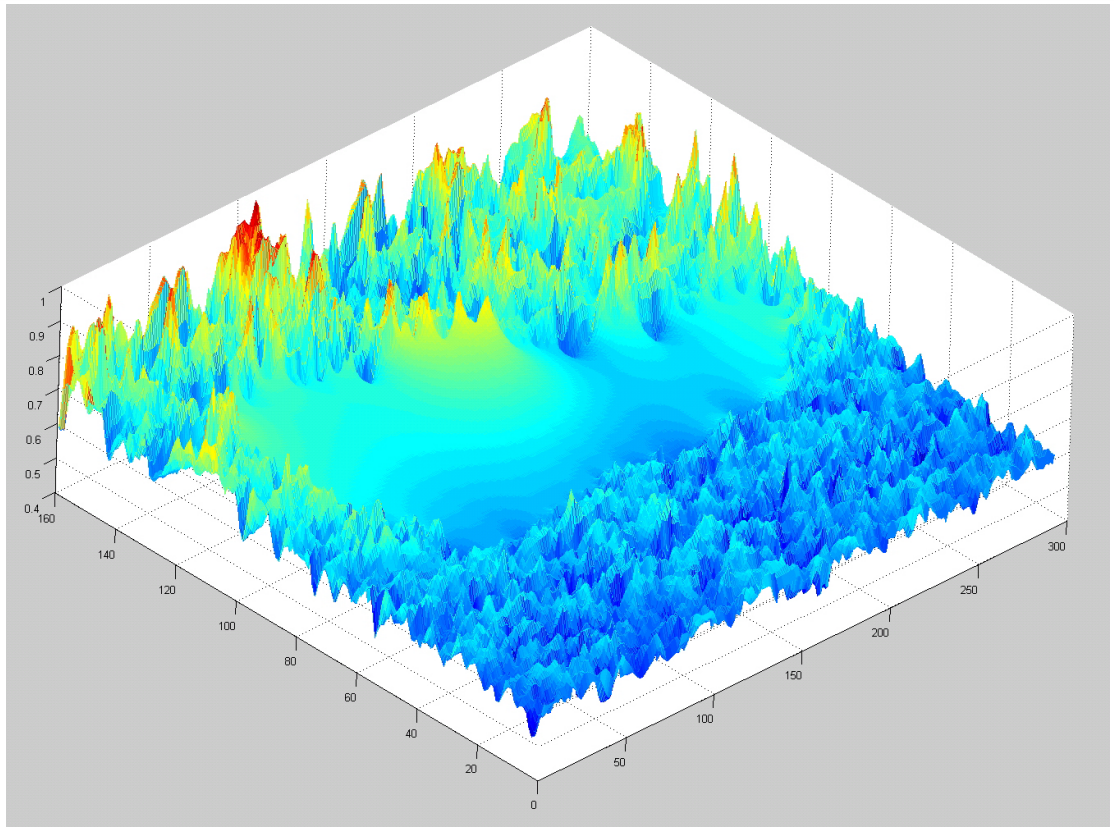
$$\begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 12 \\ 0 \\ 0 \\ 2 \end{pmatrix} \qquad \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 5 \\ 4 \\ 3 \\ 2 \end{pmatrix}$$

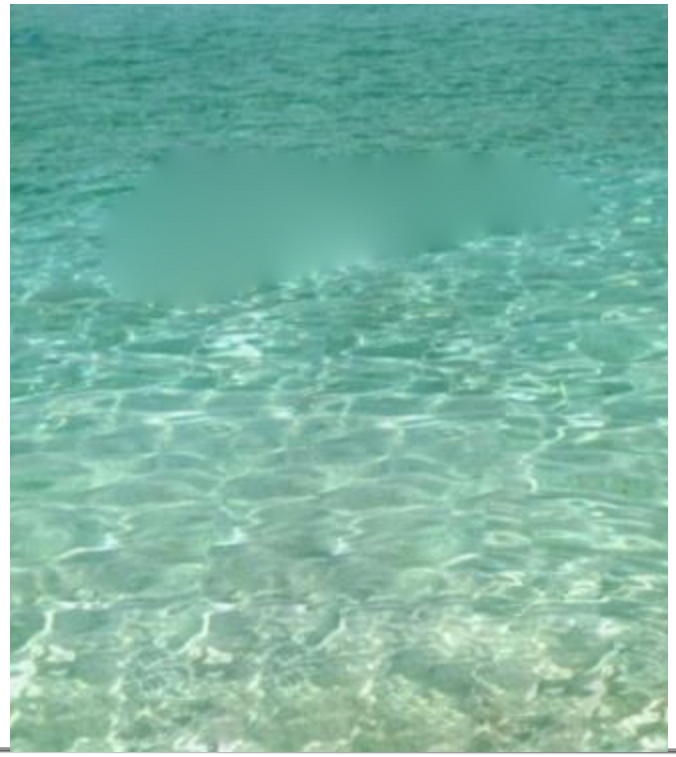
Intuition

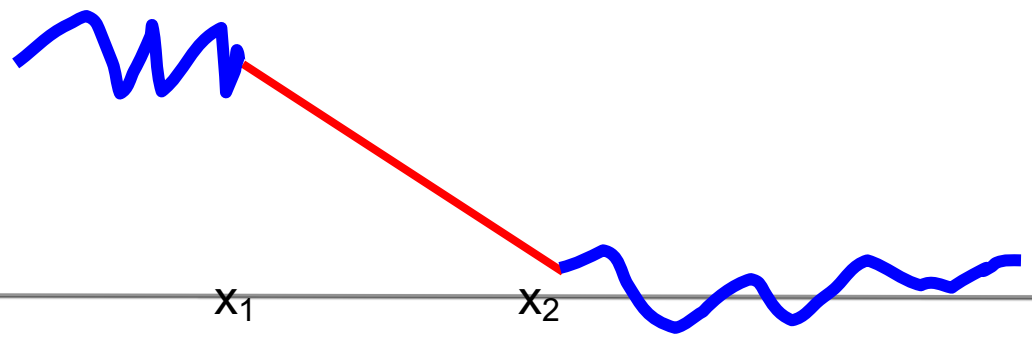
- In 1D; just linear interpolation!
  - The min of $\int f'$ is the slope integrated over the interval
- Locally, if the second derivative was not zero, this would mean that the first derivative is varying, which is bad since we want $\int f'$ to be minimized
- Note that, in 1D: by setting f'', we leave 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



