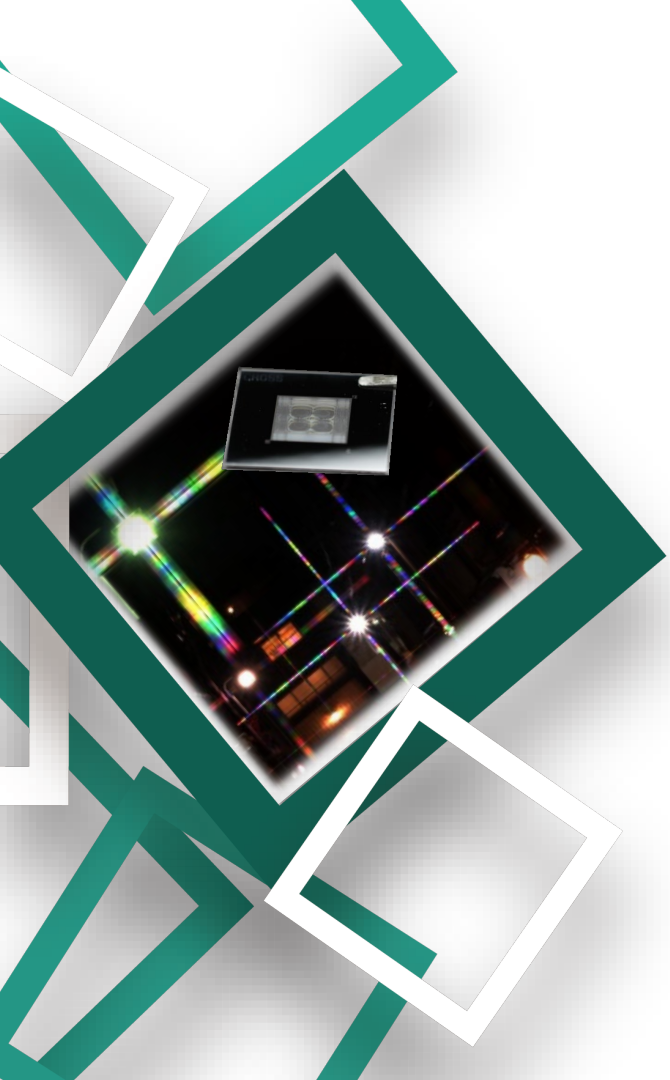
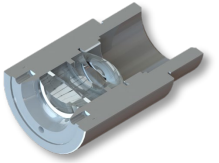
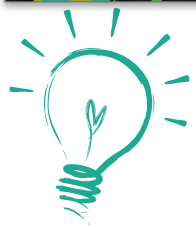






Computational Imaging

Lecture 12: Computing Toolbox:
Image Gradient and Processing II

Qilin Sun（孙启霖）

School of Data Science

The Chinese University of Hong Kong, Shenzhen

- Efficient Poisson Solver
- Poisson Image Editing
- Flash/no-flash Photography
- Gradient Cameras
  - Diff Sensor
  - Diff Optics
  - Diff Time

Efficient Poisson Solver

The 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 vector field **v**

f is equivalent to f* at the boundaries

Input vector field:

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

Recall ...

Nabla operator definition

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

And for discrete images:

partial-x derivative filter

1	-1

partial-y derivative filter

1
-1

The 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 ...

Nabla operator definition

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

And for discrete images:

partial-x derivative filter

1	-1

partial-y derivative filter

1
-1

The 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.

How to solve this optimization problem?

Recall ...

Nabla operator definition

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

And for discrete images:

partial-x derivative filter

1	-1

partial-y derivative filter

1
-1

Solution 1: Compute Stationary Points

Given the loss function:

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

Compute its derivative:

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

Set that to zero:

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

It is equal to the vector b we derived previously!

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

Solution 2: Gradient Descent

Given the loss function:

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

Compute its derivative:

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

The ***residual***

Iteratively compute a solution:

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

for i = 0, 1, …, N,
where η^i are positive *step sizes*

Solution 2: Optimal Step Sizes

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

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

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

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

Solution 2: Gradient Descent

Given the loss function:

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

Minimize by iteratively computing:

$$r^i = b - Af^i, \quad \eta^i = \frac{(r^i)^T r^i}{(r^i)^T A r^i}, \quad f^{i+1} = f^i + \eta^i r^i, \quad i = 0, \dots, N$$

Is this cheaper than the pseudo-inverse approach?

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

Poractical Solution: Conjugate Gradient (CG)

Given the loss function:

Minimize by iteratively computing:

$$d^i = r^i + \beta^i d^i, \quad \eta^i = \frac{(r^i)^T r^i}{(d^i)^T A d^i}, \quad f^{i+1} = f^i + \eta^i d^i, \quad i = 0, \dots, N$$

$$r^{i+1} = r^i - \eta^i A d^i, \quad \beta^i = \frac{(r^{i+1})^T r^{i+1}}{(r^i)^T r^i}$$

- Smarter way for selecting **update** directions
- Everything can still be done using **convolutions**
- Only **one** convolution needed per iteration

Poractical Solution: CG Initialization

Does the initialization f^0 matter?

- It doesn't matter. The loss is convex.

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

It does matter in terms of **convergence speed**.

- We can use a *multi-scale* approach (like mipmap):
  - Solve an initial problem for a very low-resolution f (e.g., 2x2).
  - Use the solution to initialize gradient descent for a higher resolution f (e.g., 4x4).

  ...

  - Use the solution to initialize gradient descent for an f with the original resolution

PxP.

- *Multi-grid* algorithms alternative between higher and lower resolutions during the (conjugate) gradient descent iterative procedure.

Reminder from Variational Case

Poisson equation (with Dirichlet boundary conditions)

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

After discretization, equivalent to:

$$\begin{bmatrix} D & I & 0 & 0 & 0 & \cdots & 0 \\ I & D & I & 0 & 0 & \cdots & 0 \\ 0 & I & D & I & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & I & D & I & 0 \\ 0 & \cdots & \cdots & 0 & I & D & I \\ 0 & \cdots & \cdots & \cdots & 0 & I & D \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ \vdots \\ f_{q_1} \\ \vdots \\ f_{q_2} \\ f_p \\ f_{q_3} \\ \vdots \\ f_{q_4} \\ \vdots \\ f_P \end{bmatrix} = \begin{bmatrix} (\nabla \cdot \mathbf{v})_1 \\ \vdots \\ (\nabla \cdot \mathbf{v})_{q_1} \\ \vdots \\ (\nabla \cdot \mathbf{v})_{q_2} \\ (\nabla \cdot \mathbf{v})_p \\ (\nabla \cdot \mathbf{v})_{q_3} \\ \vdots \\ (\nabla \cdot \mathbf{v})_{q_4} \\ \vdots \\ (\nabla \cdot \mathbf{v})_P \end{bmatrix}$$

Linear system of equations:

$$Af = b$$

Remember that what we are doing is equivalent to solving this linear system.

Note: Preconditioning

We are solving this linear system:

$$Af = b$$

For any **invertible matrix P,** this is equivalent to solving:

$$P^{-1}Af = P^{-1}b$$

Preconditioning can be incorporated in the conjugate gradient descent algorithm.

When is it preferable to solve this alternative linear system?

- Ideally: If A is invertible, and P is the same as A, the linear system becomes trivial! But computing the inverse of A is even more expensive than solving the original linear system.
- In practice: If the matrix $P^{-1}A$ has a better condition number, or its singular values are more uniformly distributed, the linear system becomes more *numerically stable*.

What *preconditioner* P should we use?

- Standard preconditioners like Jacobi.
- More effective preconditioners. Active area of research.

