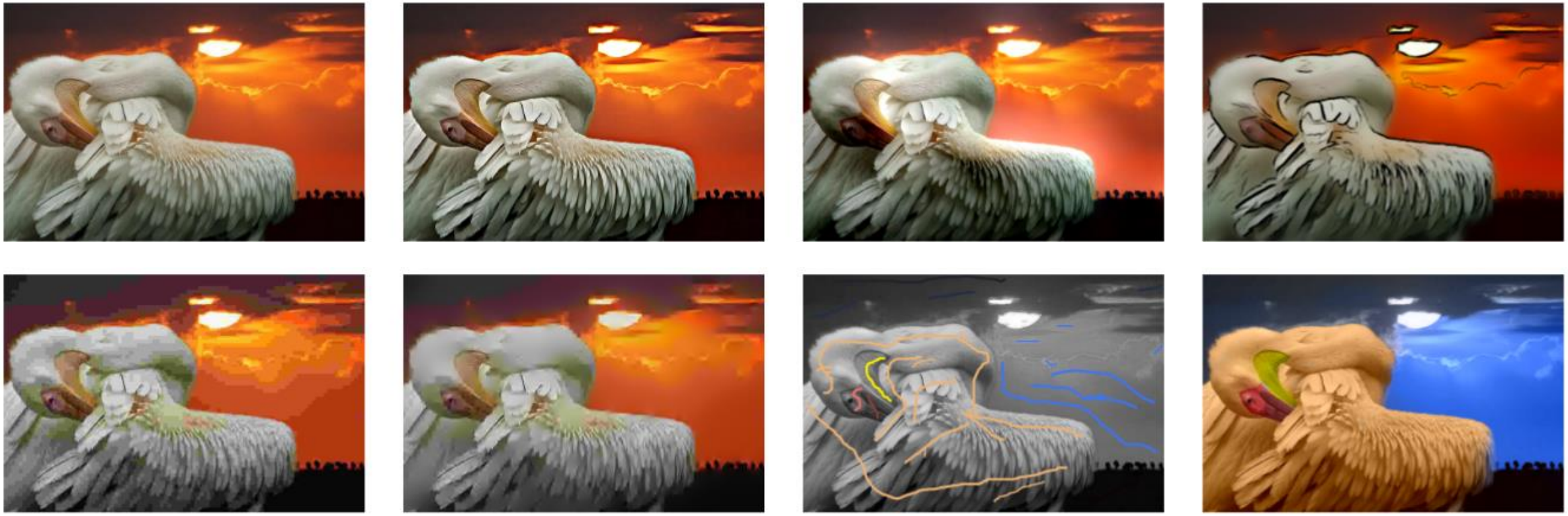
Gradient-domain image processing



15-463, 15-663, 15-862
Computational Photography
Fall 2018, Lecture 10

http://graphics.cs.cmu.edu/courses/15-463

Course announcements

- Homework 3 is out.
  - (Much) smaller than homework 2, but you should still start early to take advantage of bonus questions.
  - Requires a camera with flash for the second part.
- Grades for homework 1 have been posted.
- Make-up lecture will be scheduled soon.
  - What day does majority of the class prefer?
- How was Ravi's lecture on Monday?
- Thoughts on homework 2?

Overview of today's lecture

- Gradient-domain image processing.
- Basics on images and gradients.
- Integrable vector fields.
- Poisson blending.
- A more efficient Poisson solver.
- Poisson image editing examples.
- Flash/no-flash photography.

Slide credits

Many of these slides were adapted from:

- Kris Kitani (15-463, Fall 2016).
- Fredo Durand (MIT).
- James Hays (Georgia Tech).
- Amit Agrawal (MERL).

Gradient-domain image processing

Someone leaked season 8 of Game of Thrones

or, more likely, they made some creative use of Poisson blending

Application: Poisson blending

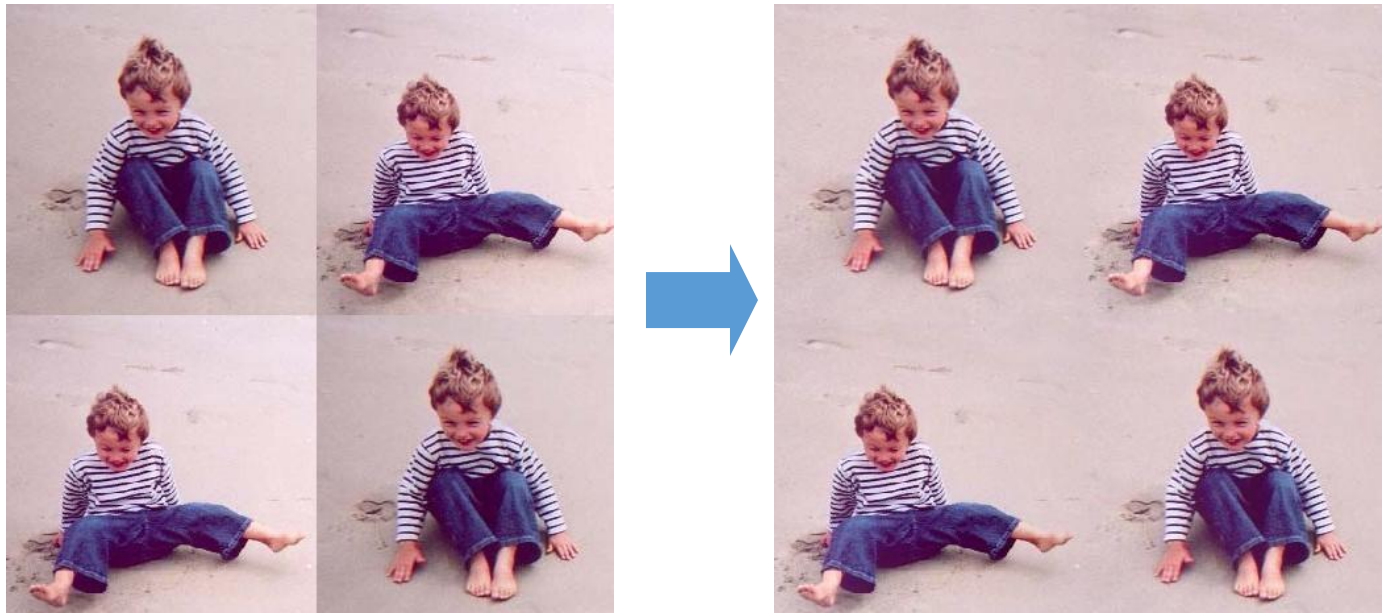
originals

copy-paste

Poisson blending

More applications

Removing Glass Reflections



Seamless Image Stitching

Yet more applications

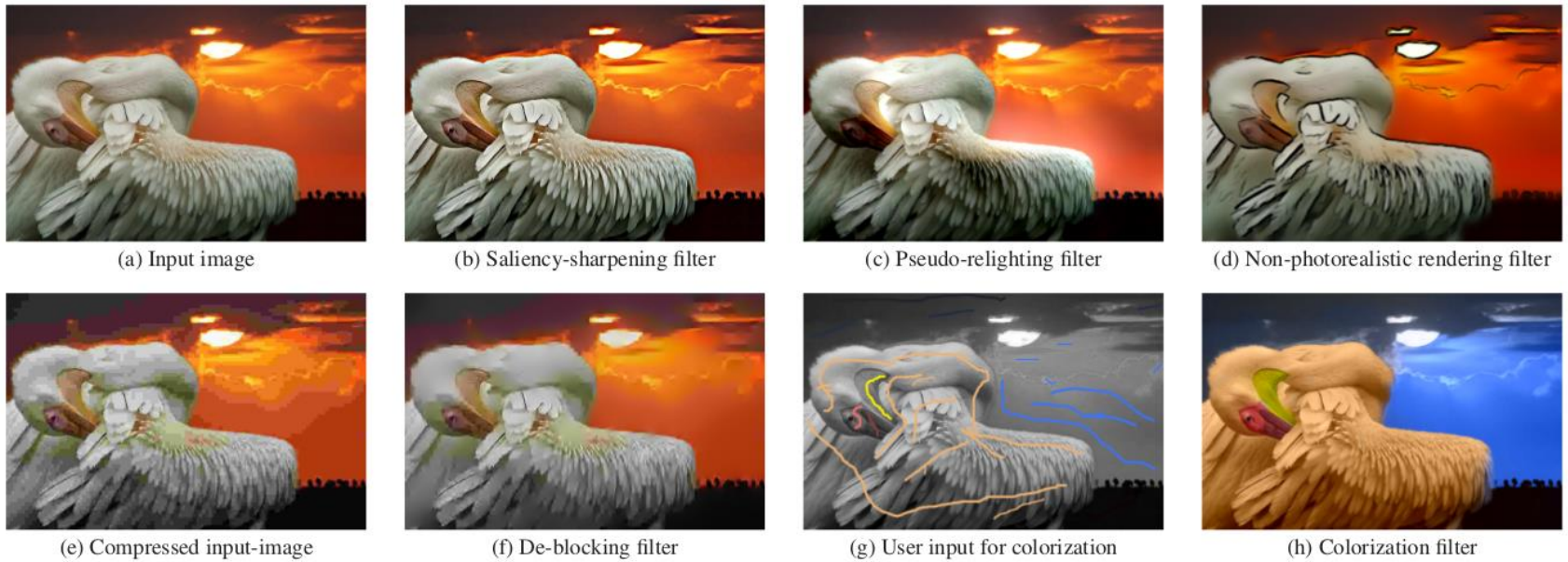
Fusing day and night photos

Tonemapping

Entire suite of image editing tools

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

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

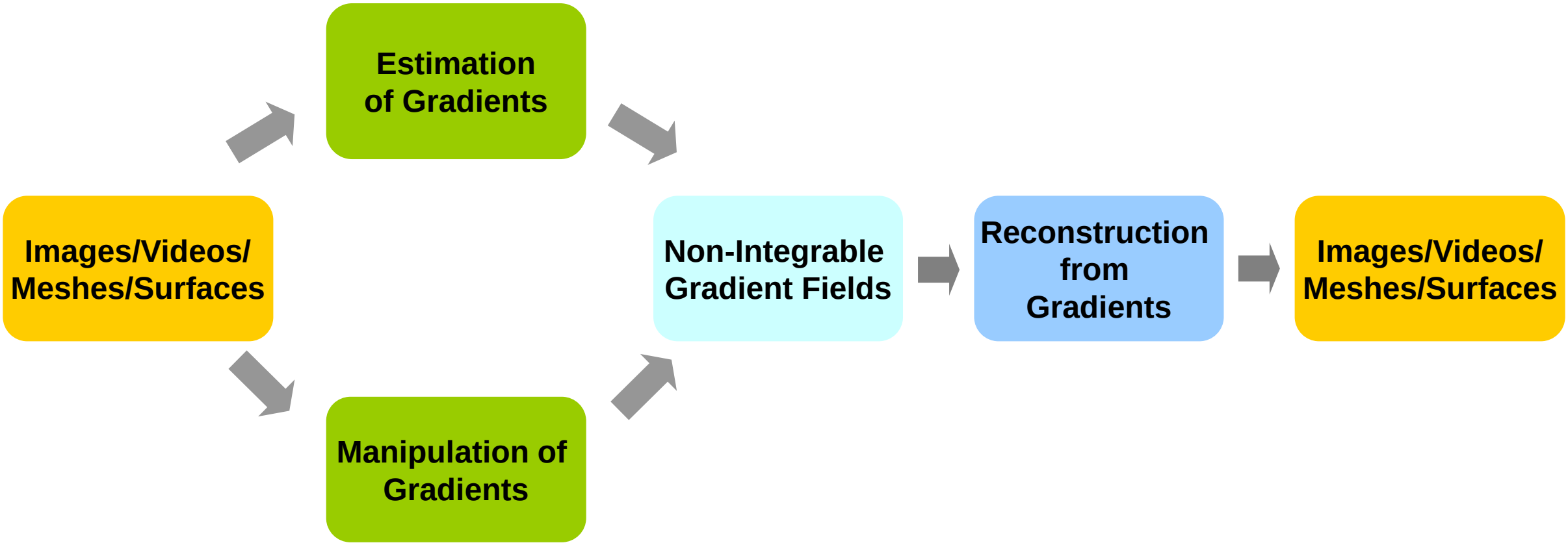


(a) Input image (b) Saliency-sharpening filter (c) Pseudo-relighting filter (d) Non-photorealistic rendering filter

(e) Compressed input-image (f) De-blocking filter (g) User input for colorization (h) Colorization filter

Figure 1: The figure shows some of the image-enhancement filters we have created using the GradientShop optimization-framework. GradientShop has been designed to allow applications to explore gradient-domain solutions for various image processing problems.