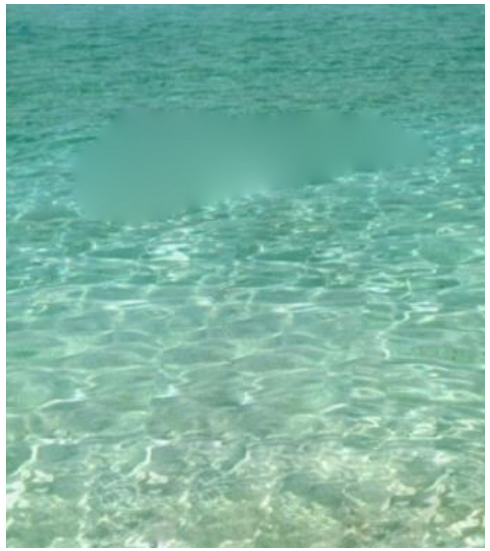




Membrane interpolation

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

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

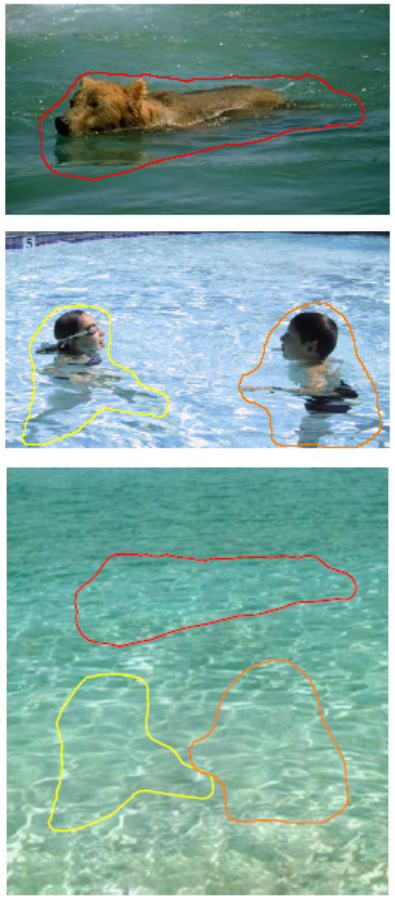


- Mathematicians will tell you there is an Associated Euler-Lagrange equation:

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

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

What is v is not null?

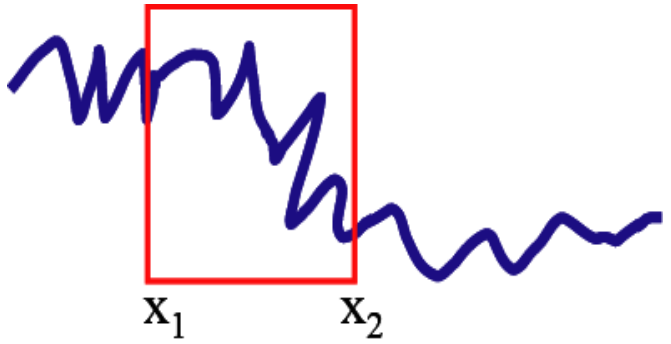


sources/destinations

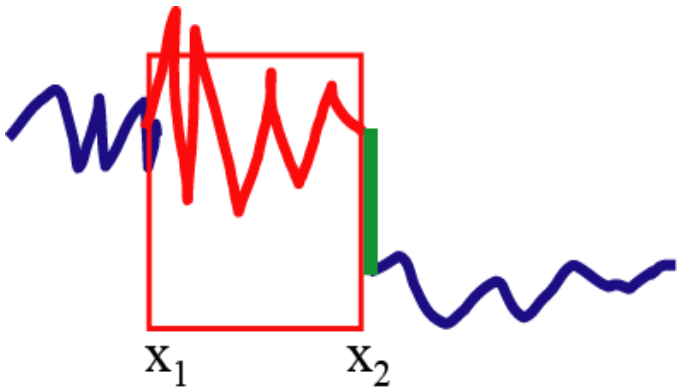
seamless cloning

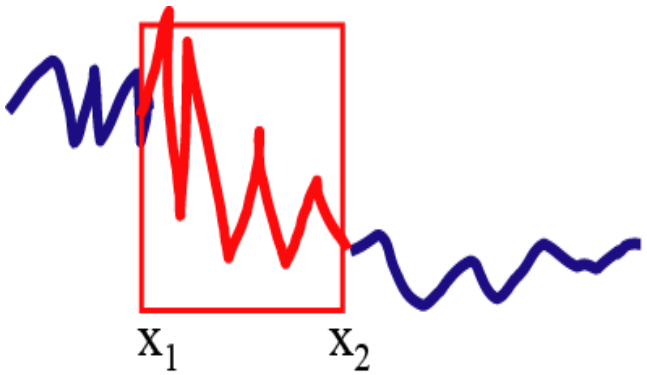
What if v is not null?

- 1D case

Seamlessly paste  onto 

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





(Review) 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

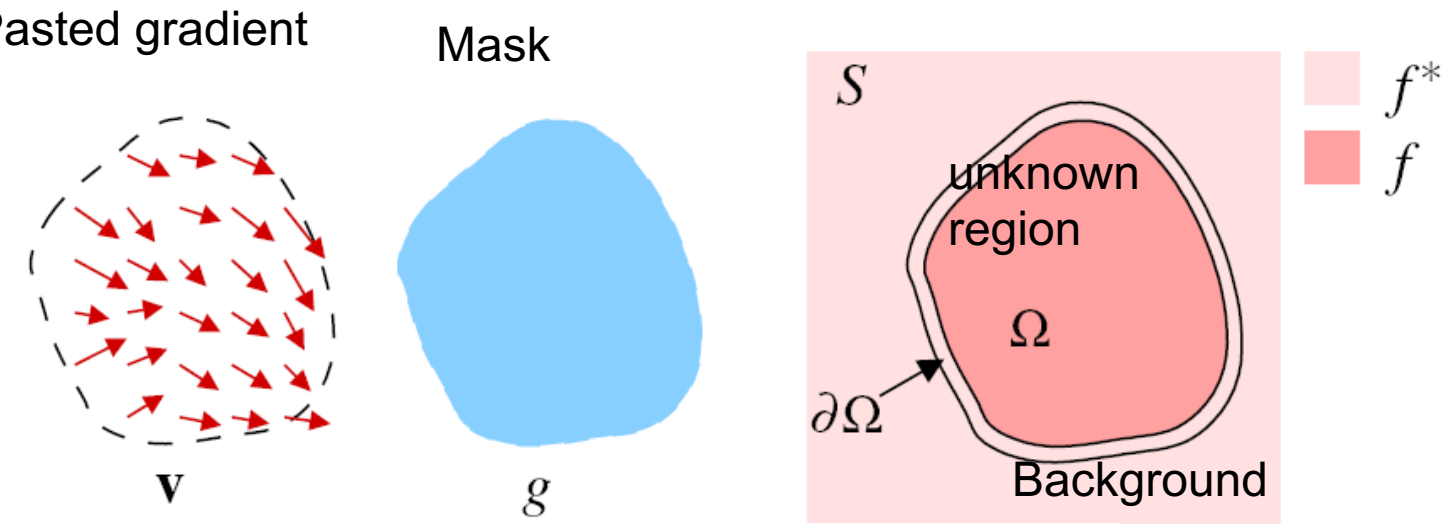


Figure 1: **Guided interpolation notations**. Unknown function f interpolates in domain Ω the destination function f^*, under guidance of vector field $\mathbf{v}$, which might be or not the gradient field of a source function g.