Is this effective for Poisson solvers?

$$P_{\text{Jacobi}} = \text{diag}(A)$$

Reminder from Variational Case

Poisson equation (with Dirichlet boundary conditions)

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

After discretization, equivalent to:

$$\begin{bmatrix} D & I & 0 & 0 & 0 & \cdots & 0 \\ I & D & I & 0 & 0 & \cdots & 0 \\ 0 & I & D & I & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & I & D & I & 0 \\ 0 & \cdots & \cdots & 0 & I & D & I \\ 0 & \cdots & \cdots & \cdots & 0 & I & D \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ \vdots \\ f_{q_1} \\ \vdots \\ f_{q_2} \\ f_p \\ f_{q_3} \\ \vdots \\ f_{q_4} \\ \vdots \\ f_P \end{bmatrix} = \begin{bmatrix} (\nabla \cdot \mathbf{v})_1 \\ \vdots \\ (\nabla \cdot \mathbf{v})_{q_1} \\ \vdots \\ (\nabla \cdot \mathbf{v})_{q_2} \\ (\nabla \cdot \mathbf{v})_p \\ (\nabla \cdot \mathbf{v})_{q_3} \\ \vdots \\ (\nabla \cdot \mathbf{v})_{q_4} \\ \vdots \\ (\nabla \cdot \mathbf{v})_P \end{bmatrix}$$

Linear system of equations:

$$Af = b$$

Matrix is $P \times P \rightarrow$ billions of entries

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

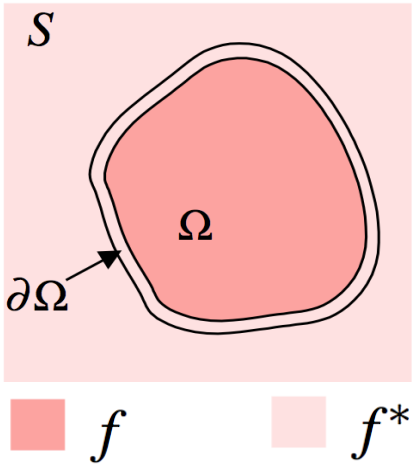
Note: (Dirichlet) Boundary Bonditions

- Form a **mask B** that is 0 for pixels that should ***not*** **be updated** (pixels on S-Ω and $\partial\Omega$) and 1 otherwise.
- Use **convolution** to perform **Laplacian** filtering over the *entire image*.
- Use (conjugate) gradient descent rules to only update pixels for which the mask is 1. Equivalently, change the update rules to:

$$f^{i+1} = f^i + B\eta^i r^i \qquad \text{(gradient descent)}$$

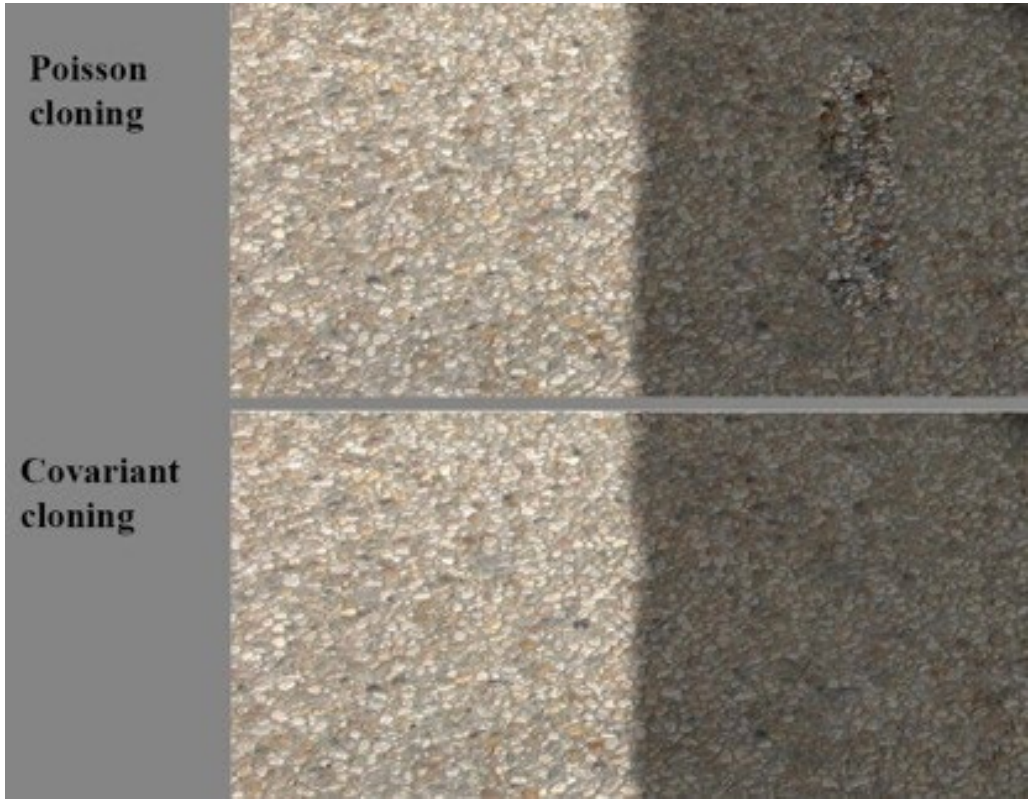
$$f^{i+1} = f^i + B\eta^i d^i \qquad \text{(conjugate gradient descent)}$$

In practice, masking is also required at other steps of (conjugate) gradient descent, to deal with invalid boundaries (e.g., from convolutions).



Poisson Image Editing Examples

Photoshop's "Healing Brush"



Slightly more advanced version of what we covered here:

- Uses higher-order derivatives

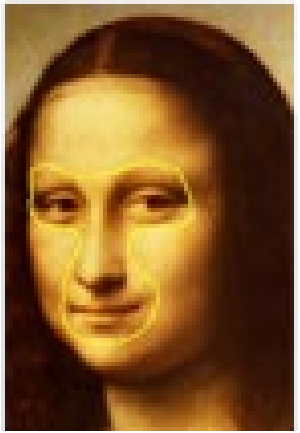
Contrast Problem

Loss of **contrast** when pasting from dark to bright:

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

Solution: Do blending in **log-domain**.

More Blending



originals

copy-paste

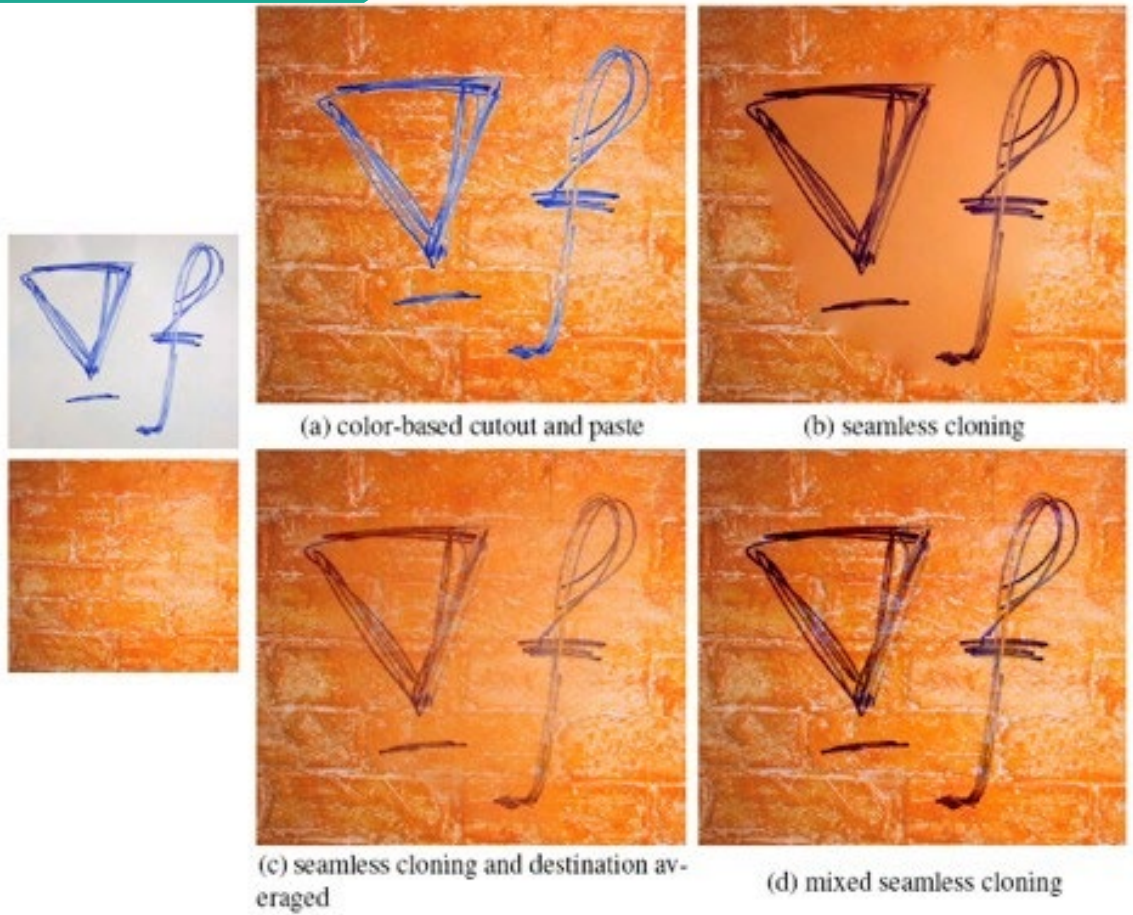
Poisson blending

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

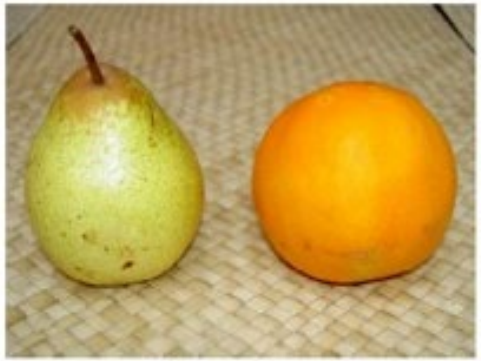
Editing

Concealment

How would you do this with Poisson blending?