Main pipeline



Basics of images and gradients

Image representation

We can treat images as scalar fields (i.e.. two dimensional functions)

$I(x,y): \mathbb{R}^2 \rightarrow \mathbb{R}$

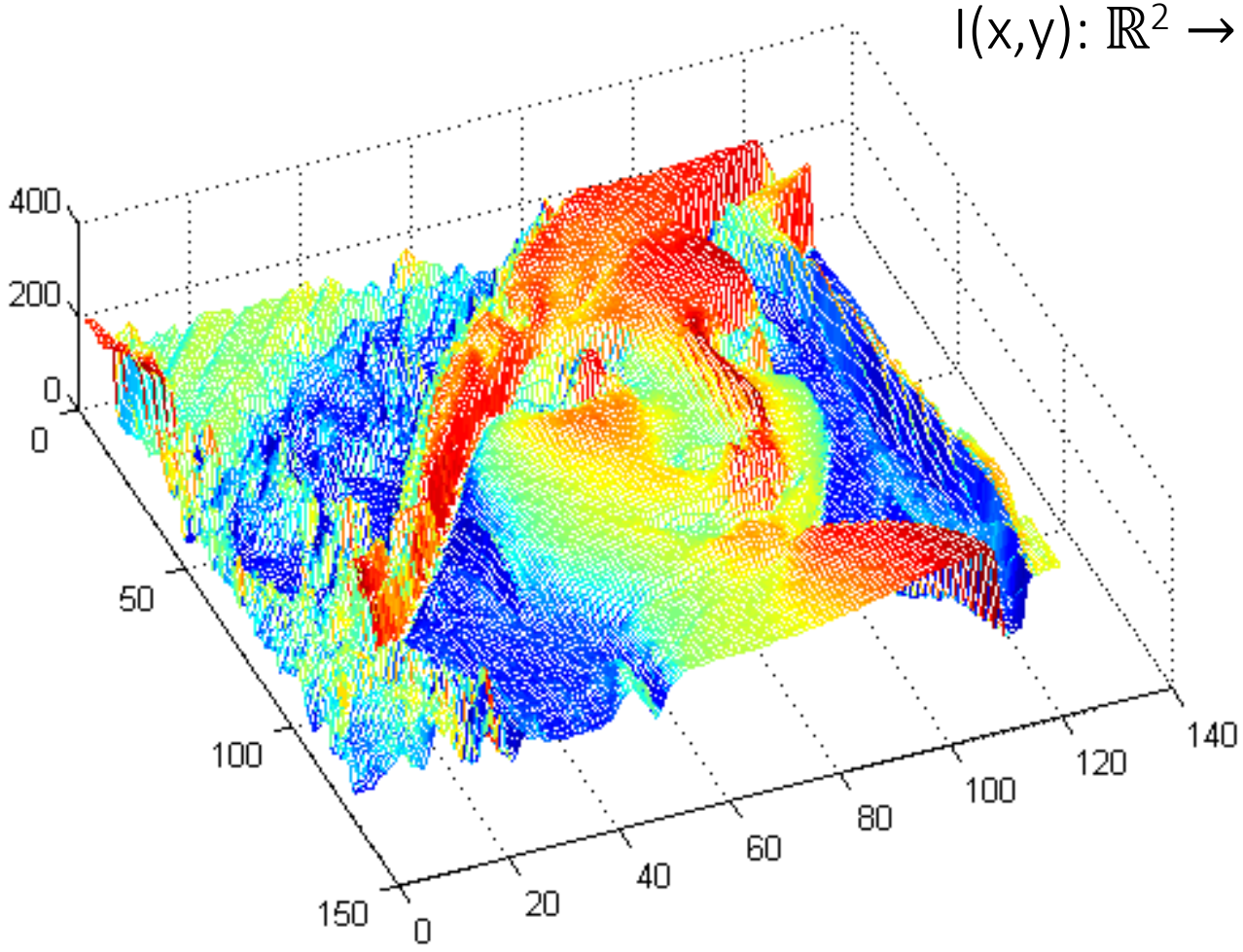


Image gradients

Convert the *scalar* field into a *vector* field through differentiation.

scalar field $I(x, y) : \mathbb{R}^2 \rightarrow \mathbb{R}$ ⟹ vector field $\nabla I = \{\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$

Image gradients

Convert the *scalar* field into a *vector* field through differentiation.

scalar field $I(x, y) : \mathbb{R}^2 \rightarrow \mathbb{R}$ ⇒ vector field $\nabla I = \{\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$

- How do we do this differentiation in real *discrete* images?

Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

How do you efficiently compute this?

Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

What convolution kernel does this correspond to?

Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

-1	0	1	?

1	0	-1	?

Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

1D derivative filter

1	0	-1

Image gradients

Convert the *scalar* field into a *vector* field through differentiation.

scalar field $I(x, y) : \mathbb{R}^2 \rightarrow \mathbb{R}$ ⇒ vector field $\nabla I = \{\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$

- How do we do this differentiation in real *discrete* images?
- Can we go in the opposite direction, from gradients to images?

Vector field integration

Two core questions:

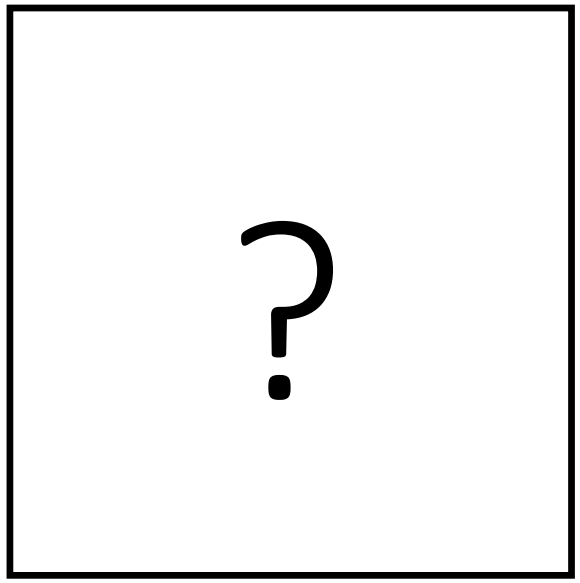
- When is integration of a vector field possible?

- How can integration of a vector field be performed?

Integrable vector fields

Integrable fields

Given an arbitrary vector field $(\mathbf{u}, \mathbf{v})$, can we always integrate it into a scalar field $\mathbf{I}$?



$I(x, y)$: $\mathbb{R}^2 \to \mathbb{R}$

$u(x, y)$: $\mathbb{R}^2 \to \mathbb{R}$

$v(x, y)$: $\mathbb{R}^2 \to \mathbb{R}$

such that

$$\frac{\partial I}{\partial x}(x, y) = u(x, y)$$

$$\frac{\partial I}{\partial y}(x, y) = v(x, y)$$

Curl and divergence

Curl: vector operator showing the rate of rotation of a vector field.

$$Curl\ (\nabla I) = \nabla \times \nabla I$$

Divergence: vector operator showing the isotropy of a vector field.

$$Div\ (\nabla I) = \nabla \bullet \nabla I$$

Do you know of some simpler versions of these operators?

Curl and divergence

Curl: vector operator showing the rate of rotation of a vector field.

$$Curl\ (\nabla I) = \det \begin{vmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \\ I_x & I_y \end{vmatrix} = \frac{\partial I_y}{\partial x} - \frac{\partial I_x}{\partial y} = I_{yx} - I_{xy}$$

Divergence: vector operator showing the isotropy of a vector field.

$$div\,(I_x, I_y) = \frac{\partial I_x}{\partial x} + \frac{\partial I_y}{\partial y} = I_{xx} + I_{yy}$$

Can you use either of these operators to derive an integrability condition?

Integrability condition

Curl of the gradient field should be zero:

$$Curl\ (\nabla I) = I_{yx} - I_{xy} = 0$$

What does that mean intuitively?

Integrability condition

Curl of the gradient field should be zero:

$$Curl\ (\nabla I) = I_{yx} - I_{xy} = 0$$

What does that mean intuitively?

- Same result independent of order of differentiation.

$$I_{yx} = I_{xy}$$

Demonstration

Image

I_x

I_y

How do we compute this? → $Div(I_x, I_y)$

$Curl(I_x, I_y)$

I_{xy}

=

I_{yx}

Laplace filter

Basically a second derivative filter.

- We can use finite differences to derive it, as with first derivative filter.

first-order finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

1D derivative filter

1	0	-1

second-order finite difference

$$f''(x) = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}$$

Laplace filter

?

Laplace filter

Basically a second derivative filter.

- We can use finite differences to derive it, as with first derivative filter.

first-order finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

1D derivative filter

1	0	-1

second-order finite difference

$$f''(x) = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}$$

Laplace filter

1	-2	1

Vector field integration

Two core questions:

- When is integration of a vector field possible?
  - Use curl to check for equality of mixed partial second derivatives.

- How can integration of a vector field be performed?

Different types of integration problems

- Reconstructing height field from gradients
  Applications: shape from shading, photometric stereo

- Manipulating image gradients
  Applications: tonemapping, image editing, matting, fusion, mosaics

- Manipulation of 3D gradients
  Applications: mesh editing, video operations

Key challenge: Most vector fields in applications are not integrable.

- Integration must be done *approximately*.