What if *v* is not null: 2D

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

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

- Euler-Lagrange equation:

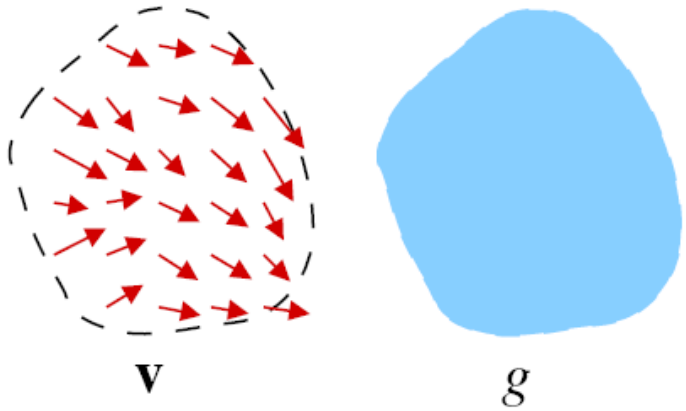
$$\Delta f = \text{div}\mathbf{v} \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

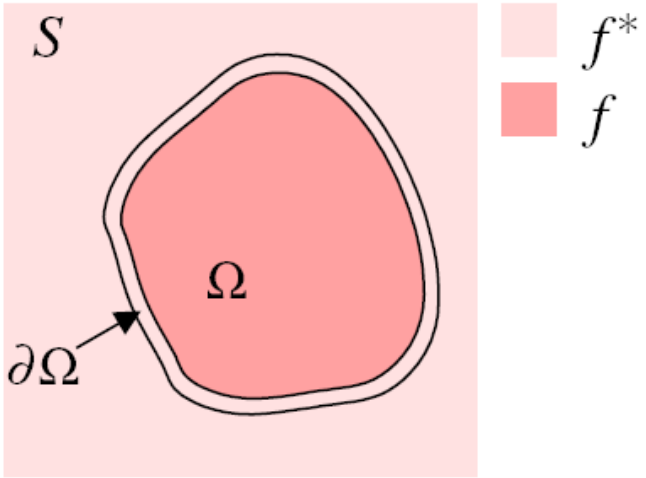
$$\text{where } \text{div}\mathbf{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \text{ is the divergence of } \mathbf{v} = (u, v)$$

In 2D, if v is conservative

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

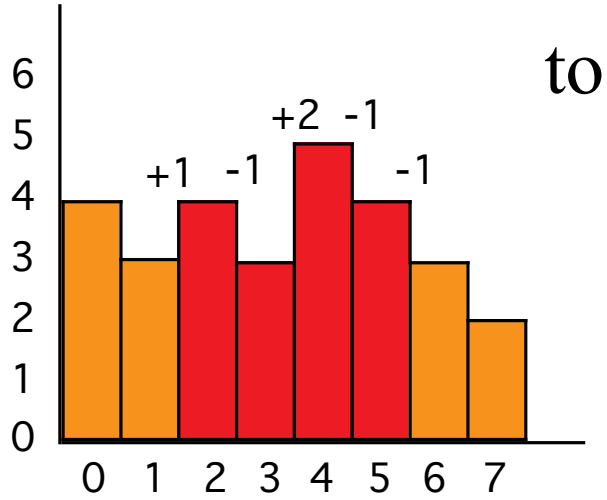
$$\Delta \tilde{f} = 0 \text{ over } \Omega, \ \tilde{f}|_{\partial\Omega} = (f^* - g)|_{\partial\Omega}$$