- Insert a copy of the background.

Texture Swapping



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

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

Entire Suite of Image Editing Tools

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

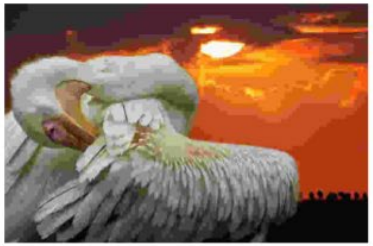
Pravin Bhat[1] C. Lawrence Zitnick[2] Michael Cohen[1,2] Brian Curless[1]
[1]University of Washington [2]Microsoft Research


(a) Input image

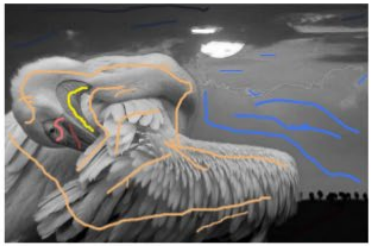

(b) Saliency-sharpening filter


(c) Pseudo-relighting filter


(d) Non-photorealistic rendering filter

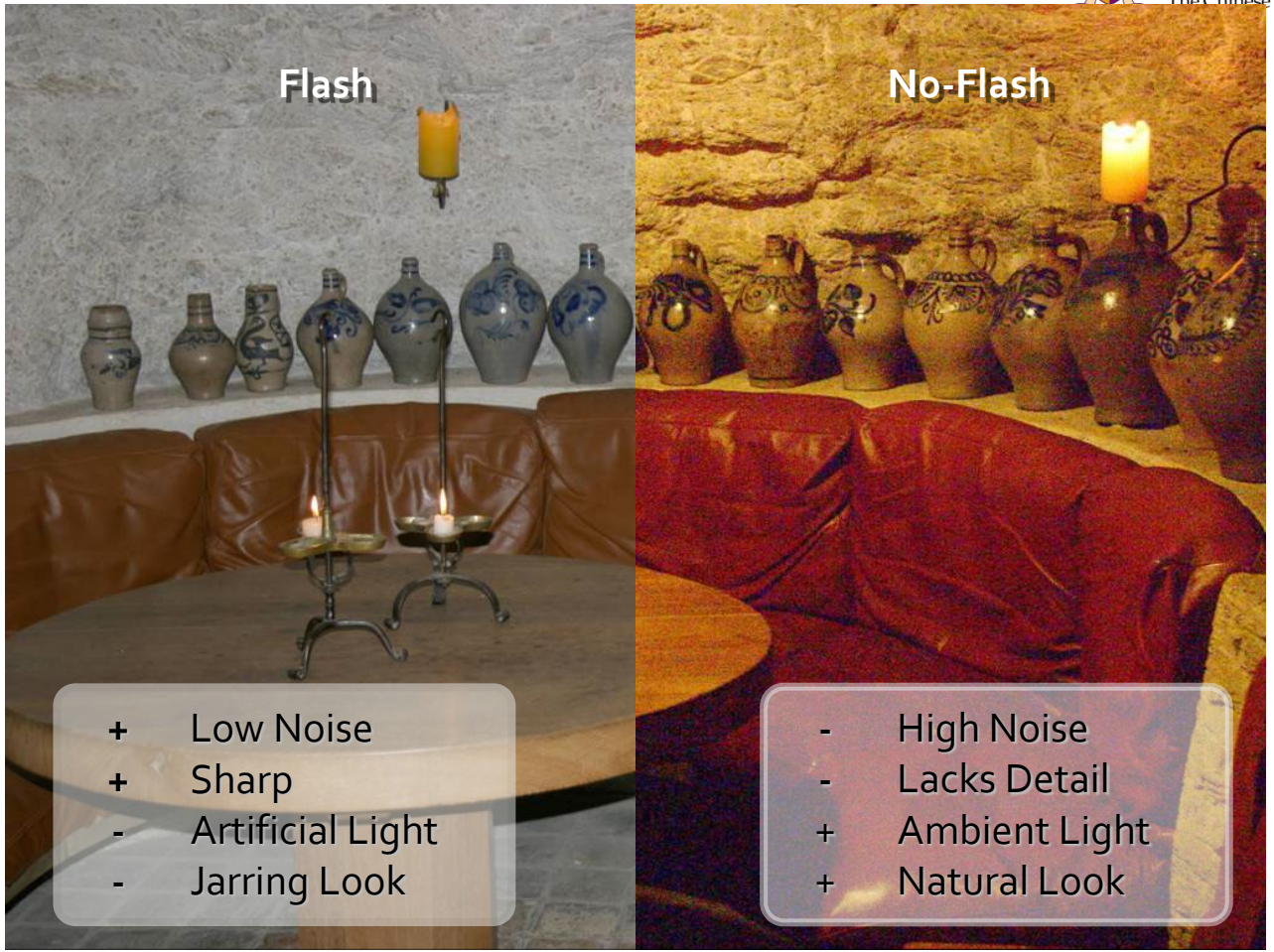

(e) Compressed input-image


(f) De-blocking filter


(g) User input for colorization


(h) Colorization filter

Flash/no-flash Photography


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


Denoising Result


No-Flash

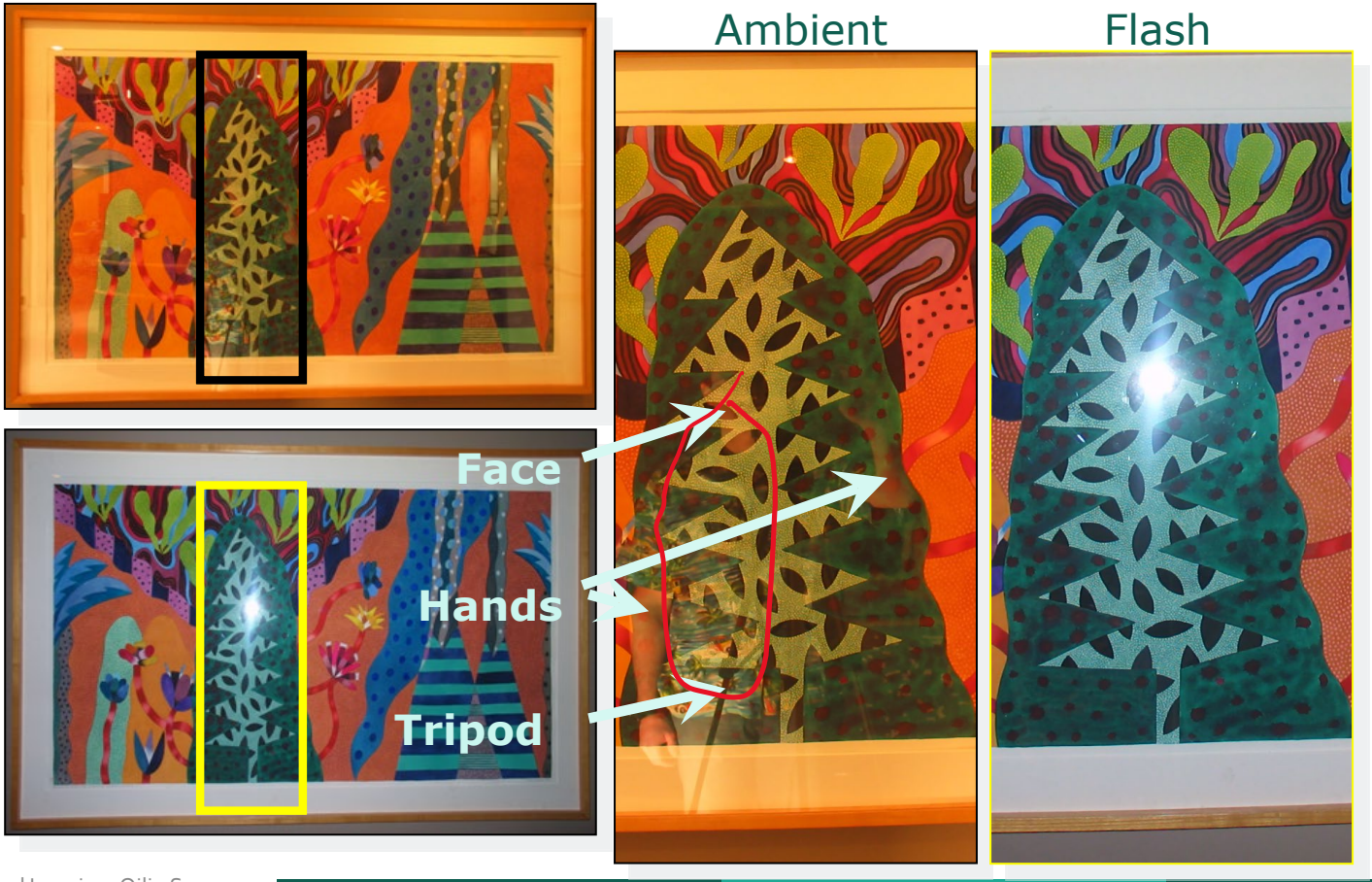

Denoising Result

Key Idea

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

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

Removing Self-reflections and Hot-spots



Removing Self-reflections and Hot-spots

Ambient

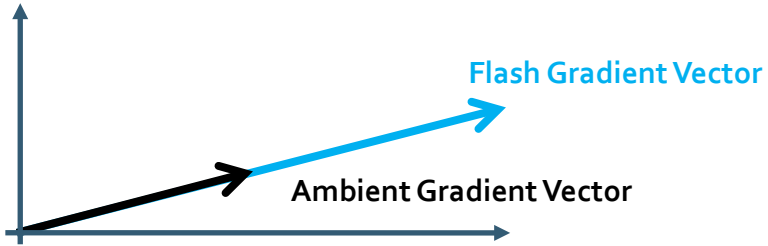
Flash

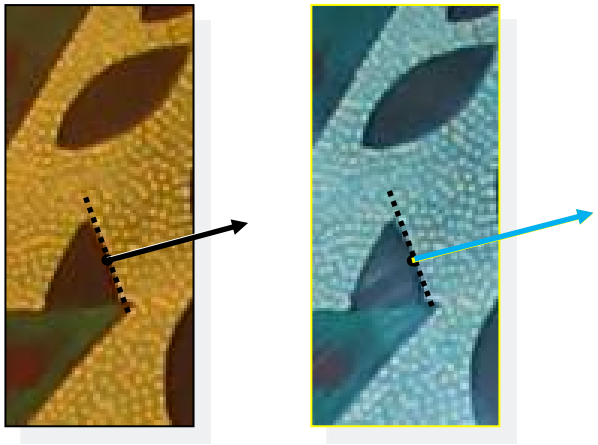
Result

Reflection Layer