Poisson blending

Application: Poisson blending

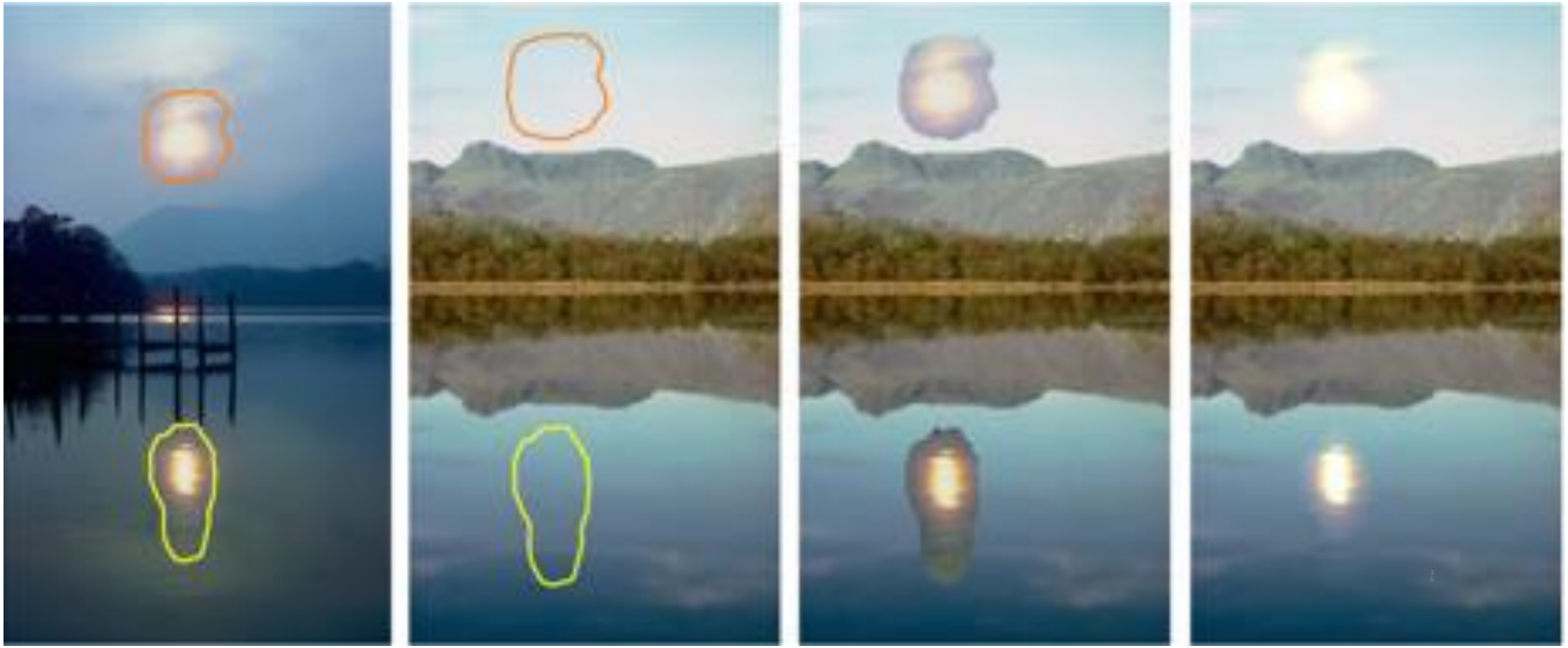
originals

copy-paste

Poisson blending

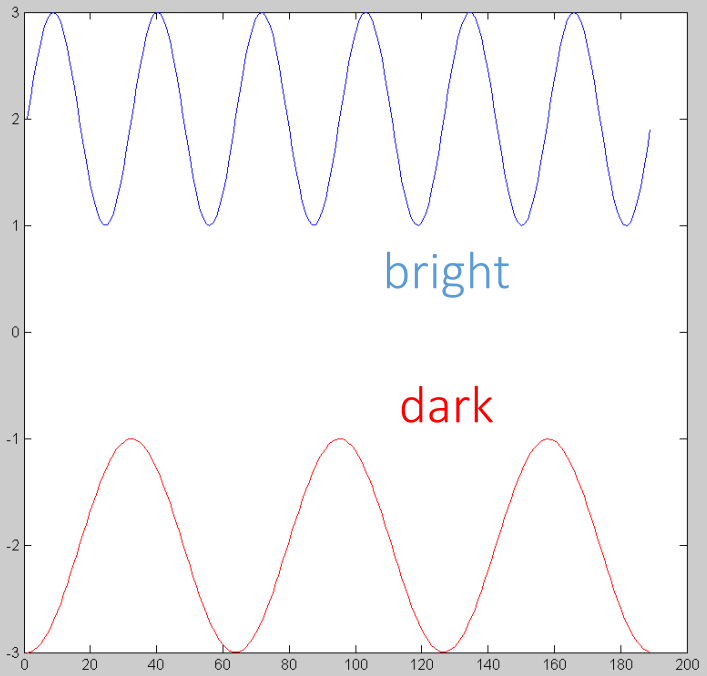
Key idea

When blending, retain the gradient information as best as possible

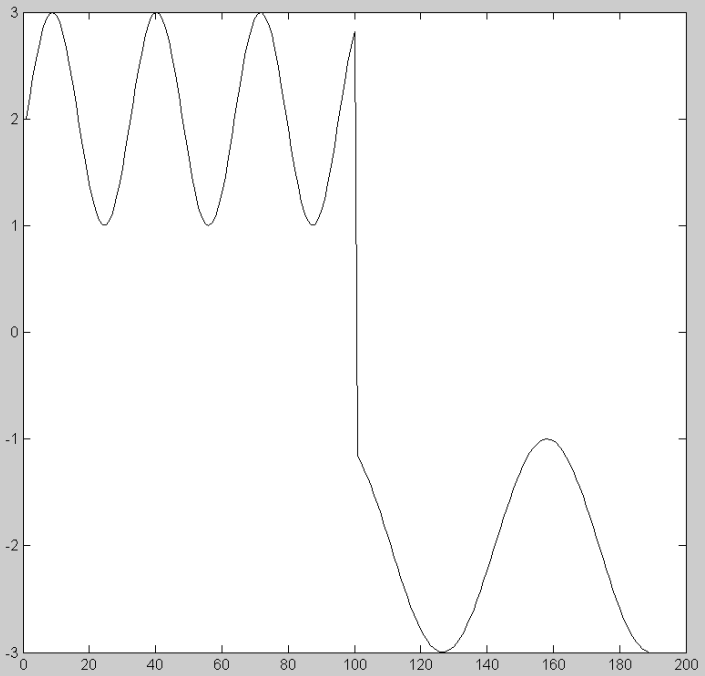


source | destination | copy-paste | Poisson blending

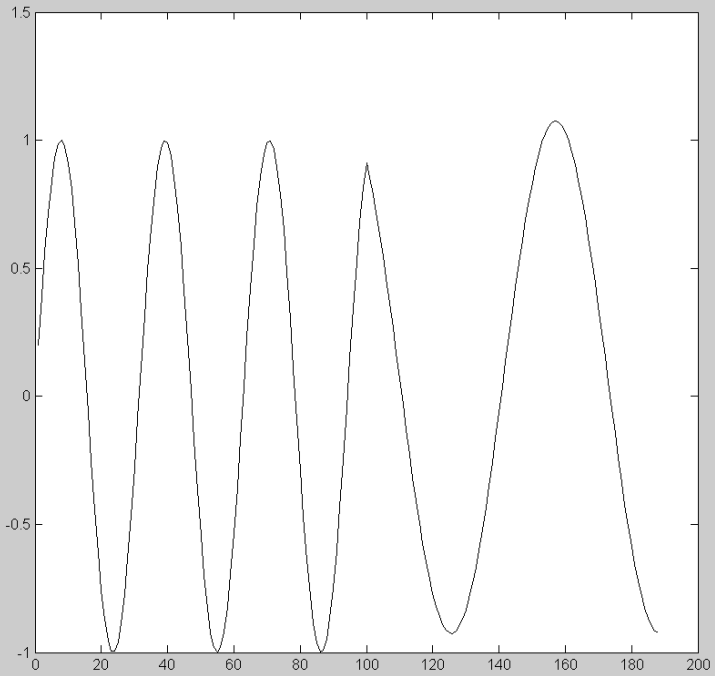
Poisson blending: 1D example



two signals



regular blending



blending derivatives

Definitions and notation

Notation

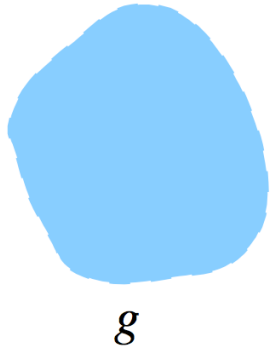
g: source function

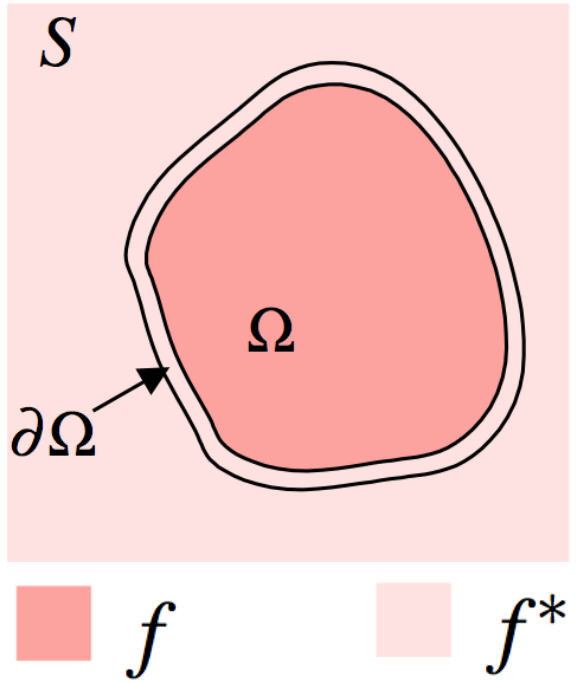
S: destination

Ω: destination domain

f: interpolant function

f*: destination function





Which one is the unknown?

Definitions and notation

Notation

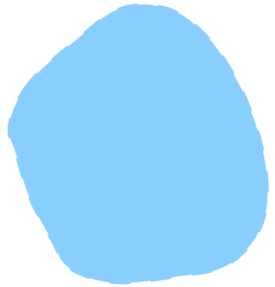
g: source function

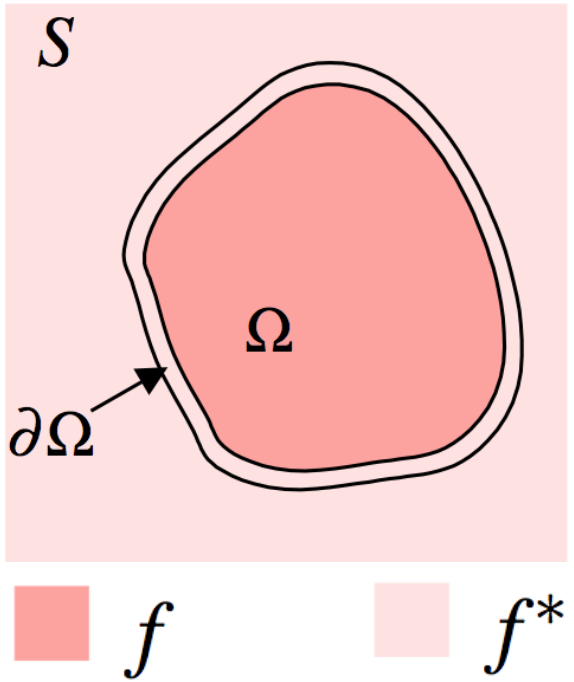
S: destination

Ω: destination domain

f: interpolant function

f*: destination function





How should we determine f?

- should it look like g?
- should it look like f*?

Interpolation criterion

"Variational" means optimization where the unknown is an entire function

Variational problem

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

what does this term do?

what does this term do?

Recall ...

Image gradient

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

is this known?

$$\mathbf{v} = (u, v) = \nabla g$$

Interpolation criterion

"Variational" means optimization where the unknown is an entire function

Variational problem

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

gradient of f looks like gradient of g

f is equivalent to f* at the boundaries

Recall ...

Image gradient

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

Yes, since the source function g is known

$$\mathbf{v} = (u, v) = \nabla g$$

Equivalently

This is where *Poisson* blending comes from

Poisson equation (with Dirichlet boundary conditions)

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

what does this term do?

Gradient $\mathbf{v} = (u, v) = \nabla g$

Laplacian $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Divergence $\text{div } \mathbf{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$

$$\begin{aligned} \text{div } \mathbf{v} &= \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \\ &= \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \\ &= \Delta g \end{aligned}$$

Equivalently

Poisson equation (with Dirichlet boundary conditions)

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

Laplacian of f same as g

Gradient $\mathbf{v} = (u, v) = \nabla g$

Laplacian $\Delta f = \dfrac{\partial^2 f}{\partial x^2} + \dfrac{\partial^2 f}{\partial y^2}$

Divergence $\operatorname{div} \mathbf{v} = \dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y}$

$$\begin{aligned} \operatorname{div} \mathbf{v} &= \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \\ &= \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \\ &= \Delta g \end{aligned}$$

Equivalently

Poisson equation (with Dirichlet boundary conditions)

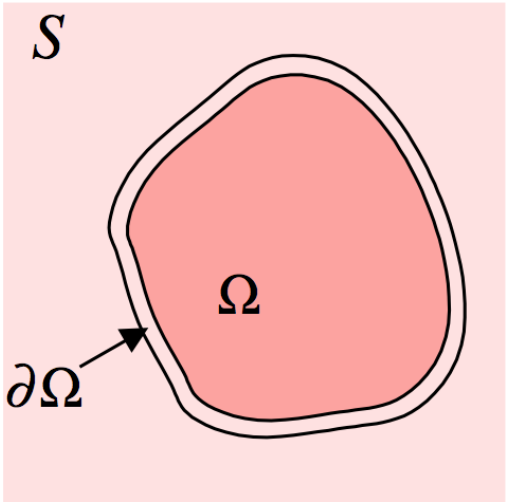
$$\Delta f = \operatorname{div} \mathbf{v} \quad \text{over} \quad \Omega, \quad \text{with} \quad f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

g

so make these guys ...

Δg Δf

the same



How can we do this?

Equivalently

Poisson equation (with Dirichlet boundary conditions)

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

So for each pixel p, do: $\Delta f_p = \Delta g_p$

Or for discrete images: $4f_p - \sum_{q \in N_p} f_q = 4g_p - \sum_{q \in N_p} g_q$

How did we compute the Laplacian?

Equivalently

Poisson equation (with Dirichlet boundary conditions)

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

So for each pixel p, do:

$$\Delta f_p = \Delta g_p$$

Or for discrete images:

$$4f_p - \sum_{q \in N_p} f_q = 4g_p - \sum_{q \in N_p} g_q$$

Recall...

Laplace filter

0	1	0
1	-4	1
0	1	0

What's known and what's unknown?

Equivalently

Poisson equation (with Dirichlet boundary conditions)

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

So for each pixel p, do: $\Delta f_p = \Delta g_p$

Or for discrete images: $4f_p - \sum_{q \in N_p} f_q = 4g_p - \sum_{q \in N_p} g_q$

Recall...

Laplace filter

0	1	0
1	-4	1
0	1	0

f is unknown except at the boundary

g and its Laplacian are known

We can rewrite this as

linear equation of N variables

$$4f_p - \sum_{q \in N_p} f_q = 4g_p - \sum_{q \in N_p} g_q$$

one for each pixel in destination

In vector form:

$$[0 \cdots -1 \cdots -1 \quad 4 \quad -1 \cdots -1 \cdots 0] \begin{bmatrix} f_1 \\ \vdots \\ f_{q_1} \\ \vdots \\ f_{q_2} \\ f_p \\ f_{q_3} \\ \vdots \\ f_{q_4} \\ \vdots \\ f_N \end{bmatrix} = \begin{bmatrix} \Delta g_1 \\ \vdots \\ \Delta g_{q_1} \\ \vdots \\ \Delta g_{q_2} \\ \Delta g_p \\ \Delta g_{q_3} \\ \vdots \\ \Delta g_{q_4} \\ \vdots \\ \Delta g_N \end{bmatrix}$$

What is this? ↗

(each pixel adds another ‘sparse’ row here)

Linear system of equations

$$Af = b$$

How would you solve this?

WARNING: requires special treatment at the borders
(target boundary values are same as source)

Solving the linear system

Convert the system to a linear least-squares problem:

$$E_{\mathrm{LLS}} = \|\mathbf{A}f - \boldsymbol{b}\|^2$$

Expand the error:

$$E_{\mathrm{LLS}} = f^\top(\mathbf{A}^\top\mathbf{A})f - 2f^\top(\mathbf{A}^\top\boldsymbol{b}) + \|\boldsymbol{b}\|^2$$

Minimize the error:

Set derivative to 0 $(\mathbf{A}^\top\mathbf{A})f = \mathbf{A}^\top\boldsymbol{b}$

Solve for x $f = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\boldsymbol{b}$

Solving the linear system

Convert the system to a linear least-squares problem:

$$E_{\mathrm{LLS}} = \|\mathbf{A}f - \boldsymbol{b}\|^2$$

Expand the error:

$$E_{\mathrm{LLS}} = f^\top(\mathbf{A}^\top\mathbf{A})f - 2f^\top(\mathbf{A}^\top\boldsymbol{b}) + \|\boldsymbol{b}\|^2$$

Minimize the error:

Set derivative to 0 $(\mathbf{A}^\top\mathbf{A})f = \mathbf{A}^\top\boldsymbol{b}$

Solve for x $f = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\boldsymbol{b}$ ←

Note: You almost <u>never</u> want to compute the inverse of a matrix.