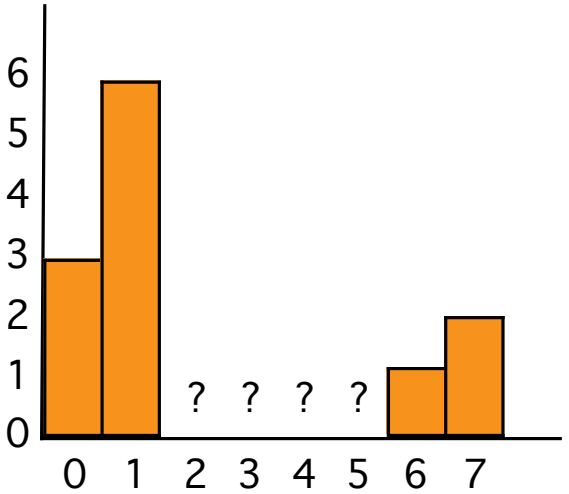


1D example

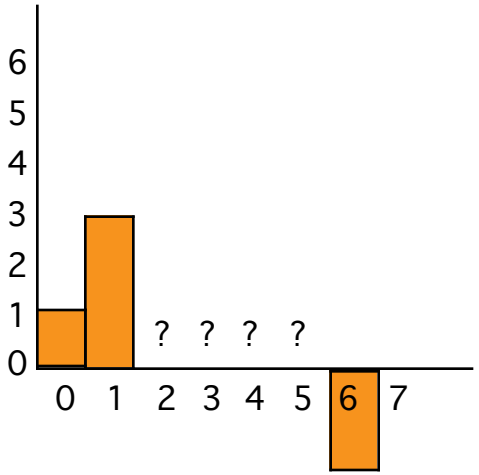
- Copy



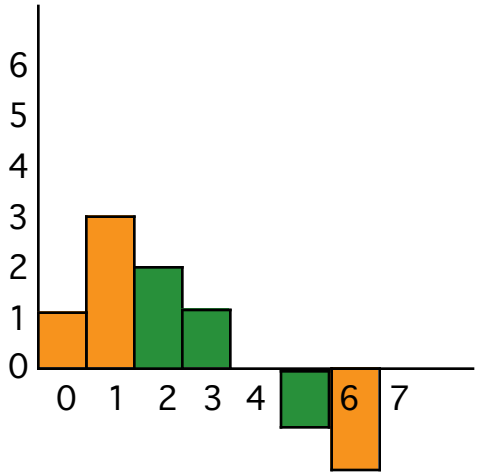
to



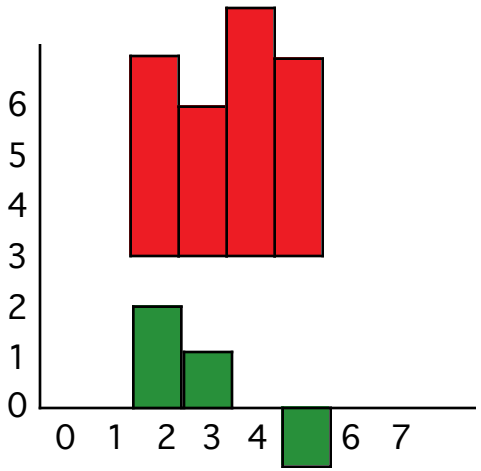
Difference



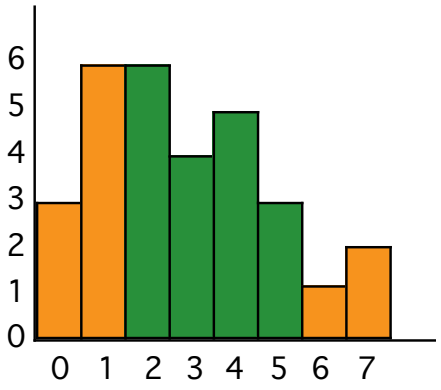
Solve Laplace



Add

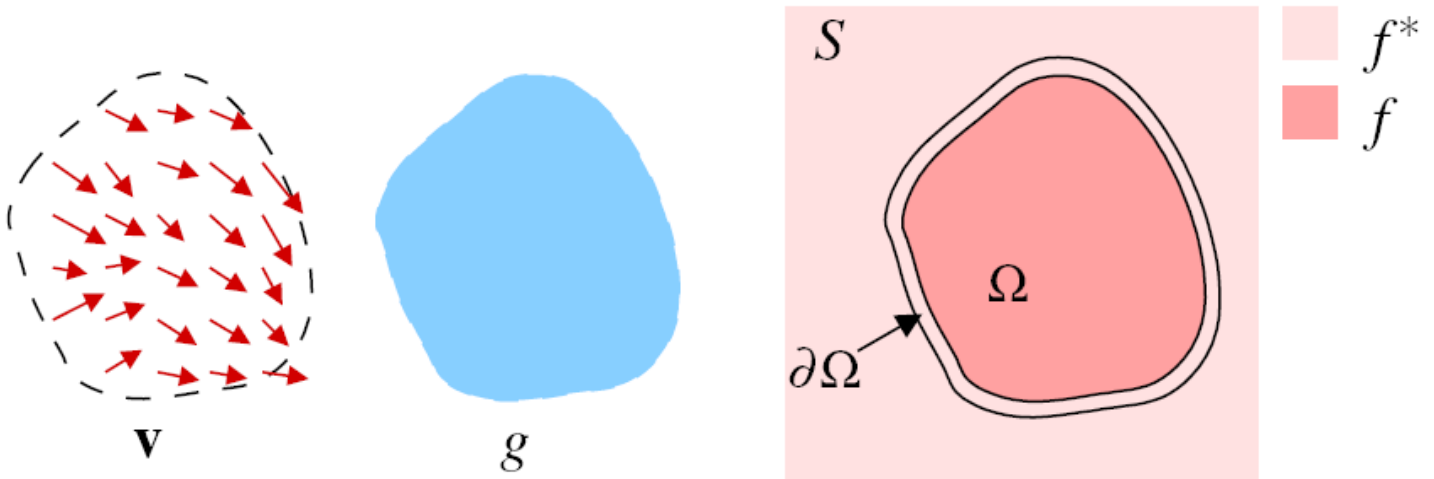


Result



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

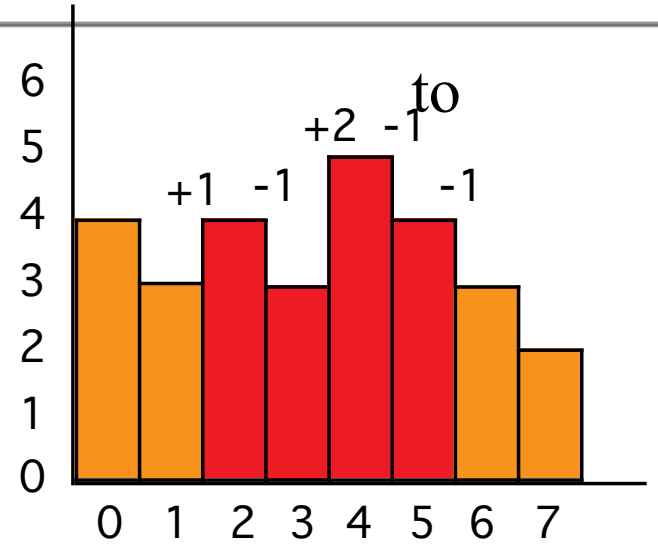


Recap

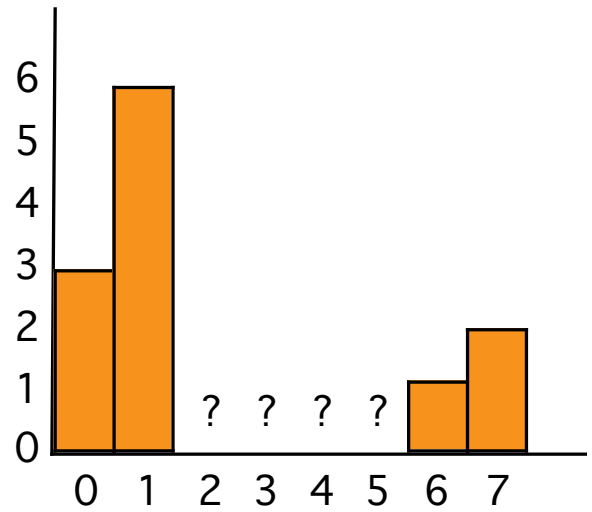
- Find image whose gradient best approximates the input gradient
  - least square Minimization
- Discrete case: turns into linear equation
  - Set derivatives to zero
  - Derivatives of quadratic ==> linear
- Continuous: turns into Euler-Lagrange form
  - $\Delta f = \text{div } v$
- When gradient is null, membrane interpolation
  - Linear interpolation in 1D