Idea: Look at How Gradients are Affected

Same gradient vector direction

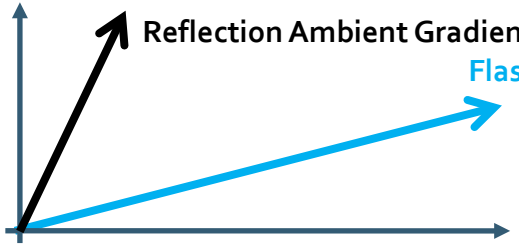


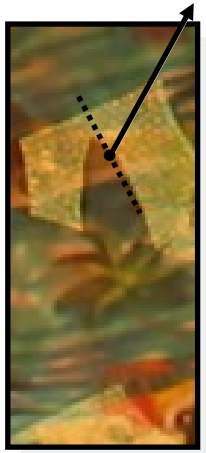


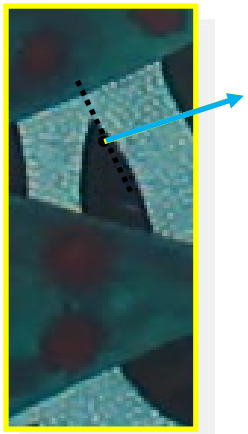
No reflections

Idea: Look at How Gradients are Affected

Different gradient vector direction

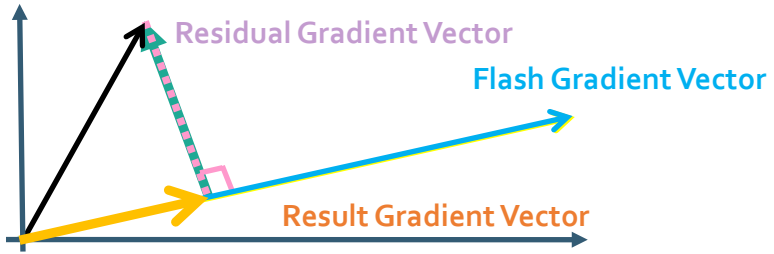






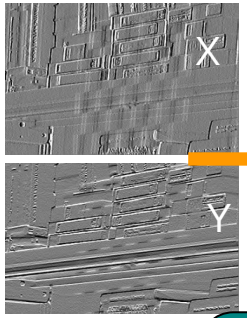
With reflections

Gradient Projections



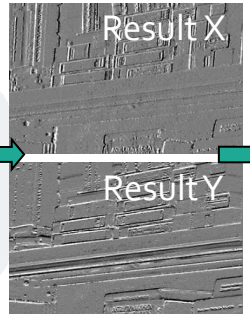
Flash/no-flash with Gradient-domain Processing

Flash



Intensity Gradient

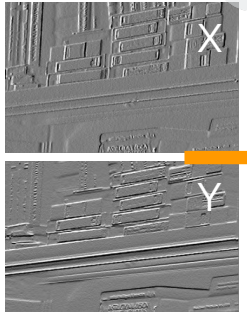
Vector Projection



2D Integration

Result

Ambient



Gradient Cameras


What implication would this have on a camera?
gradients of natural images are *sparse* (close to zero in most places)
Primal domain
Gradient domain

Why Gradient Cameras?

Why I want a Gradient Camera

Jack Tumblin
Northwestern University
jet@cs.northwestern.edu

Amit Agrawal
University of Maryland
aagrawal@umd.edu

Ramesh Raskar
MERL
raskar@merl.com

Why a gradient camera?

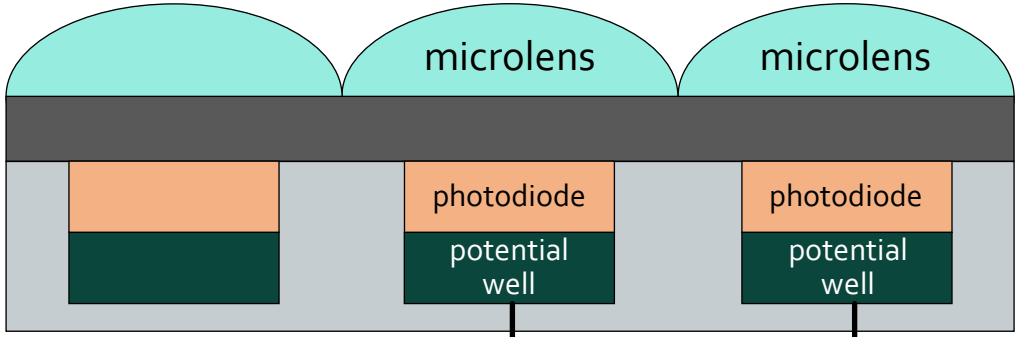
- **Faster** frame rate, only very few pixels (gradient).
- **Higher dynamic range,** if also combined with logarithmic gradients.

Can you directly display the measurements of such a camera?