In Matlab:

```
f = A \ b
```

Integration procedures

- Poisson solver (i.e., least squares integration)
  + Generally applicable.
  - Matrices A can become very large.

- Acceleration techniques:
  + (Conjugate) gradient descent solvers.
  + Multi-grid approaches.
  + Pre-conditioning.
  + Quadtree decompositions.

- Alternative solvers: projection procedures.
  We will discuss one of these when we cover photometric stereo.

A more efficient Poisson solver

Let’s look again at our optimization problem

Variational problem

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

gradient of f looks like gradient of g

f is equivalent to f* at the boundaries

Recall ...

Image gradient

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

Let's look again at our optimization problem

Variational problem

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

gradient of f looks like gradient of g

f is equivalent to f* at the boundaries

Recall ...

Image gradient

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

And for discrete images:

$$\frac{\partial}{\partial x} \approx \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \end{array}$$

$$\frac{\partial}{\partial y} \approx \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline -1 \\ \hline \end{array}$$

Let's look again at our optimization problem

We can use the gradient approximation to discretize the variational problem

Discrete problem

What are G, f, and v?

$$\min_f \|Gf - v\|^2$$

We will ignore the boundary conditions for now.

Recall ...

Image gradient

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

And for discrete images:

$$\frac{\partial}{\partial x} \approx \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \end{array}$$

$$\frac{\partial}{\partial y} \approx \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline -1 \\ \hline \end{array}$$

Let's look again at our optimization problem

We can use the gradient approximation to discretize the variational problem

Discrete problem

matrix G formed by stacking together discrete gradients

$$\min_f \|Gf - v\|^2$$

vectorized version of the unknown image

vectorized version of the target gradient field

We will ignore the boundary conditions for now.

Recall ...

Image gradient

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

And for discrete images:

$$\frac{\partial}{\partial x} \approx \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \end{array}$$

$$\frac{\partial}{\partial y} \approx \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline -1 \\ \hline \end{array}$$

Let's look again at our optimization problem

We can use the gradient approximation to discretize the variational problem

Discrete problem

matrix G formed by stacking together discrete gradients

$$\min_f \|Gf - v\|^2$$

vectorized version of the unknown image

vectorized version of the target gradient field

We will ignore the boundary conditions for now.

Recall ...

Image gradient

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

And for discrete images:

$$\frac{\partial}{\partial x} \approx \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \end{array}$$

$$\frac{\partial}{\partial y} \approx \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline -1 \\ \hline \end{array}$$

Let's look again at our optimization problem

Discrete problem

matrix G formed by stacking together discrete gradients

$$\min_f \|Gf - v\|^2$$

vectorized version of the unknown image

vectorized version of the target gradient field

How do we solve this optimization problem?

Recall ...

Image gradient

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

And for discrete images:

$$\frac{\partial}{\partial x} \approx \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \end{array}$$

$$\frac{\partial}{\partial y} \approx \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline -1 \\ \hline \end{array}$$

Approach 1: Compute stationary points

Given the loss function:

$$E(f) = \|Gf - v\|^2$$

… we compute its derivative:

$$\frac{\partial E}{\partial f} = ?$$

Approach 1: Compute stationary points

Given the loss function:

$$E(f) = \|Gf - v\|^2$$

… we compute its derivative:

$$\frac{\partial E}{\partial f} = G^T G f - v$$

… and we do what with it?

Approach 1: Compute stationary points

Given the loss function:

$$E(f) = \|Gf - v\|^2$$

… we compute its derivative:

$$\frac{\partial E}{\partial f} = G^T G f - v$$

… and we set that to zero:

$$\frac{\partial E}{\partial f} = 0 \Rightarrow \underbrace{G^T G f} = v$$

What is this matrix?

Approach 1: Compute stationary points

Given the loss function:

$$E(f) = \|Gf - v\|^2$$

… we compute its derivative:

$$\frac{\partial E}{\partial f} = G^T G f - v$$

… and we set that to zero:

$$\frac{\partial E}{\partial f} = 0 \Rightarrow \underbrace{G^T G f} = v$$

It is equal to the Laplacian matrix A we derived previously!

Reminder from variational case

Poisson equation (with Dirichlet boundary conditions)

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

So for each pixel p, do: $\Delta f_p = \Delta g_p$

Or for discrete images: $4f_p - \sum_{q \in N_p} f_q = 4g_p - \sum_{q \in N_p} g_q$

Recall...

Laplace filter

0	1	0
1	-4	1
0	1	0

What's known and what's unknown?

Reminder from variational case

linear equation of N variables

$$4f_p - \sum_{q \in N_p} f_q = 4g_p - \sum_{q \in N_p} g_q$$

one for each pixel in destination

In vector form:

$$[0 \cdots -1 \cdots -1 \quad 4 \quad -1 \cdots -1 \cdots 0] \begin{bmatrix} f_1 \\ \vdots \\ f_{q_1} \\ \vdots \\ f_{q_2} \\ f_p \\ f_{q_3} \\ \vdots \\ f_{q_4} \\ \vdots \\ f_N \end{bmatrix} = \begin{bmatrix} \Delta g_1 \\ \vdots \\ \Delta g_{q_1} \\ \vdots \\ \Delta g_{q_2} \\ \Delta g_p \\ \Delta g_{q_3} \\ \vdots \\ \Delta g_{q_4} \\ \vdots \\ \Delta g_N \end{bmatrix}$$

What is this? ↗

(each pixel adds another 'sparse' row here)

Linear system of equations

$$Af = b$$

Same system as:

$$G^T G f = v$$

We arrive at the same system, no matter whether we discretize the continuous Laplace equation or the variational optimization problem.

Approach 1: Compute stationary points

Given the loss function:

$$E(f) = \|Gf - v\|^2$$

… we compute its derivative:

$$\frac{\partial E}{\partial f} = G^T G f - v$$

… and we set that to zero:

$$\frac{\partial E}{\partial f} = 0 \Rightarrow G^T G f = v$$

Solving this is <u>exactly</u> as expensive as what we had before.

Approach 2: Use gradient descent

Given the loss function:

$$E(f) = \|Gf - v\|^2$$

… we compute its derivative:

$$\frac{\partial E}{\partial f} = G^T Gf - v = Af - v \equiv r$$

We call this term the *residual*

Approach 2: Use gradient descent

Given the loss function:

$$E(f) = \|Gf - v\|^2$$

… we compute its derivative:

$$\frac{\partial E}{\partial f} = G^T Gf - v = Af - v \equiv r$$

We call this term the *residual*

… and then we *iteratively* compute a solution:

$$f^{i+1} = f^i - \eta^i r^i$$

for $\mathrm{i} = 0, 1, \ldots, \mathrm{N}$, where

η^i are positive *step sizes*

Selecting optimal step sizes

Make derivative of loss function *with respect to* η^i equal to zero:

$$E(f) = \|Gf - v\|^2$$

$$E\left(f^{i+1}\right) = \left\|G\left(f^i - \eta^i r^i\right) - v\right\|^2$$

$$\frac{\partial E\left(f^{i+1}\right)}{\partial r^i} = \left[v - A\left(f^i - \eta^i r^i\right)\right]^T r^i = 0 \Rightarrow \eta^i = \frac{\left(r^i\right)^T r^i}{(r^i)^T A r^i}$$

Gradient descent

Given the loss function:

$$E(f) = \|Gf - v\|^2$$

Minimize by iteratively computing:

$$f^{i+1} = f^i - \eta^i r^i, \quad r^i = v - Af^i, \quad \eta^i = \frac{\left(r^i\right)^T r^i}{\left(r^i\right)^T A r^i} \qquad \text{for i} = 0, 1, \ldots, \text{N}$$

Is this cheaper than the pseudo-inverse approach?

Gradient descent

Given the loss function:

$$E(f) = \|Gf - v\|^2$$

Minimize by iteratively computing:

$$f^{i+1} = f^i - \eta^i r^i, \quad r^i = v - Af^i, \quad \eta^i = \frac{\left(r^i\right)^T r^i}{\left(r^i\right)^T A r^i} \qquad \text{for i} = 0, 1, \ldots, \text{N}$$

Is this cheaper than the pseudo-inverse approach?

- We never need to compute A, only its products with vectors f, r.

Gradient descent

Given the loss function:

$$E(f) = \|Gf - v\|^2$$

Minimize by iteratively computing:

$$f^{i+1} = f^i - \eta^i r^i, \quad r^i = v - Af^i, \quad \eta^i = \frac{(r^i)^T r^i}{(r^i)^T A r^i} \qquad \text{for i} = 0, 1, \ldots, \text{N}$$

Is this cheaper than the pseudo-inverse approach?

- We never need to compute $\mathbf{A}$, only its products with vectors $\mathbf{f}$, $\mathbf{r}$.
- Vectors $\mathbf{f}$, $\mathbf{r}$ are images.

Gradient descent

Given the loss function:

$$E(f) = \|Gf - v\|^2$$

Minimize by iteratively computing:

$$f^{i+1} = f^i - \eta^i r^i, \quad r^i = v - Af^i, \quad \eta^i = \frac{(r^i)^T r^i}{(r^i)^T A r^i} \qquad \text{for i} = 0, 1, \ldots, \text{N}$$

Is this cheaper than the pseudo-inverse approach?

- We never need to compute $\mathbf{A}$, only its products with vectors $\mathbf{f}$, $\mathbf{r}$.
- Vectors $\mathbf{f}$, $\mathbf{r}$ are images.
- Because $\mathbf{A}$ is the *Laplacian matrix*, these matrix-vector products can be efficiently computed using *convolutions* with the *Laplacian kernel*.

In practice: conjugate gradient descent

Given the loss function:

$$E(f) = \|Gf - v\|^2$$

Minimize by iteratively computing:

$$f^{i+1} = f^i + \eta^i d^i, \quad r^i = v - Af^i, \qquad \text{for i} = 0, 1, \ldots, \text{N}$$

$$d^{i+1} = r^{i+1} + \beta^{i+1} d^i,$$

$$\beta^{i+1} = \frac{\left(r^{i+1}\right)^T r^{i+1}}{(r^i)^T r^i} \qquad \eta^i = \frac{\left(d^i\right)^T r^i}{(d^i)^T A d^i}$$

- Smarter way for selecting update directions
- Everything can still be done using convolutions

Note: initialization

Does the initialization f^0 matter?

Note: initialization

Does the initialization $\mathbf{f}^0$ matter?

- It doesn't matter in terms of what final $\mathbf{f}$ we converge to, because the loss function is convex.

$$E(f) = \|Gf - v\|^2$$

Note: initialization

Does the initialization $\mathbf{f}^0$ matter?

- It doesn't matter in terms of what final $\mathbf{f}$ we converge to, because the loss function is convex.

$$E(f) = \|Gf - v\|^2$$

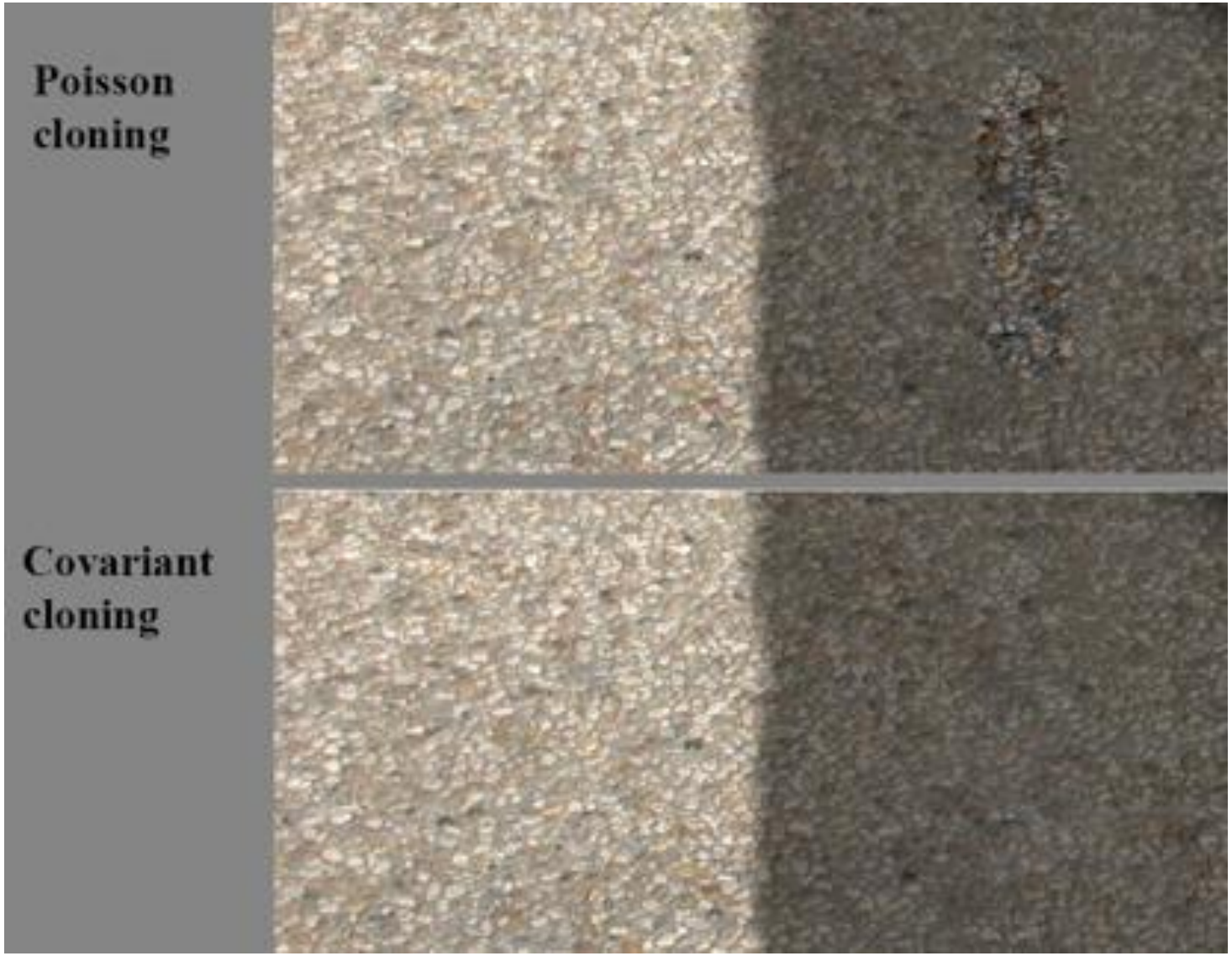
- It does matter in terms of convergence speed.
- We typically use a *multi-grid* approach:
  - Solve an initial problem for a very low-resolution f (e.g., 2x2).
  - Use the solution to initialize gradient descent for a higher resolution $\mathbf{f}$ (e.g., 4x4).
  - Use the solution to initialize gradient descent for a higher resolution $\mathbf{f}$ (e.g., 8x8).