Discrete solver: Recall 1D

- Copy



to



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

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

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

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

==>

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

Discrete Poisson solver

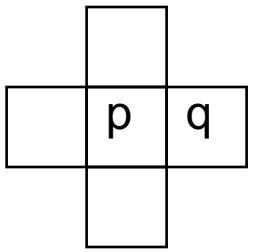
- Two approaches:
  - Minimize variational problem
  - Solve Euler-Lagrange equation

  In practice, variational is best

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

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

- In both cases, need to discretize derivatives
  - Finite differences over 4 pixel neighbors
  - We are going to work using pairs
    - Partial derivatives are easy on pairs
    - Same for the discretization of v



Discrete Poisson solver

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

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)

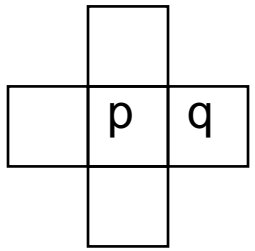
Boundary condition

- Rearrange and call N_p the neighbors of p

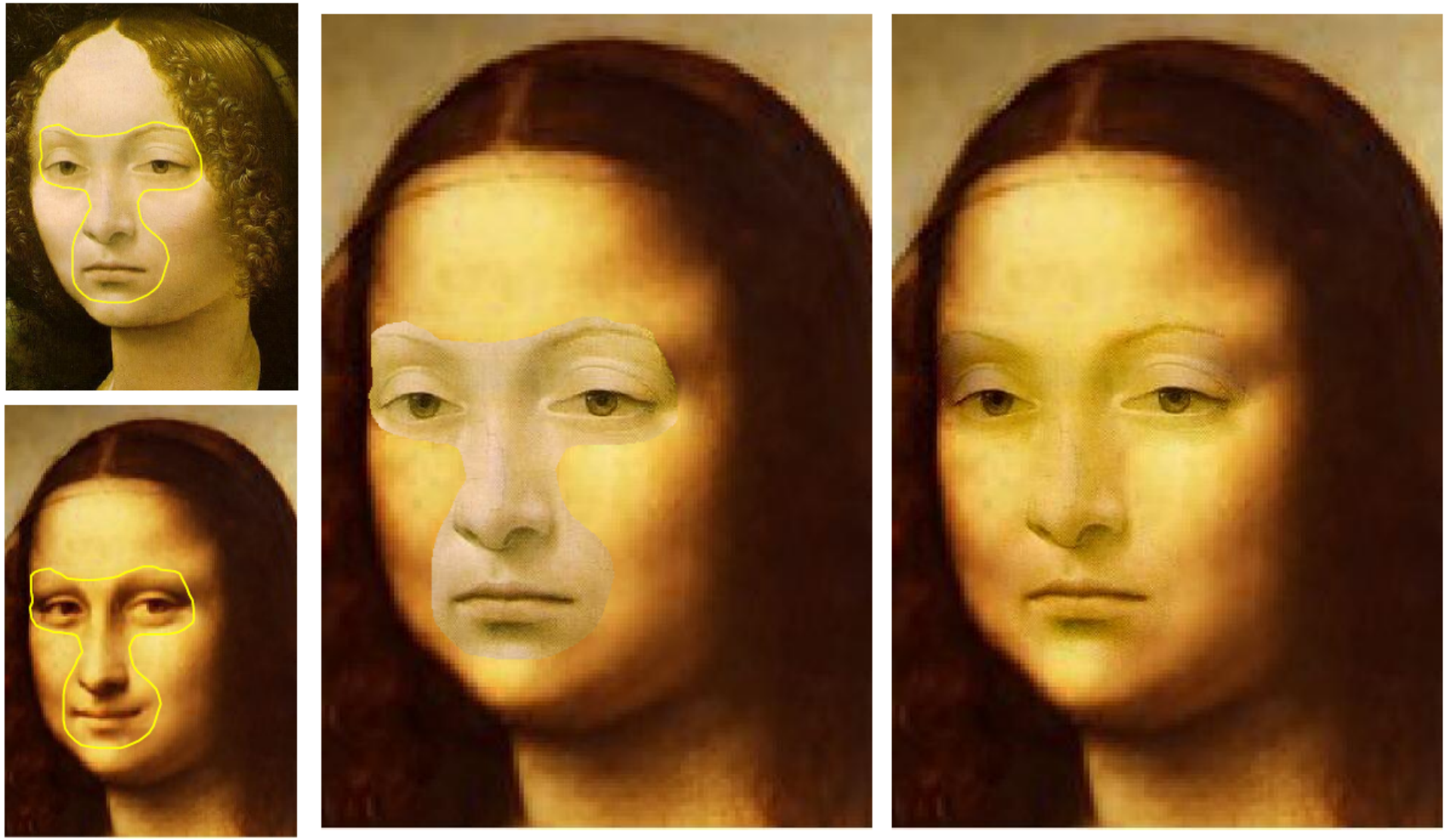
$$\text{for all } p \in \Omega, \quad |N_p| f_p - \sum_{q \in N_p \cap \Omega} f_q = \sum_{q \in N_p \cap \partial\Omega} f_q^* + \sum_{q \in N_p} v_{pq}$$

Only for boundary pixels

- Big yet sparse linear system



Result (eye candy)



source/destination

cloning

seamless cloning

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

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

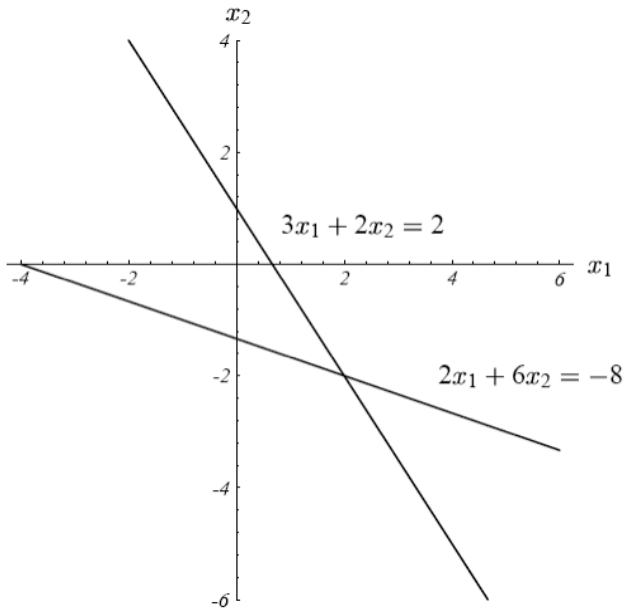
Solving big matrix systems

- Ax=b, where A is sparse (many zero entries)
- In Pset 3, we ask you to use conjugate gradient
  - http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf
  - http://www.library.cornell.edu/nr/bookcpdf/c10-6.pdf