- Use a Poisson solver to reconstruct the image from the measured gradients.

How to build a gradient camera?

Change the Sensor



Any disadvantages?

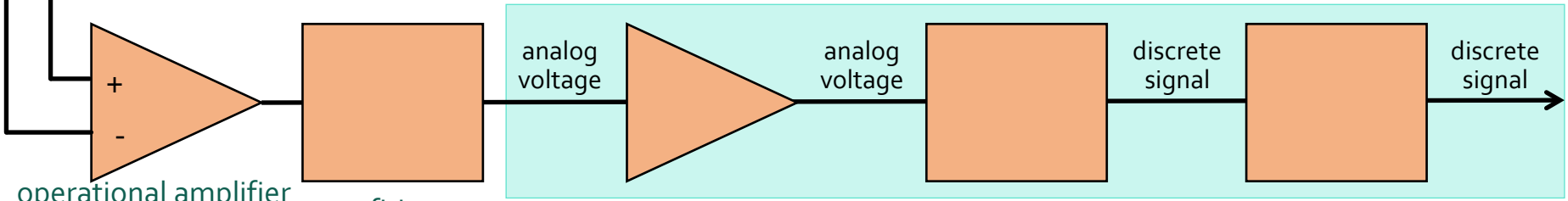
- Spatial resolution is reduced by 2x.
- Photosensitive area is reduced.

Why is this better than computing gradients in post-processing?

- Additive noise is reduced.
- Acquisition is faster thanks to the firing mechanism and sparsity of edges.

What about Poisson noise?

- Poisson noise is the same in both cases.

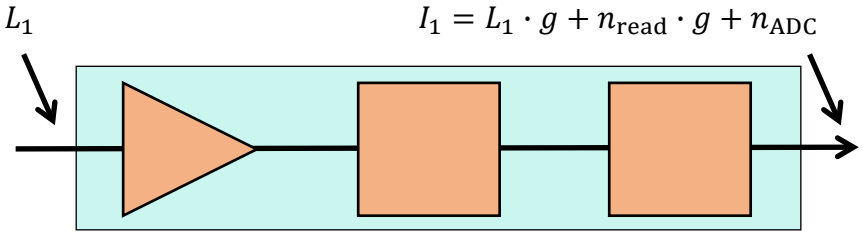


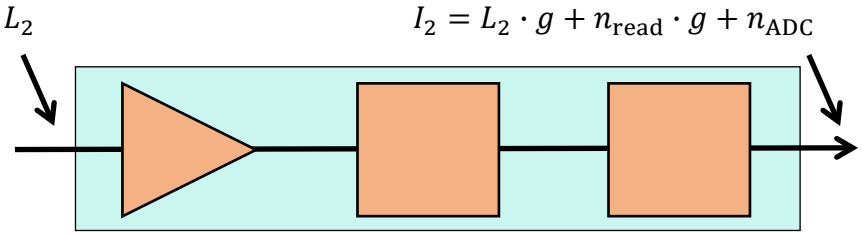
operational amplifier (amplify difference of inputs)

firing mechanism

typical analog front-end

Noise of Diff Sensor





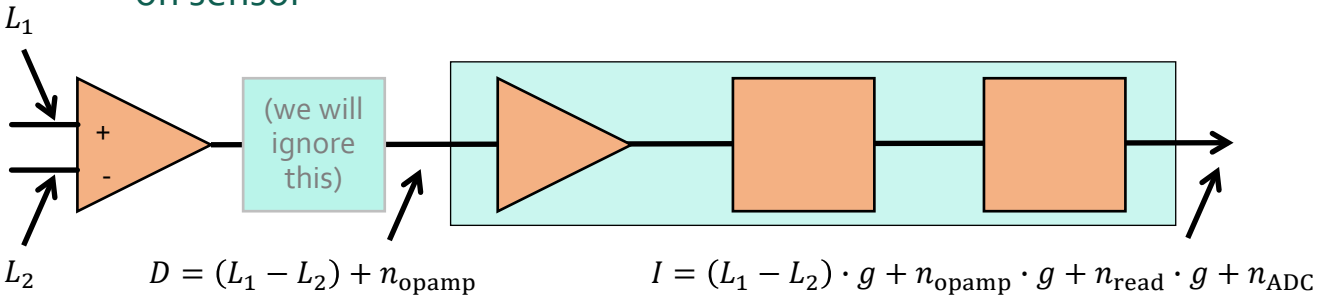
Digital subtraction in post-processing

$$\sigma(I_1 - I_2)^2 = \sigma(L_1 - L_2)^2 + 2 \cdot \sigma_{\text{read}}^2 \cdot g^2 + 2 \cdot \sigma_{\text{ADC}}^2$$

which variance is better?

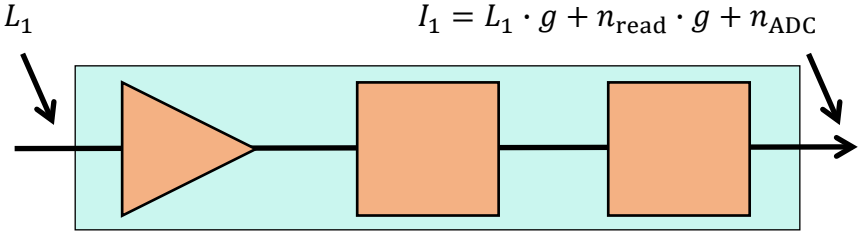
Analog subtraction on sensor

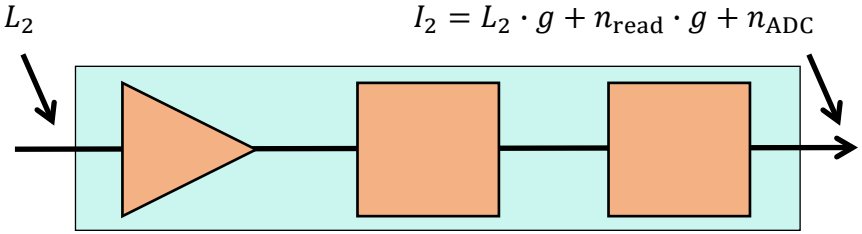
$$\sigma(I)^2 = \sigma(L_1 - L_2)^2 + \sigma_{\text{opamp}}^2 \cdot g^2 + \sigma_{\text{read}}^2 \cdot g^2 + \sigma_{\text{ADC}}^2$$



$$L_1 \sim \text{Poisson}(t \cdot (a \cdot \Phi_1 + D))$$
$$L_2 \sim \text{Poisson}(t \cdot (a \cdot \Phi_2 + D))$$
$$n_{\text{opamp}} \sim \text{Normal}(0, \sigma_{\text{opamp}})$$
$$n_{\text{read}} \sim \text{Normal}(0, \sigma_{\text{read}})$$
$$n_{\text{ADC}} \sim \text{Normal}(0, \sigma_{\text{ADC}})$$

Noise of Diff Sensor





Digital subtraction in post-processing

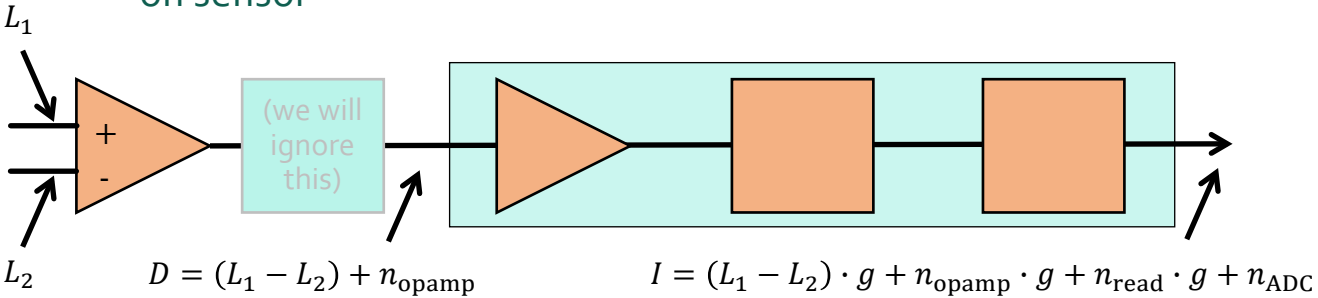
$$\sigma(I_1 - I_2)^2 = \boxed{\sigma(L_1 - L_2)^2} + \boxed{2 \cdot \sigma_{\text{read}}^2 \cdot g^2 + 2 \cdot \sigma_{\text{ADC}}^2}$$