  …

  - Use the solution to initialize gradient descent for an $\mathbf{f}$ with the original resolution NxN.

Poisson image editing examples

Photoshop's "healing brush"



Slightly more advanced version of what we covered here:

- Uses higher-order derivatives

Contrast problem

Loss of contrast when pasting from dark to bright:

- Contrast is a multiplicative property.
- With Poisson blending we are matching linear differences.

Contrast problem

Loss of contrast when pasting from dark to bright:

- Contrast is a multiplicative property.
- With Poisson blending we are matching linear differences.

Solution: Do blending in log-domain.

More blending

originals

copy-paste

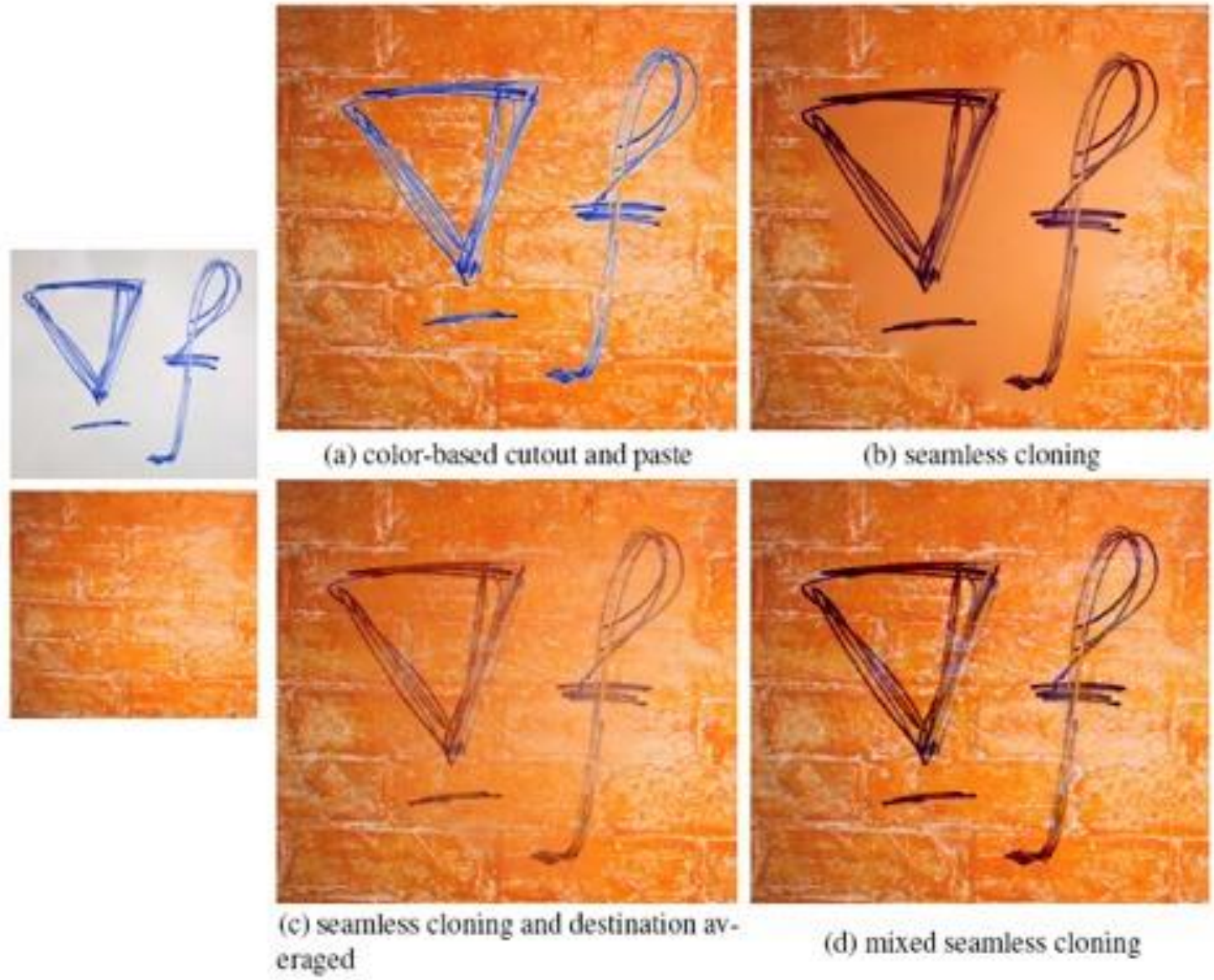
Poisson blending

Blending transparent objects

source

destination

Blending objects with holes



(a) color-based cutout and paste

(b) seamless cloning

(c) seamless cloning and destination averaged

(d) mixed seamless cloning

Editing

Concealment

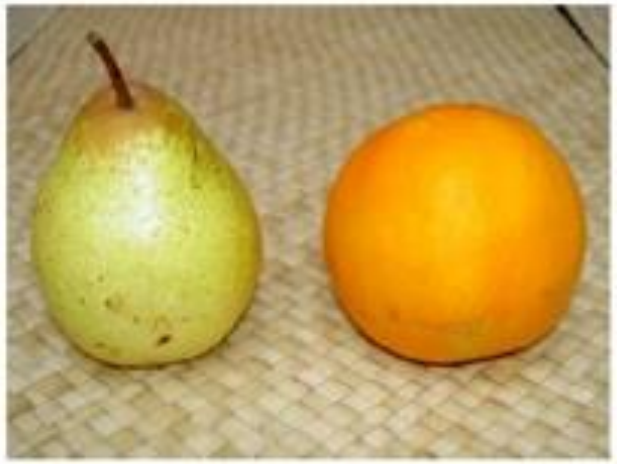
How would you do this with Poisson blending?

Concealment

How would you do this with Poisson blending?

- Insert a copy of the background.

Texture swapping



Special case: membrane interpolation

How would you do this?

Special case: membrane interpolation

How would you do this?