Ax=b

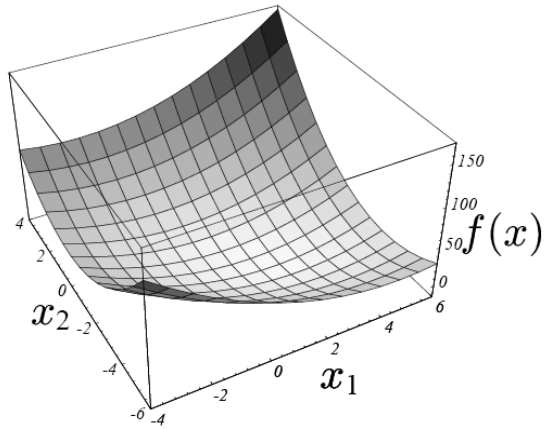
- A is square, symmetric and positive-definite
- When A is dense, you're stuck, use backsubstitution
- When A is sparse, iterative techniques (such as Conjugate Gradient) are faster and more memory efficient
- Simple example:

$$\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} x = \begin{bmatrix} 2 \\ -8 \end{bmatrix}$$

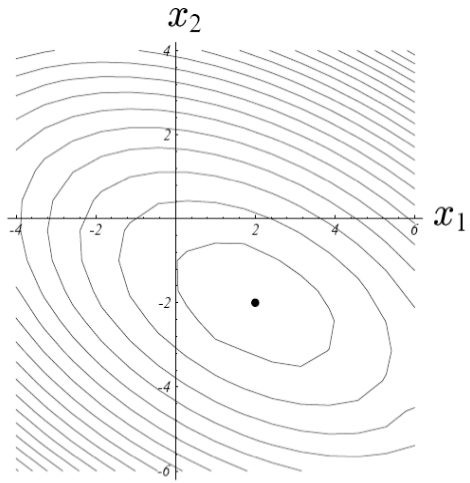


Turn Ax=b into a minimization problem

- 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



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 $f(x)$.

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:

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

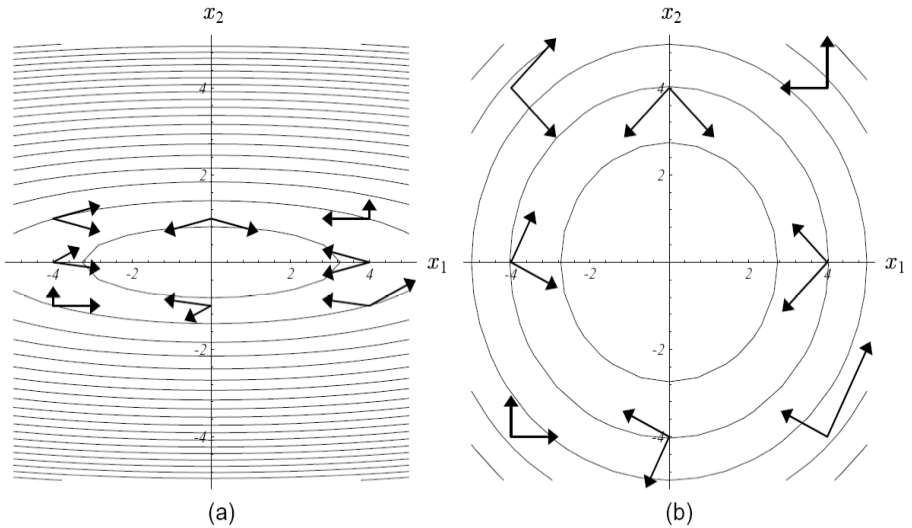


Figure 22: These pairs of vectors are A-orthogonal . . . because these pairs of vectors are orthogonal.

Recap

- Poisson image cloning: paste gradient, enforce boundary condition
- Variational formulation
- Also Euler-Lagrange formulation
- Discretize variational version,
  leads to big but sparse linear system
- Conjugate gradient is a smart iterative technique to solve it

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

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

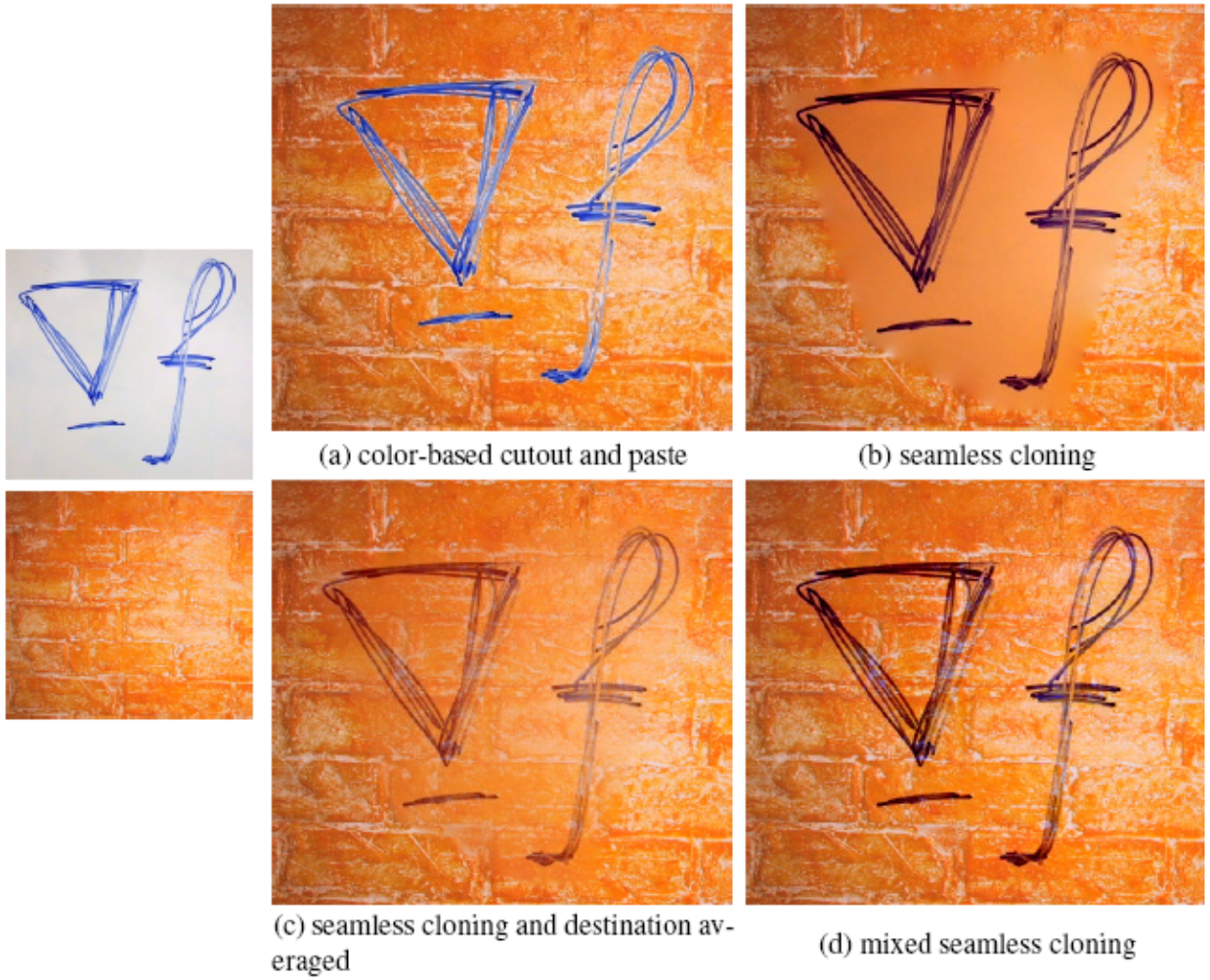
Figure 2: **Concealment**. By importing seamlessly a piece of the background, complete objects, parts of objects, and undesirable artifacts can easily be hidden. In both examples, multiple strokes (not shown) were used.