terms related to Poisson noise are the same

additive noise is reduced if opamp is well-designed

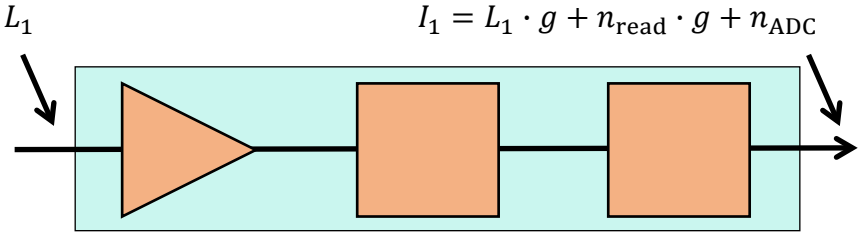
Analog subtraction on sensor

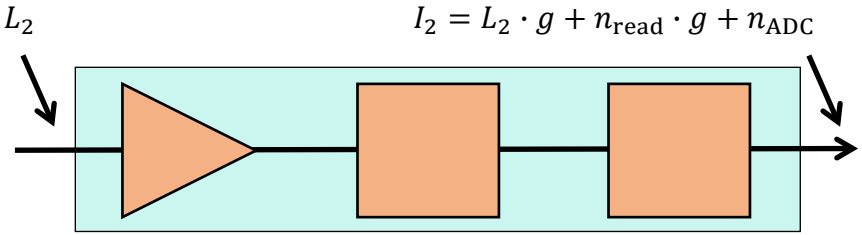
$$\sigma(I)^2 = \boxed{\sigma(L_1 - L_2)^2} + \boxed{\sigma_{\text{opamp}}^2 \cdot g^2 + \sigma_{\text{read}}^2 \cdot g^2 + \sigma_{\text{ADC}}^2}$$



$$L_1 \sim \text{Poisson}(t \cdot (a \cdot \Phi_1 + D))$$
$$L_2 \sim \text{Poisson}(t \cdot (a \cdot \Phi_2 + D))$$
$$n_{\text{opamp}} \sim \text{Normal}(0, \sigma_{\text{opamp}})$$
$$n_{\text{read}} \sim \text{Normal}(0, \sigma_{\text{read}})$$
$$n_{\text{ADC}} \sim \text{Normal}(0, \sigma_{\text{ADC}})$$

Noise of Diff Sensor





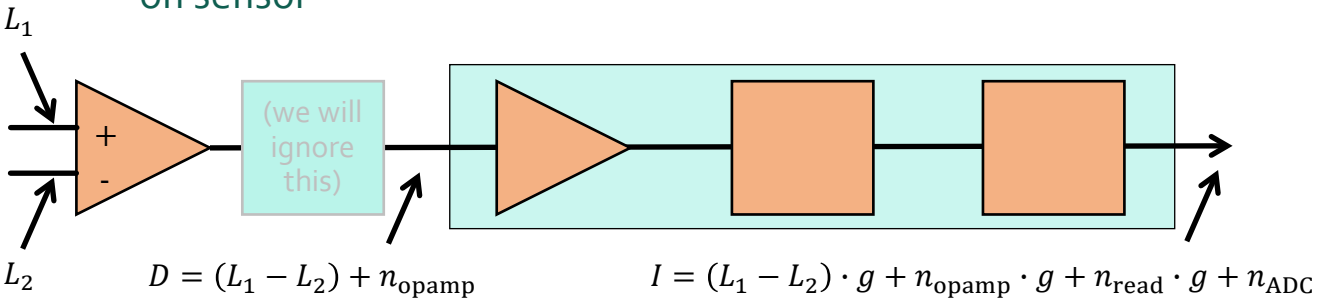
Digital subtraction in post-processing

$$\sigma(I_1 - I_2)^2 = \boxed{\sigma(L_1 - L_2)^2} + 2 \cdot \sigma_{\text{read}}^2 \cdot g^2 + 2 \cdot \sigma_{\text{ADC}}^2$$

what is the distribution of the difference $L_1 - L_2$?

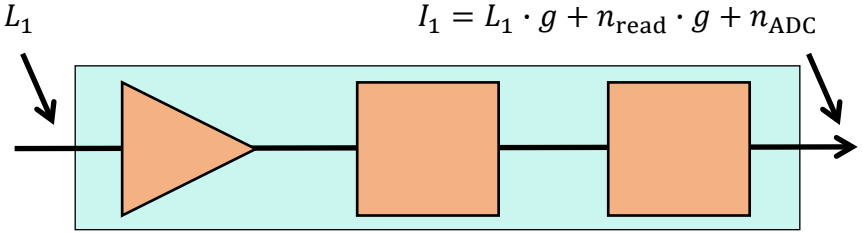
Analog subtraction on sensor

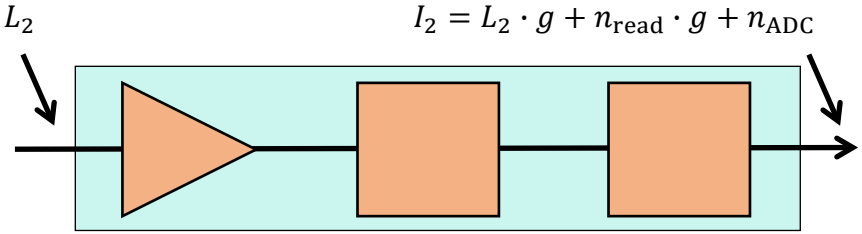
$$\sigma(I)^2 = \boxed{\sigma(L_1 - L_2)^2} + \sigma_{\text{opamp}}^2 \cdot g^2 + \sigma_{\text{read}}^2 \cdot g^2 + \sigma_{\text{ADC}}^2$$



$$L_1 \sim \text{Poisson}(t \cdot (a \cdot \Phi_1 + D))$$
$$L_2 \sim \text{Poisson}(t \cdot (a \cdot \Phi_2 + D))$$
$$n_{\text{opamp}} \sim \text{Normal}(0, \sigma_{\text{opamp}})$$
$$n_{\text{read}} \sim \text{Normal}(0, \sigma_{\text{read}})$$
$$n_{\text{ADC}} \sim \text{Normal}(0, \sigma_{\text{ADC}})$$

Noise of Diff Sensor





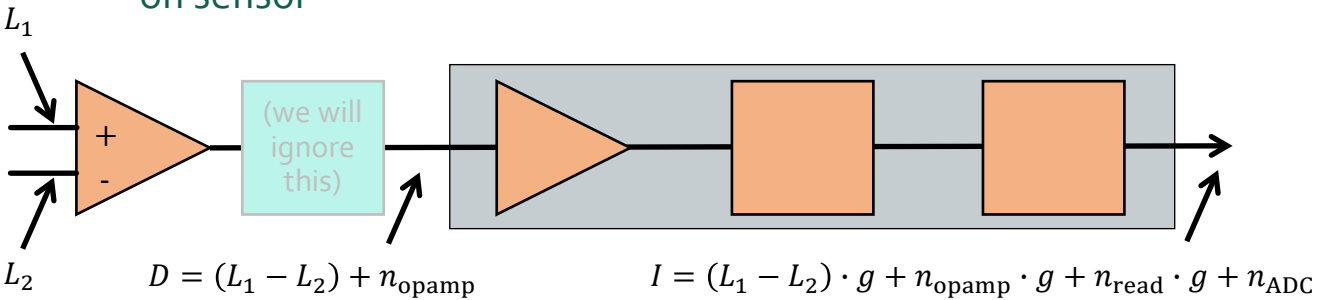
Digital subtraction in post-processing

$$\sigma(I_1 - I_2)^2 = \boxed{\sigma(L_1 - L_2)^2} + 2 \cdot \sigma_{\text{read}}^2 \cdot g^2 + 2 \cdot \sigma_{\text{ADC}}^2$$

$$L_1 - L_2 \sim \text{Skellam}\left(t \cdot a \cdot (\Phi_1 - \Phi_2), t \cdot (a \cdot (\Phi_1 + \Phi_2) + 2 \cdot D)\right)$$