Poisson problem

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

Laplacian problem

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

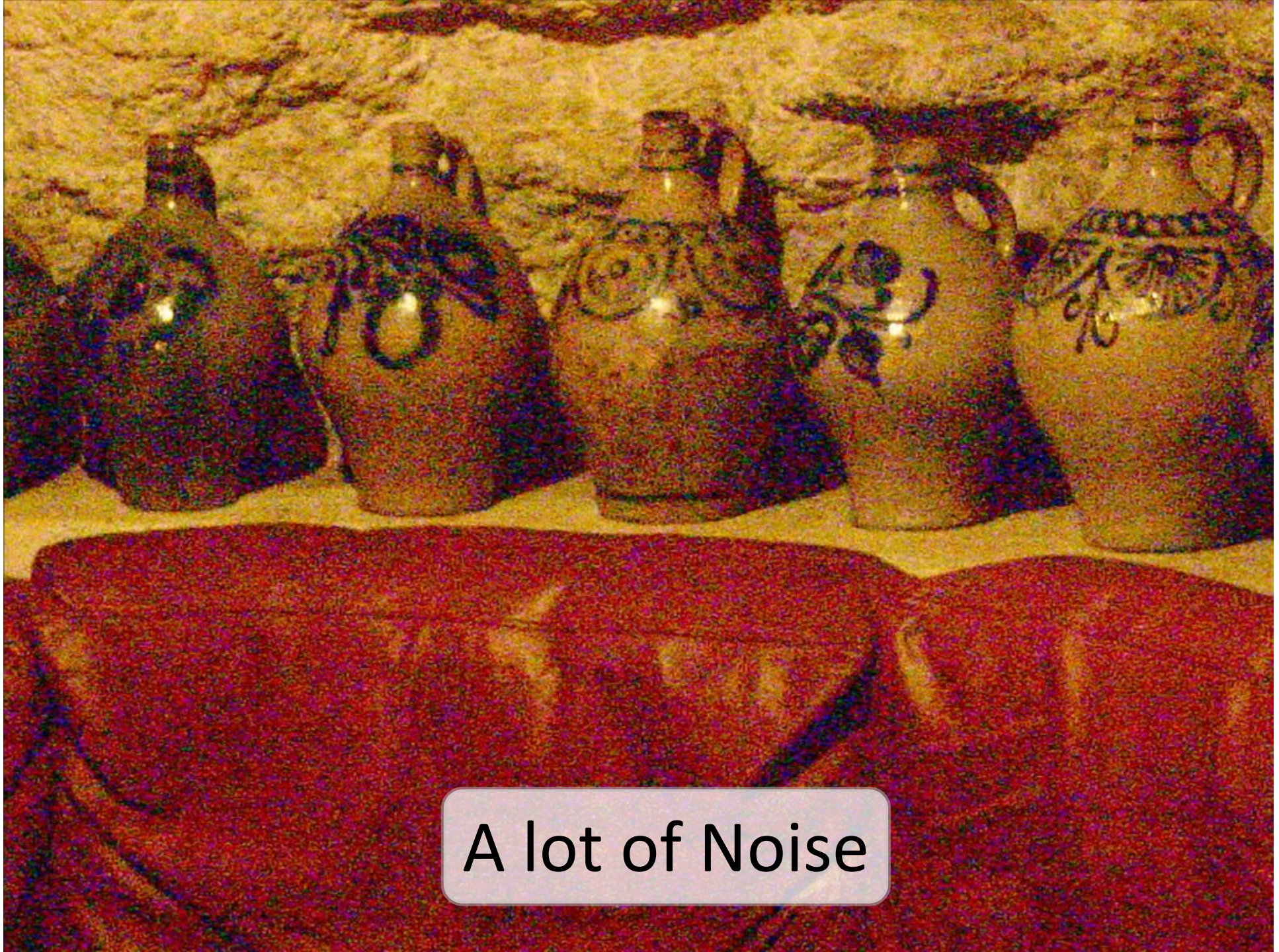
Flash/no-flash photography


Red Eye


Unflattering Lighting


Motion Blur


Noise


A lot of Noise


Ruined Ambiance

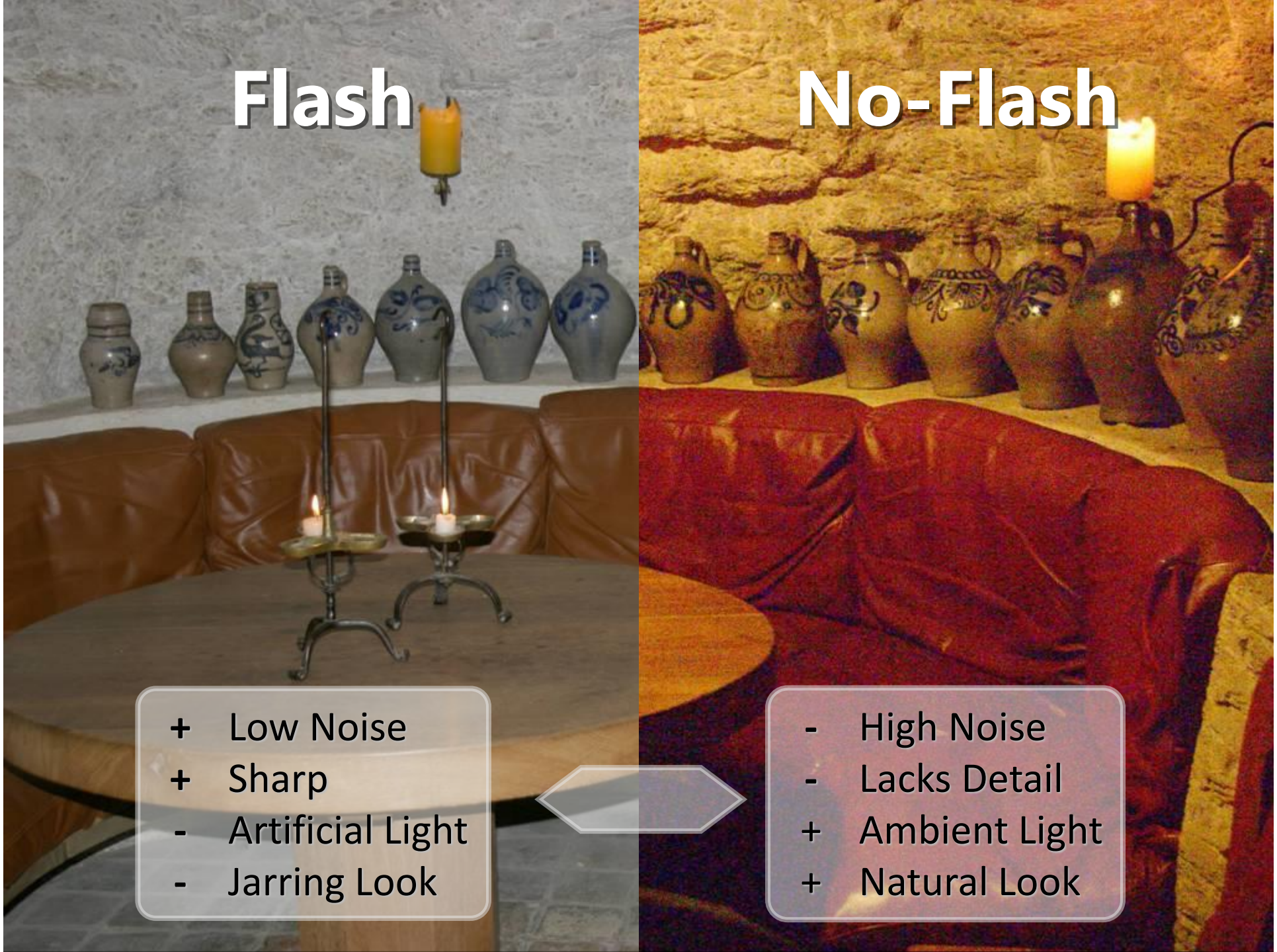
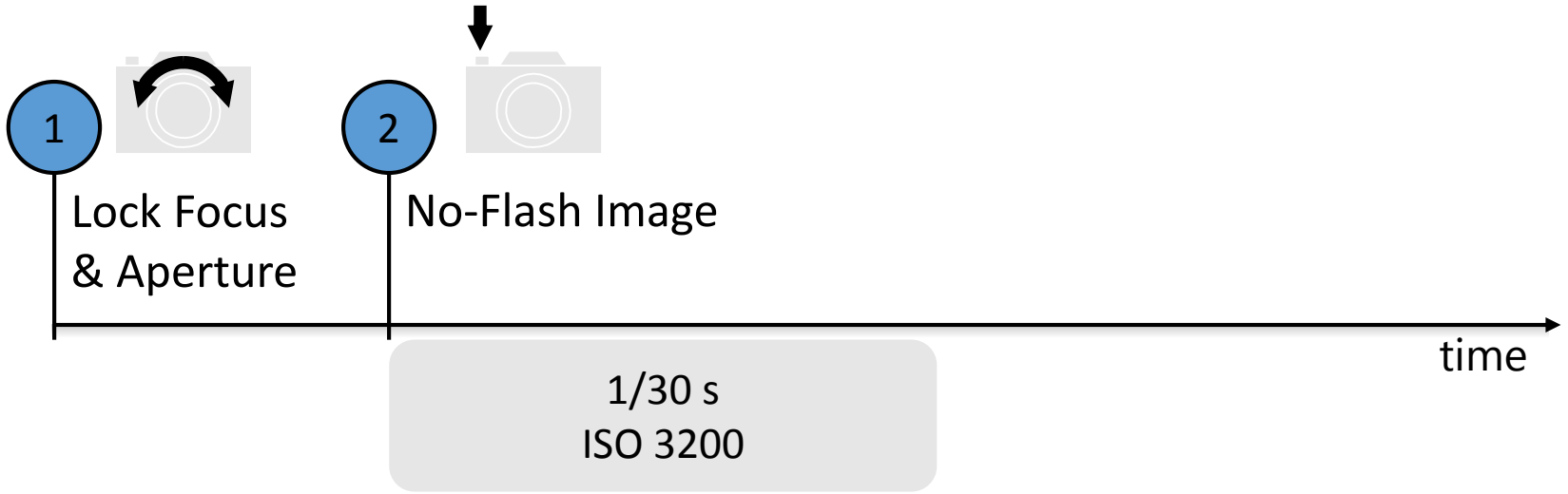

Flash
No-Flash
+ Low Noise
+ Sharp
- Artificial Light
- Jarring Look
- High Noise
- Lacks Detail
+ Ambient Light
+ Natural Look

Image acquisition

Image acquisition



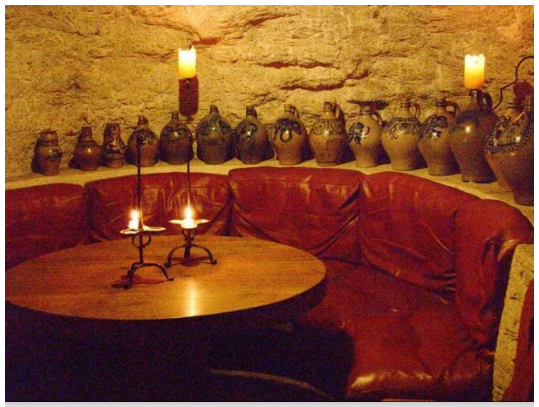
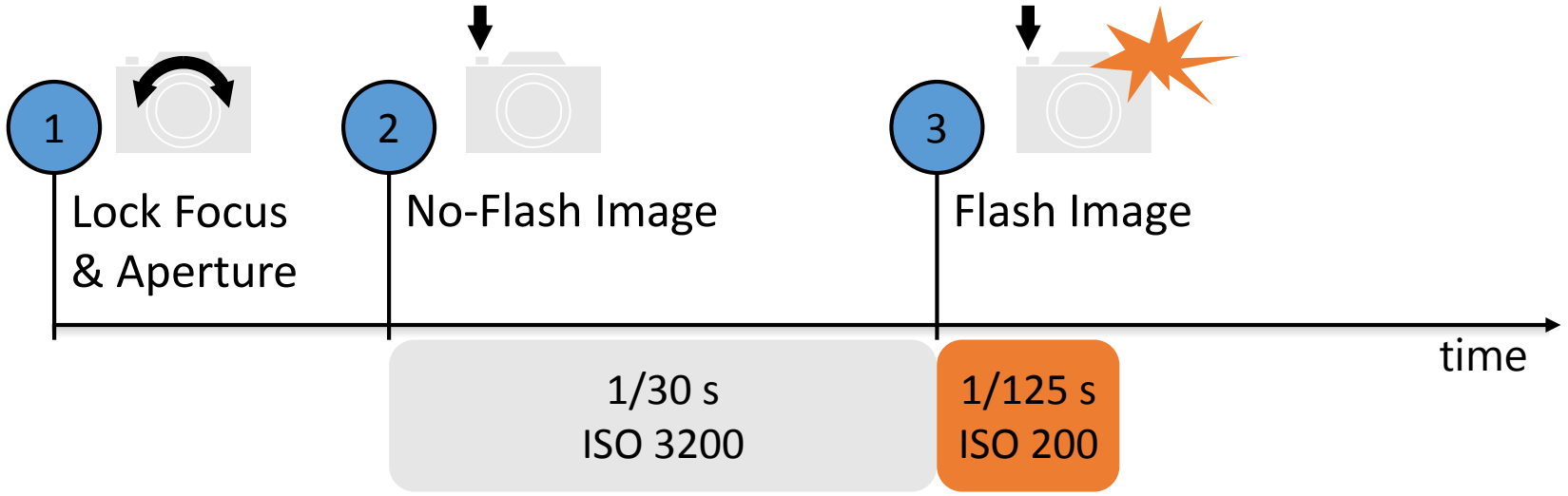
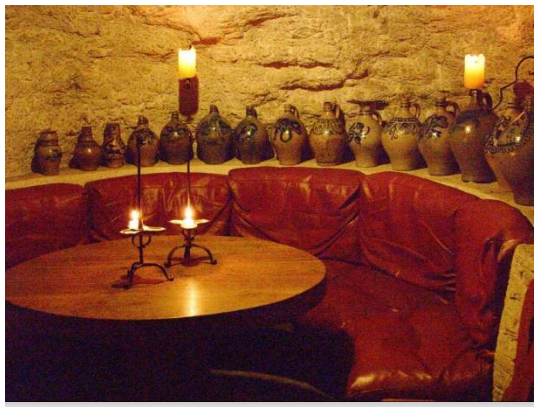


Image acquisition






Denoising Result

- Sh

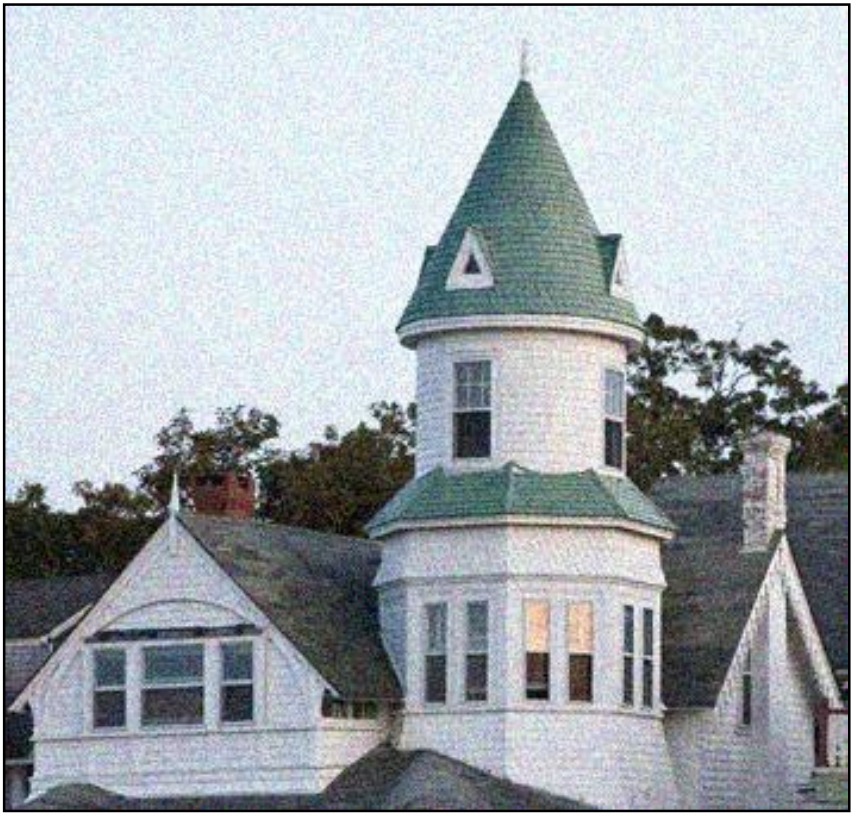

Denoising Result

Key idea

Denoise the no-flash image while maintaining the edge structure of the flash image

- How would you do this using the image editing techniques we've learned about?