sources destinations cloning seamless cloning

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.



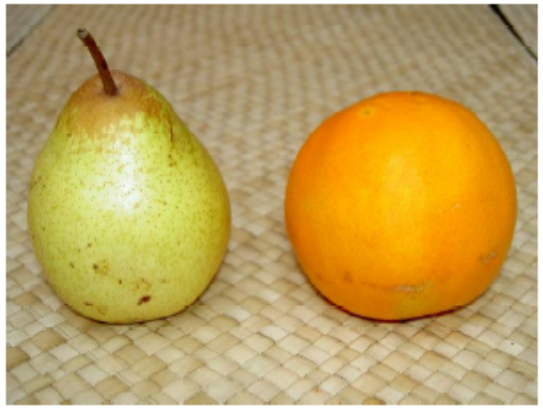
(a) color-based cutout and paste

(b) seamless cloning

(c) seamless cloning and destination averaged

(d) mixed seamless cloning

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.

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.

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)

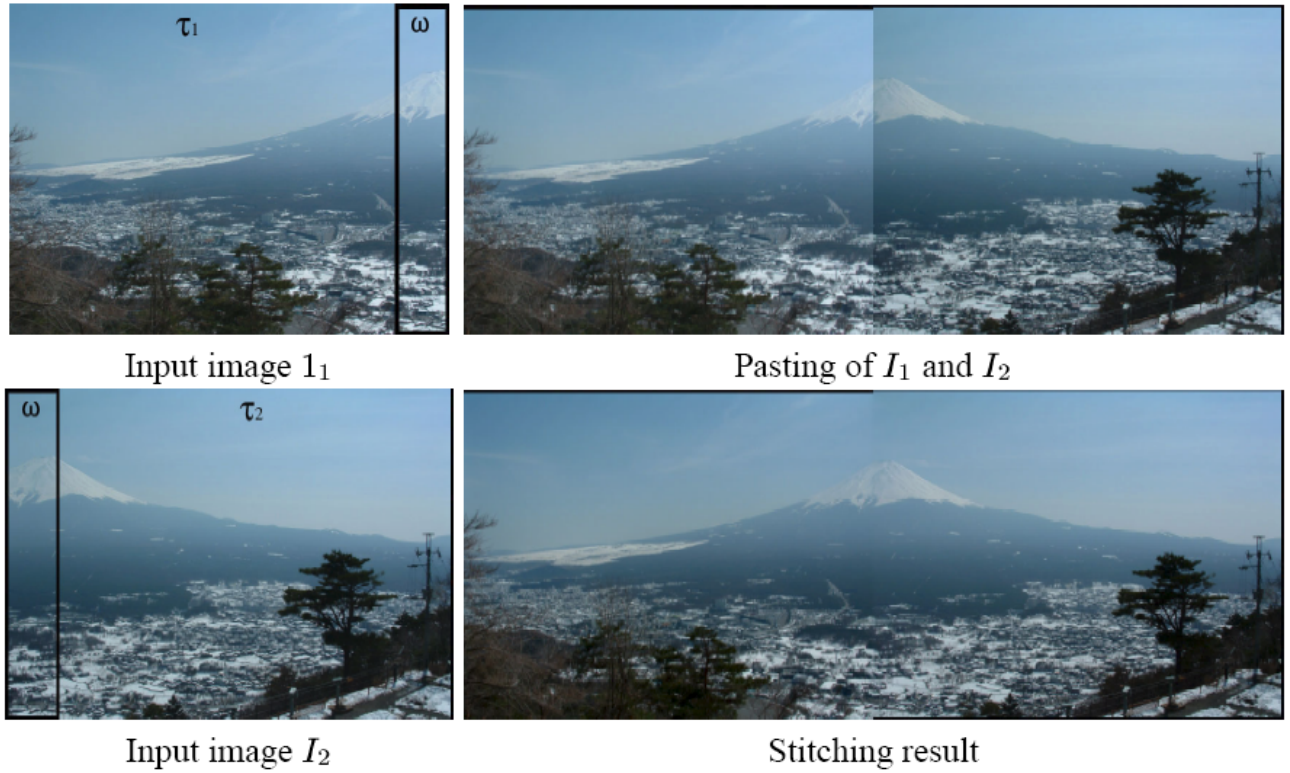


Fig. 1. Image stitching. On the left are the input images. ω is the overlap region. On top right is a simple pasting of the input images. On the bottom right is the result of the GIST1 algorithm.

Poisson Matting

- Sun et al. Siggraph 2004
- Assume gradient of F & B is negligible
- Plus various image-editing tools to refine matte

$$I = \alpha F + (1 - \alpha)B$$

$$\nabla I = (F - B)\nabla\alpha + \alpha\nabla F + (1 - \alpha)\nabla B$$

$$\nabla\alpha \approx \frac{1}{F - B}\nabla I$$

Figure 1: Pulling of matte from a complex scene. From left to right: a complex natural image for existing matting techniques where the color background is complex, a high quality matte generated by Poisson matting, a composite image with the extracted koala and a constant-color background, and a composite image with the extracted koala and a different background.