Analog subtraction on sensor

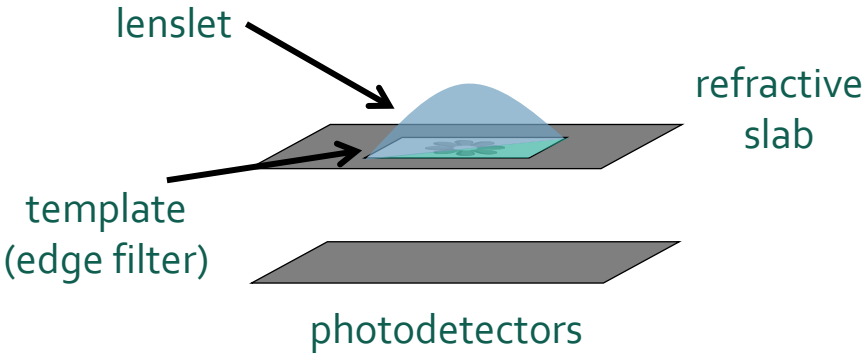
$$\sigma(I)^2 = \boxed{\sigma(L_1 - L_2)^2} + \sigma_{\text{opamp}}^2 \cdot g^2 + \sigma_{\text{read}}^2 \cdot g^2 + \sigma_{\text{ADC}}^2$$



$$L_1 \sim \text{Poisson}(t \cdot (a \cdot \Phi_1 + D))$$
$$L_2 \sim \text{Poisson}(t \cdot (a \cdot \Phi_2 + D))$$
$$n_{\text{opamp}} \sim \text{Normal}(0, \sigma_{\text{opamp}})$$
$$n_{\text{read}} \sim \text{Normal}(0, \sigma_{\text{read}})$$
$$n_{\text{ADC}} \sim \text{Normal}(0, \sigma_{\text{ADC}})$$

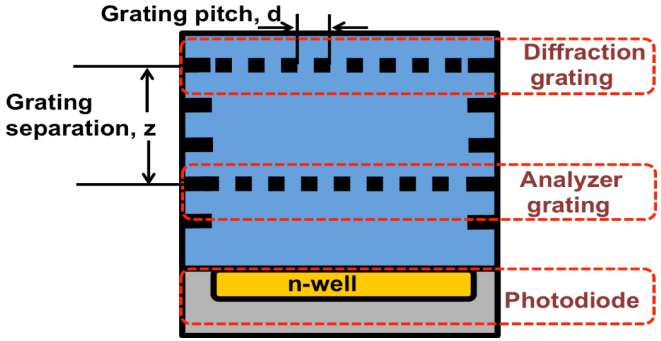
Change the Optics

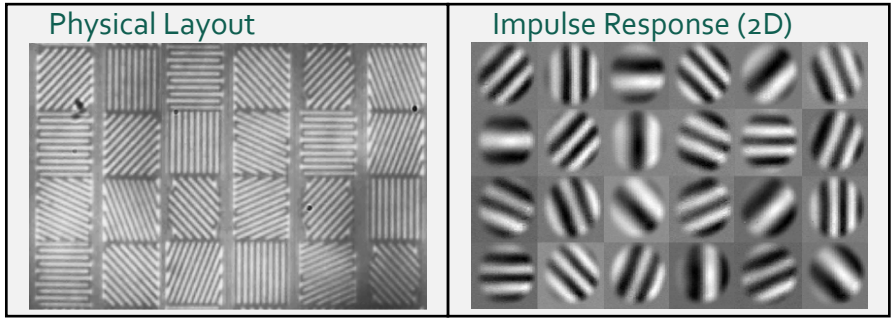
Optical filtering



resulting image

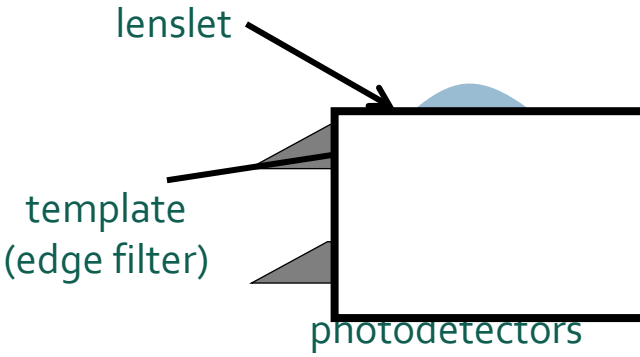
Angle-sensitive pixels





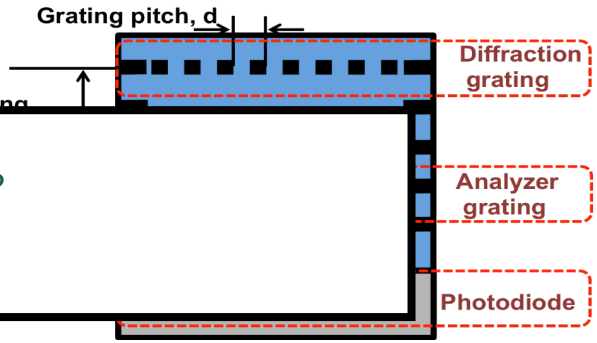
Change the Optics

Optical filtering

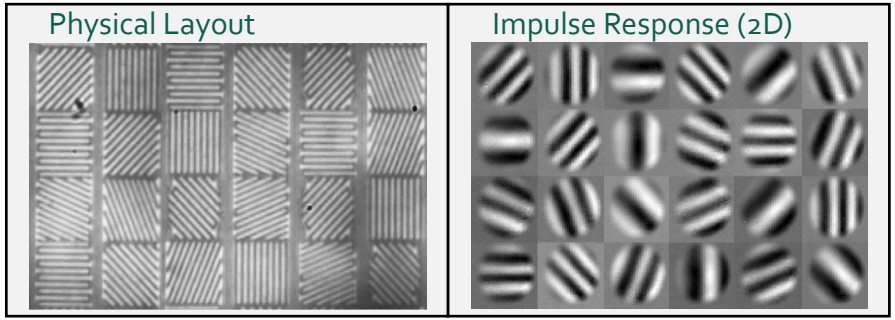


resulting image

Angle-sensitive pixels

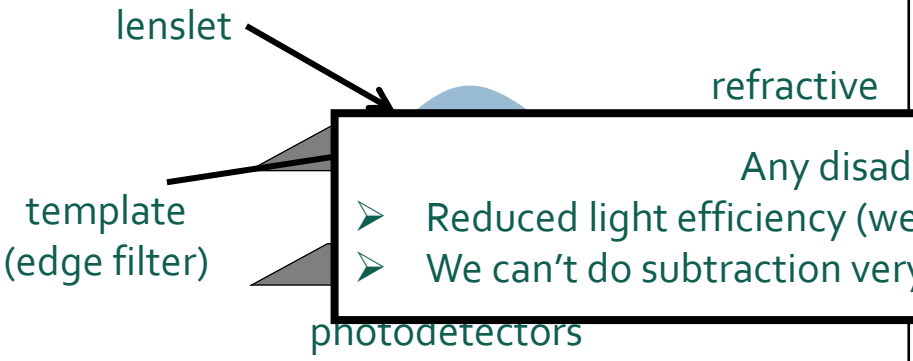


Any disadvantages?



Change the Optics

Optical filtering

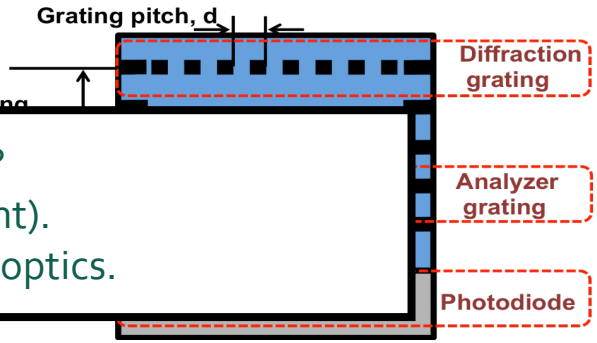


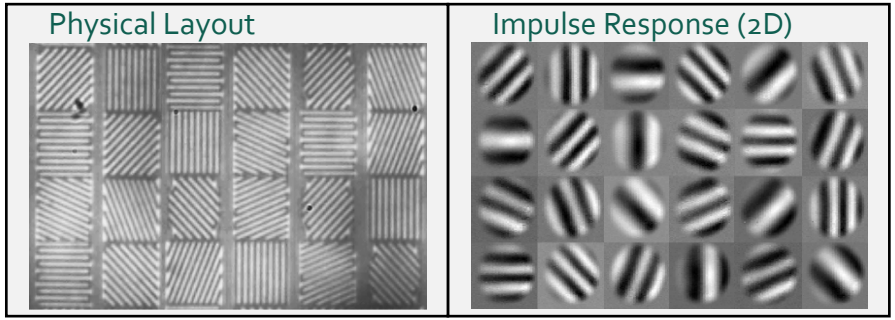
Any disadvantages?