Denoising with bilateral filtering



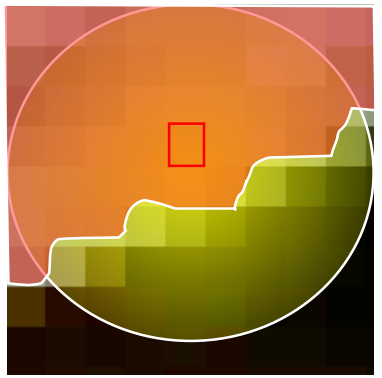
noisy input

bilateral filtering

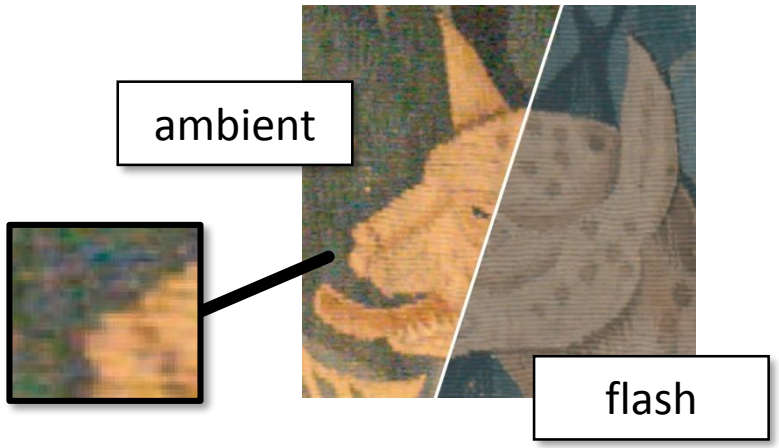
median filtering

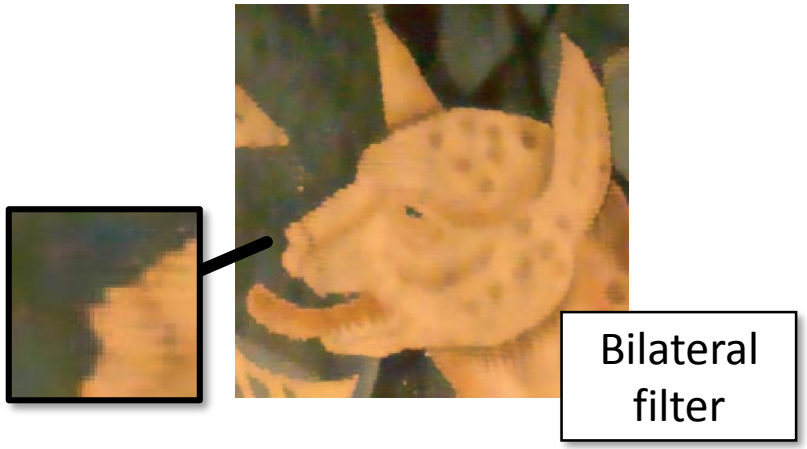
Denoising with bilateral filtering

$$A^{Base}_{p(col)} = \frac{1}{k(p(col))} \sum_{p' \in \Omega} g_d(|p - p'|)\, g_r\left(A_{p(col)} - A_{p'(col)}\right) A_{p'(col)}$$



- However, results still have noise or blur (or both)





Denoising with joint bilateral filtering

$$A^{NR}_{p(col)} = \frac{1}{k(p(col))} \sum_{p' \in \Omega} g_d(|p - p'|) \, g_r(F_{p(col)} - F_{p'(col)}) A_{p'(col)}$$

- In the flash image there are much more *details*
- Use the flash image F to find edges

Denoising with joint bilateral filtering

$$A^{NR}_{p(col)} = \frac{1}{k(p(col))} \sum_{p' \in \Omega} g_d(|p - p'|)\, g_r\left(F_{p(col)} - F_{p'(col)}\right) A_{p'(col)}$$

Bilateral filter

The difference

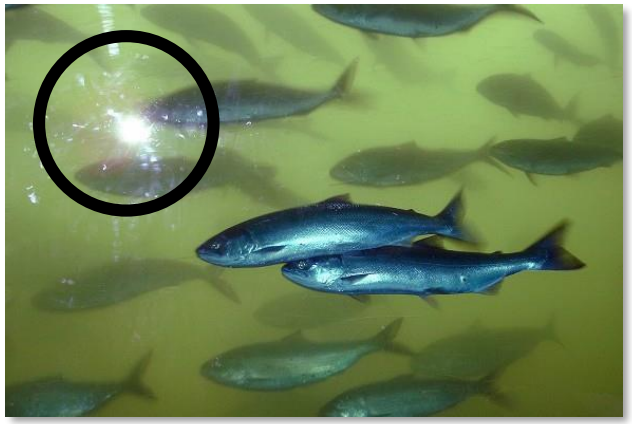
Joint Bilateral filter

Not all edges in the flash image are real

Can you think of any types of edges that may exist in the flash image but not the ambient one?

Not all edges in the flash image are real

shadows



specularities

- May cause over- or under-blur in joint bilateral filter
- We need to eliminate their effect

Detecting shadows

- **Observation:** the pixels in the flash shadow should be similar to the ambient image.
- Not identical:
    1. Noise.
    2. Inter-reflected flash.
- Compute a *shadow mask.*
- Take pixel *p* if $F^{Lin}_{p(col)} - A^{Lin}_{p(col)} \leq \tau_{Shadow}$
- τ_{Shadow} is manually adjusted
- Mask is *smoothed* and *dilated*

Detecting specularities

- Take pixels where sensor input is close to maximum (very bright).
  - Over fixed threshold τ_{Spec}
- Create a *specularity mask.*
- Also smoothed.
- **M** – the combination of shadow and specularity masks:

Where $M_p=1$, we use A^{Base}. For other pixels we use A^{NR}.

Detail transfer

- Denoising cannot add details *missing* in the ambient image
- Exist in flash image because of high SNR
- We use a *quotient image*:

$$F_{p(col)}^{Detail} = \frac{F_{p(col)} + \varepsilon}{F_{p(col)}^{Base} + \varepsilon}$$

Reduces the effect of noise in F

Bilateral filtered

- Multiply with A^{NR} to add the details
- Masked in the same way

Why does this quotient image make sense for detail?

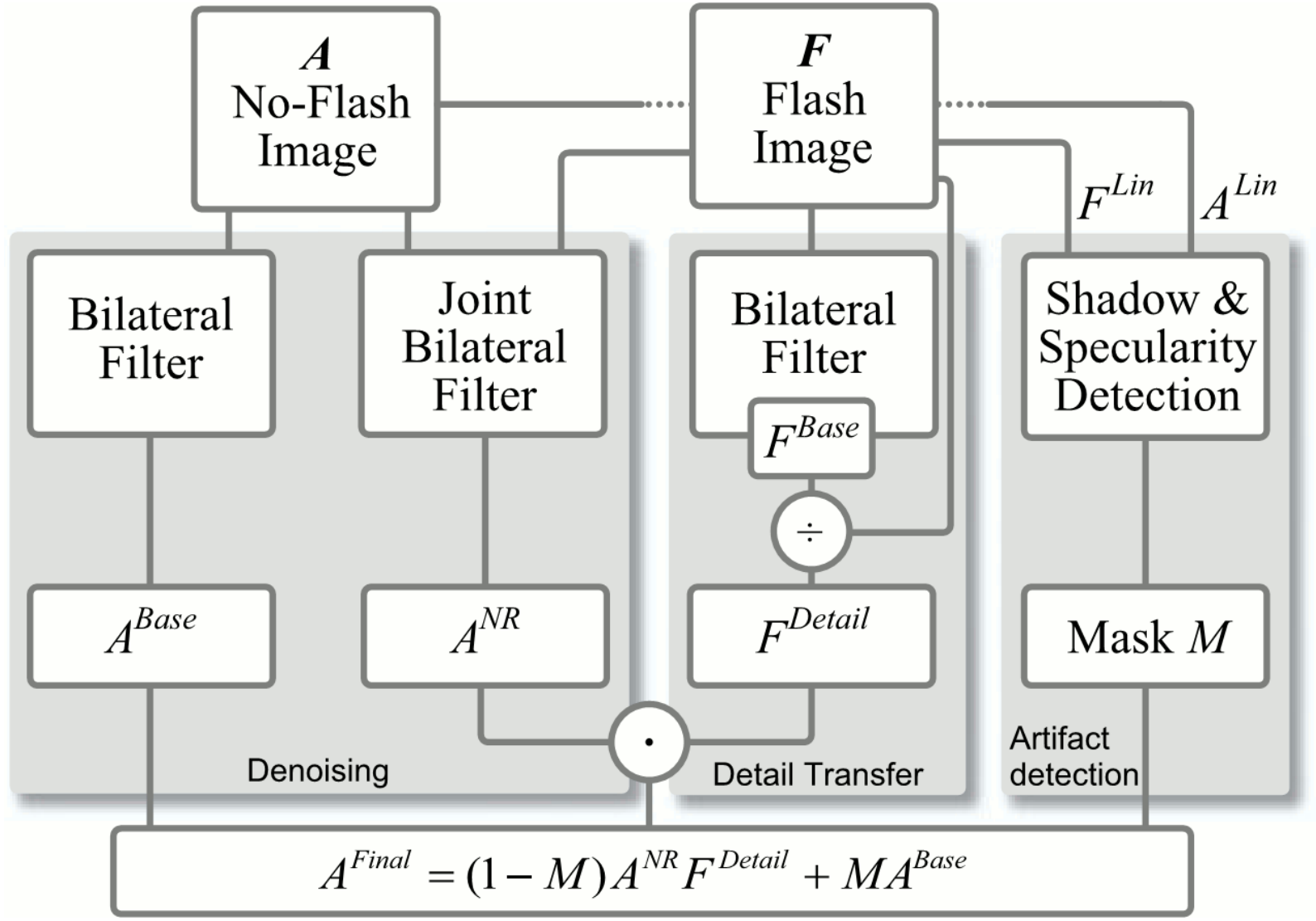
Detail transfer

- Denoising cannot add details *missing* in the ambient image
- Exist in flash image because of high SNR
- We use a *quotient image*:

$$F_{p(col)}^{Detail} = \frac{F_{p(col)} + \varepsilon}{F_{p(col)}^{Base} + \varepsilon}$$

Reduces the effect of noise in F

Full pipeline



Demonstration

ambient-only

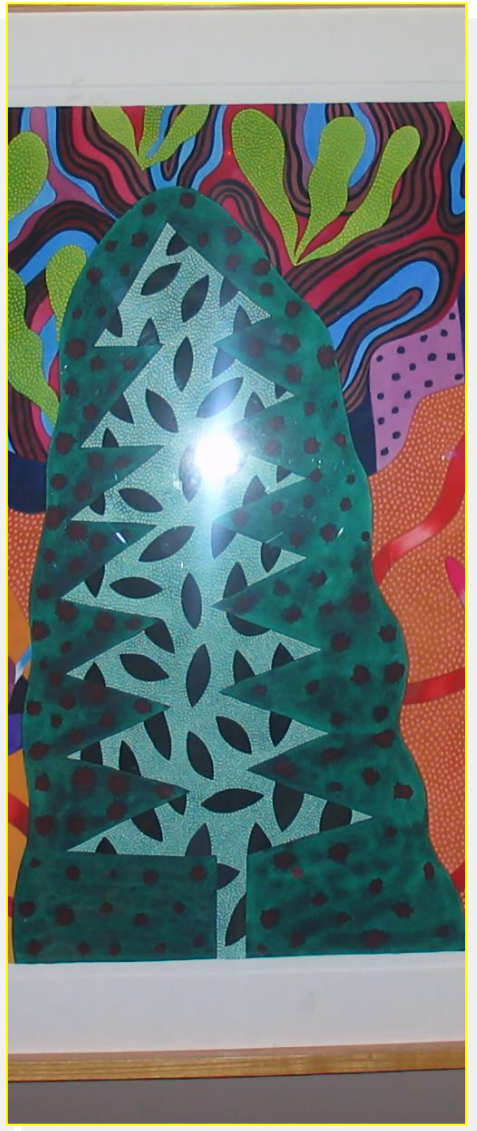
joint bilateral and detail transfer

Can we do similar flash/no-flash fusion tasks with gradient-domain processing?