Poisson-ish mesh editing

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

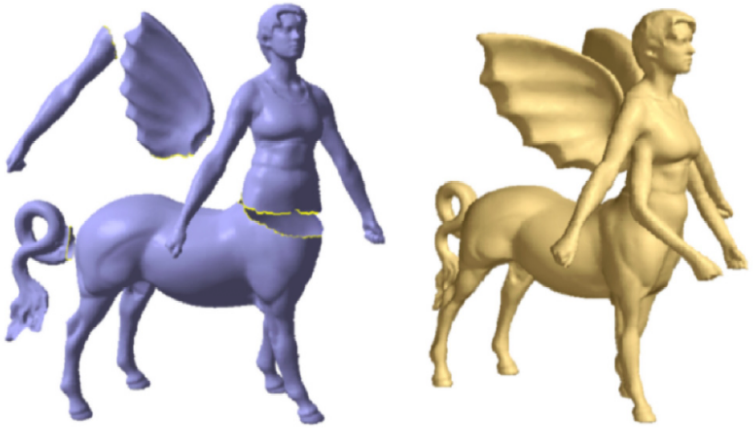


Figure 1: An unknown mythical creature. Left: mesh components for merging and deformation (the arm), Right: final editing result.

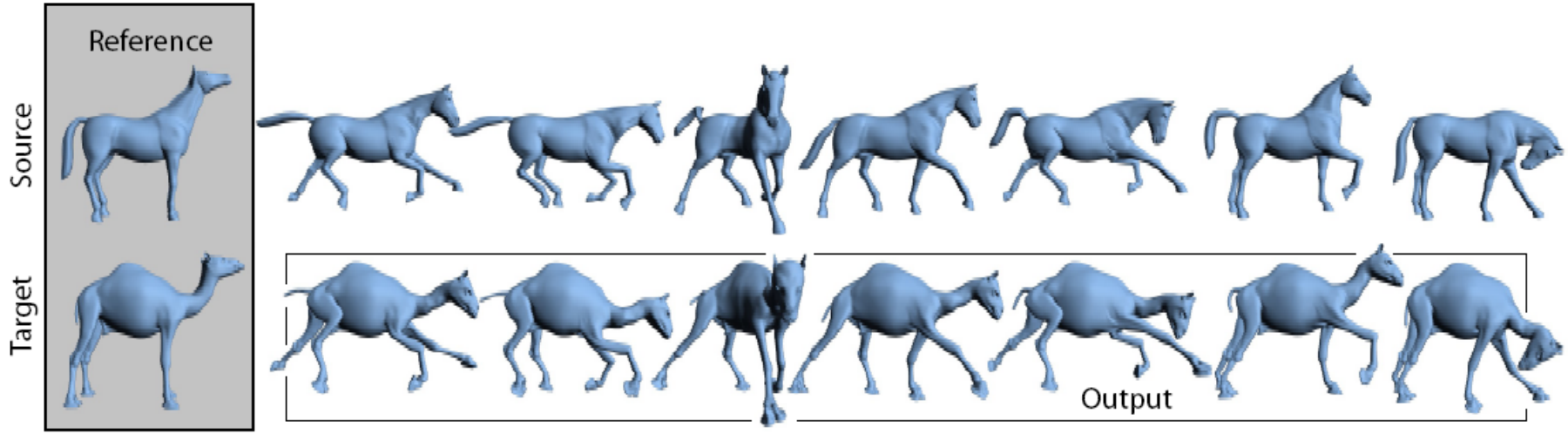


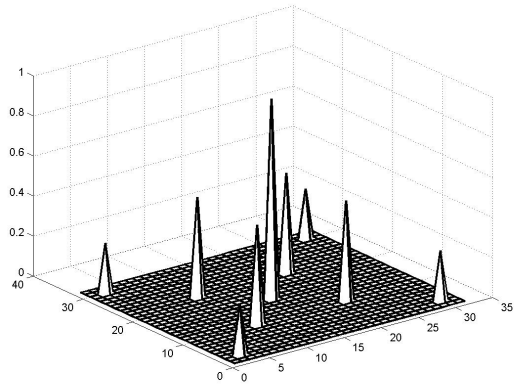
Figure 1: Deformation transfer copies the deformations exhibited by a source mesh onto a different target mesh. In this example, deformations of the reference horse mesh are transfered to the reference camel, generating seven new camel poses. Both gross skeletal changes as well as more subtle skin deformations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

Alternative to membrane

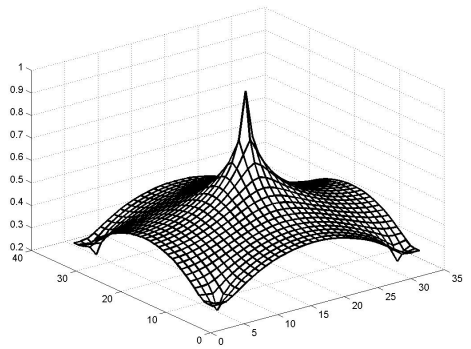
- Thin plate:
  minimize *second* derivative

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

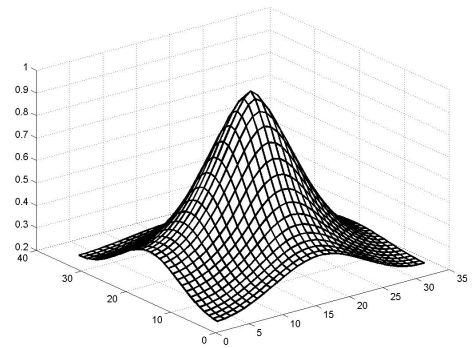
Data



Membrane interpolation

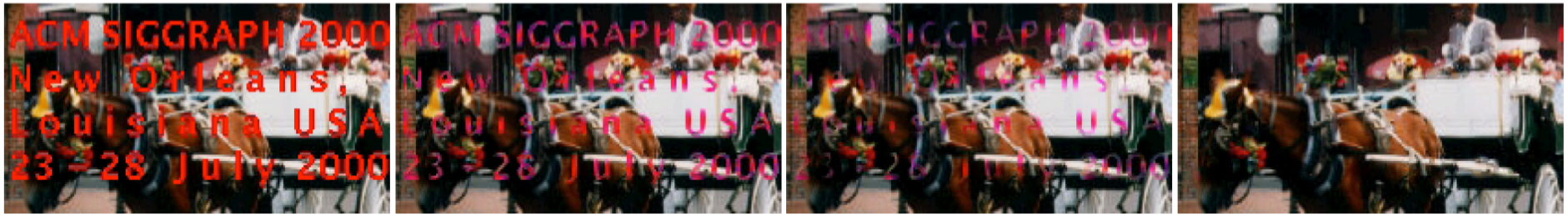


Thin-plate interpolation



Inpainting

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



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