Removing self-reflections and hot-spots

Ambient

Flash

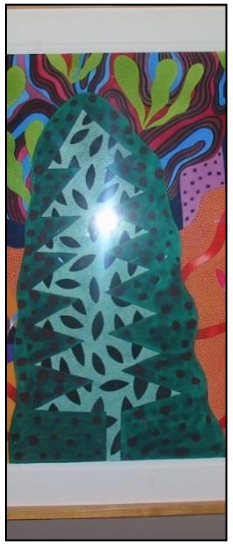


Removing self-reflections and hot-spots

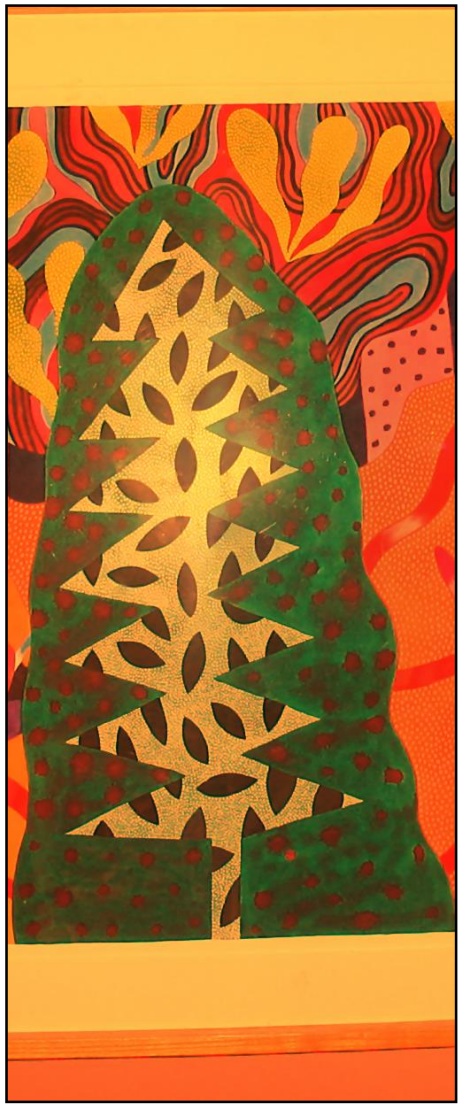
Removing self-reflections and hot-spots

Ambient

Flash



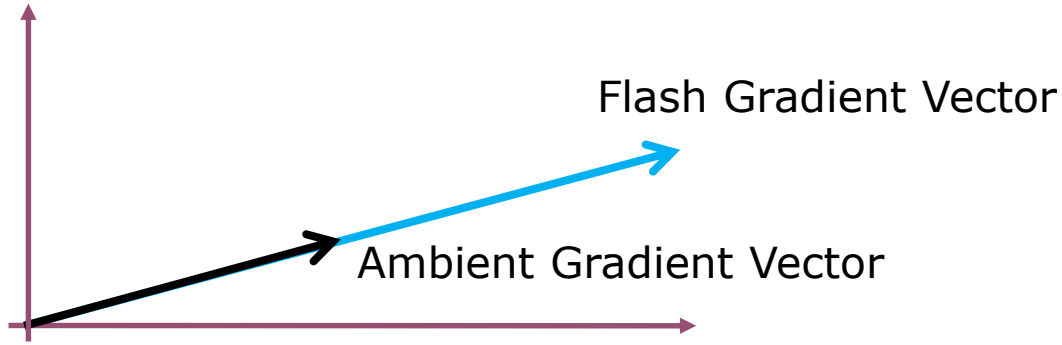
Result

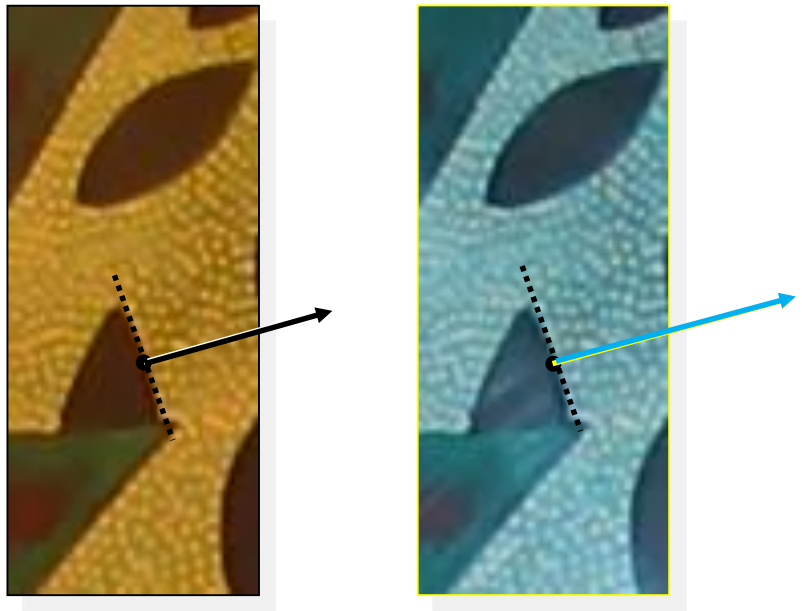


Reflection Layer

Idea: look at how gradients are affected

Same gradient vector direction

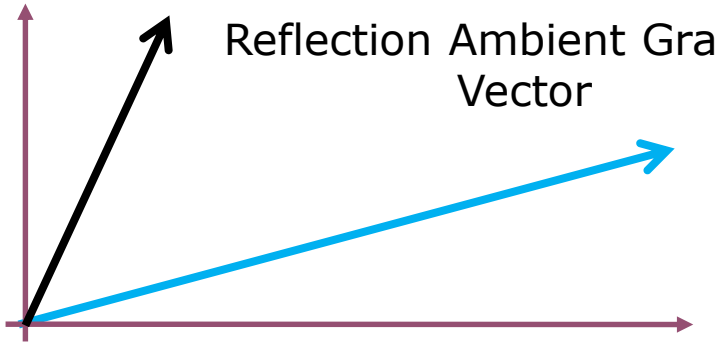


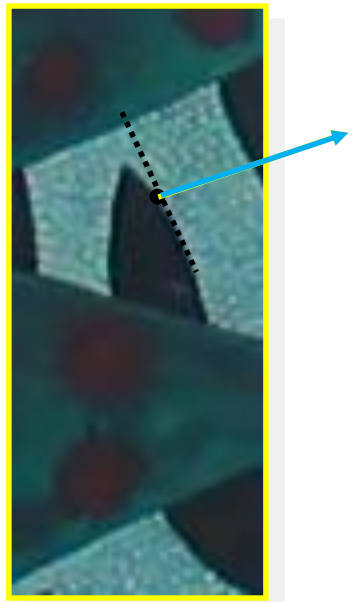


No reflections

Idea: look at how gradients are affected

Different gradient vector direction

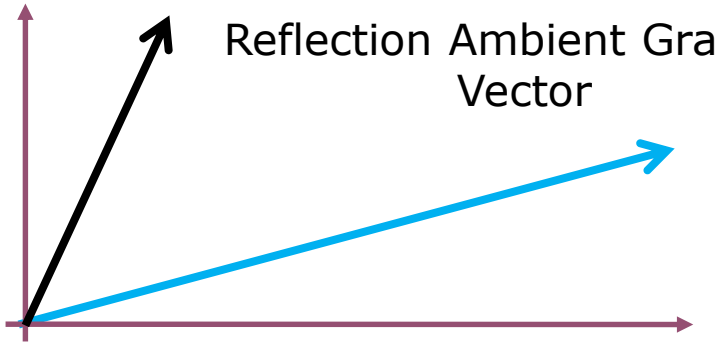


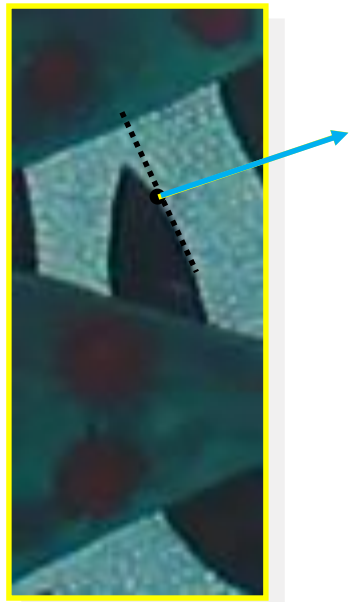


With reflections

Idea: look at how gradients are affected

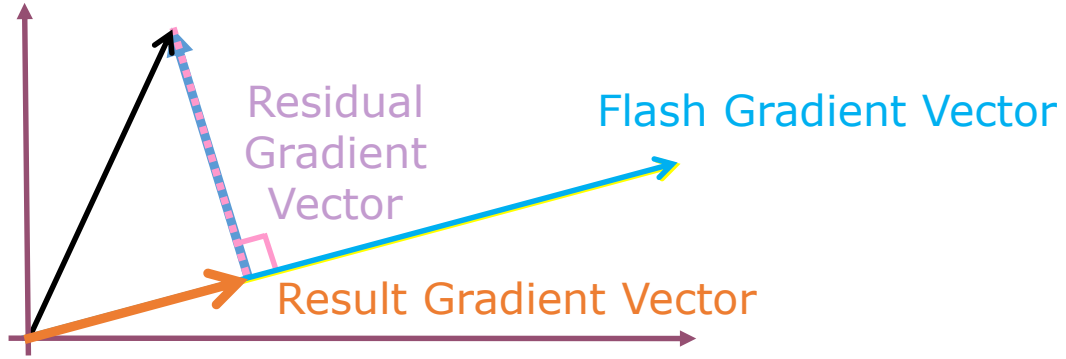
Different gradient vector direction



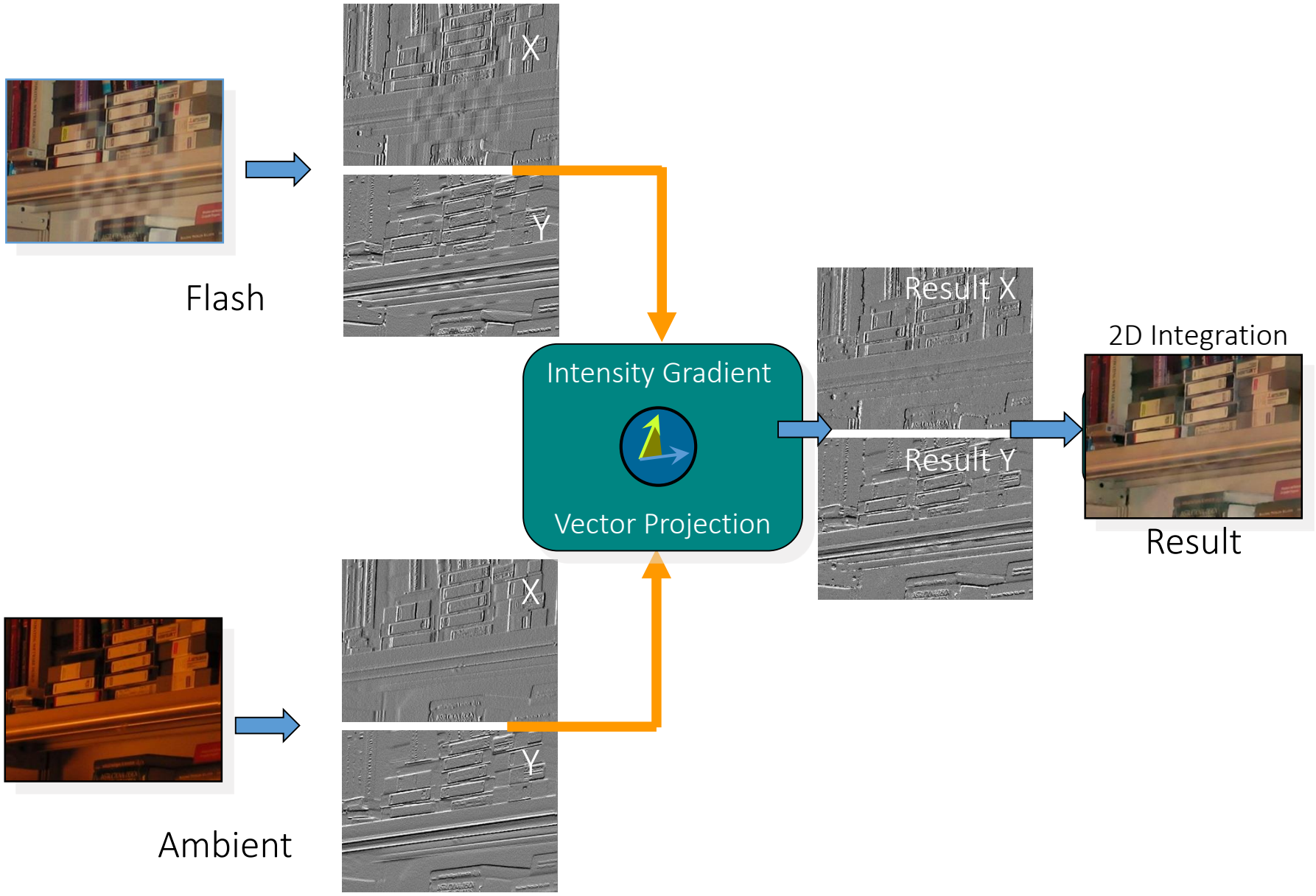


With reflections

Gradient projections



Flash/no-flash with gradient-domain processing




Flash


No-Flash


No-Flash


Result


Flash


No-Flash


No-Flash


Result


Flash


No-Flash


Flash


No-Flash


Result

References

Basic reading:

- Szeliski textbook, Sections 3.13, 3.5.5, 9.3.4, 10.4.3.
- Pérez et al., “Poisson Image Editing,” SIGGRAPH 2003.
  The original Poisson Image Editing paper.
- Agrawal and Raskar, “Gradient Domain Manipulation Techniques in Vision and Graphics,” ICCV 2007 course, http://www.amitkagrawal.com/ICCV2007Course/
  A great resource (entire course!) for gradient-domain image processing.
- Petschnigg et al., “Digital photography with flash and no-flash image pairs,” SIGGRAPH 2004.
- Eisemann and Durand, “Flash Photography Enhancement via Intrinsic Relighting,” SIGGRAPH 2004.
  The first two papers exploring the idea of photography with flash and no-flash pairs, both using variants of the joint bilateral filter.
- Agrawal et al., “Removing Photography Artifacts Using Gradient Projection and Flash-Exposure Sampling,” SIGGRAPH 2005.
  A subsequent paper on photography with flash and no-flash pairs, using gradient-domain image processing.

Additional reading:

- Georgiev, “Covariant Derivatives and Vision,” ECCV 2006.
  An paper from Adobe on the version of Poisson blending implemented in Photoshop’s “healing brush”.
- Elder and Goldberg, “Image editing in the contour domain”, PAMI 2001.
  One of the very first papers discussing gradient-domain image processing.
- Szeliski, “Locally adapted hierarchical basis preconditioning,” SIGGRAPH 2006.
  A standard reference on multi-grid and preconditioning techniques for accelerating the Poisson solver.
- Bhat et al., “Fourier Analysis of the 2D Screened Poisson Equation for Gradient Domain Problems,” ECCV 2008.
  A paper discussing the (Fourier) basis projection approach for solving the Poisson integration problem.
- Bhat et al., “GradientShop: A Gradient-Domain Optimization Framework for Image and Video Filtering,” ToG 2010.
  A paper describing gradient-domain processing as a general image processing paradigm, which can be used for a broad set of applications beyond blending, including tone-mapping, colorization, converting to grayscale, edge enhancement, image abstraction and non-photorealistic rendering.
- Krishnan and Fergus, “Dark Flash Photography,” SIGGRAPH 2009.
  A paper proposing doing flash/no-flash photography using infrared flash lights.
- Kettunen et al., “Gradient-domain path tracing,” SIGGRAPH 2015.
  In addition to *editing* images in the gradient-domain, we can also directly *render* them in the gradient-domain.
- Tumblin et al., “Why I want a gradient camera?” CVPR 2005.
  We can even directly *measure* images in the gradient domain, using so-called *gradient cameras*.