- Reduced light efficiency (we block light).
- We can't do subtraction very easily in optics.

resulting image

Angle-sensitive pixels





Why Gradient Cameras?

Why I want a Gradient Camera

Jack Tumblin
Northwestern University
jet@cs.northwestern.edu

Amit Agrawal
University of Maryland
aagrawal@umd.edu

Ramesh Raskar
MERL
raskar@merl.com

Why a gradient camera?

- Faster frame rate, only very few pixels (gradient).
- Higher dynamic range, if also combined with logarithmic gradients.

Can you directly display the measurements of such a camera?

- Use a Poisson solver to reconstruct the image from the measured gradients.

How to build a gradient camera?

- Change the sensor.
- Change the optics.

Temporal Gradients

Event-based camera:

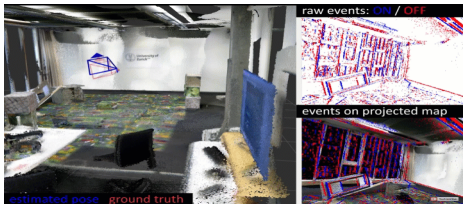
https://www.youtube.com/watch?v=kPCZESVfHoQ

High-speed output on a quadcopter:

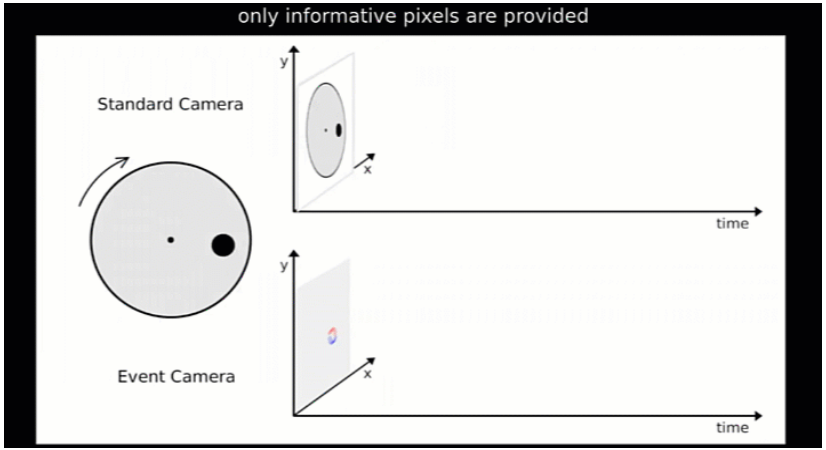
https://www.youtube.com/watch?v=LauQ6LWTkxM

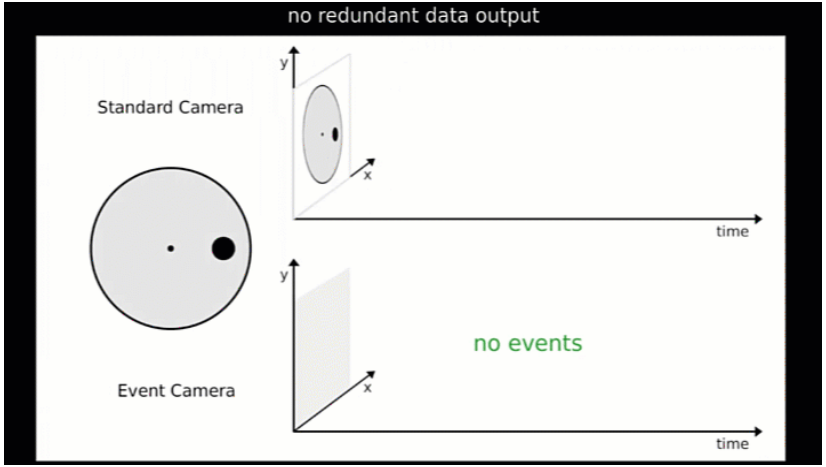
Simulator:

http://rpg.ifi.uzh.ch/esim

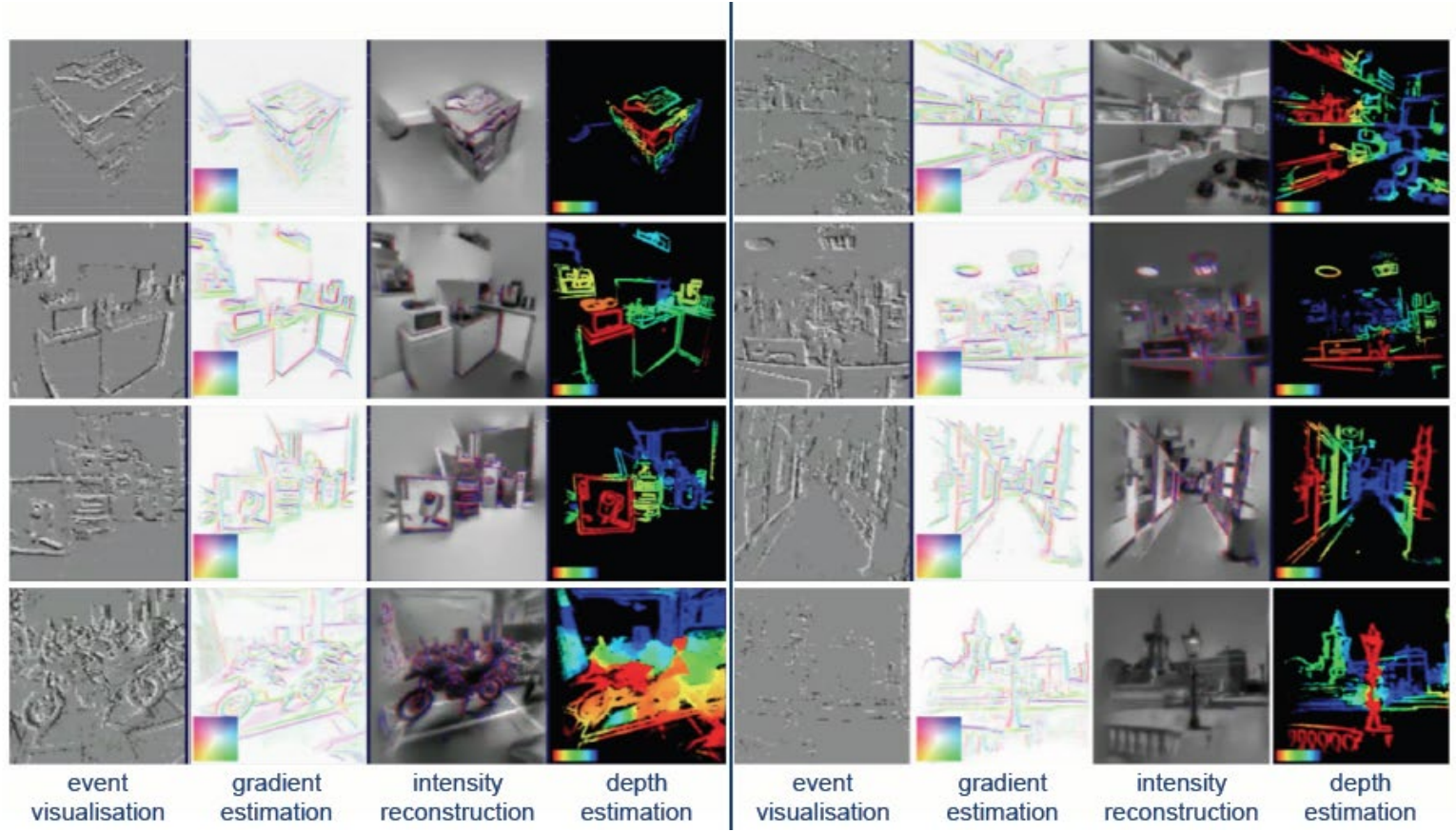


event-based cameras (a.k.a. dynamic vision sensors, or DVS)





Event Camera in Robotics and Cision



References

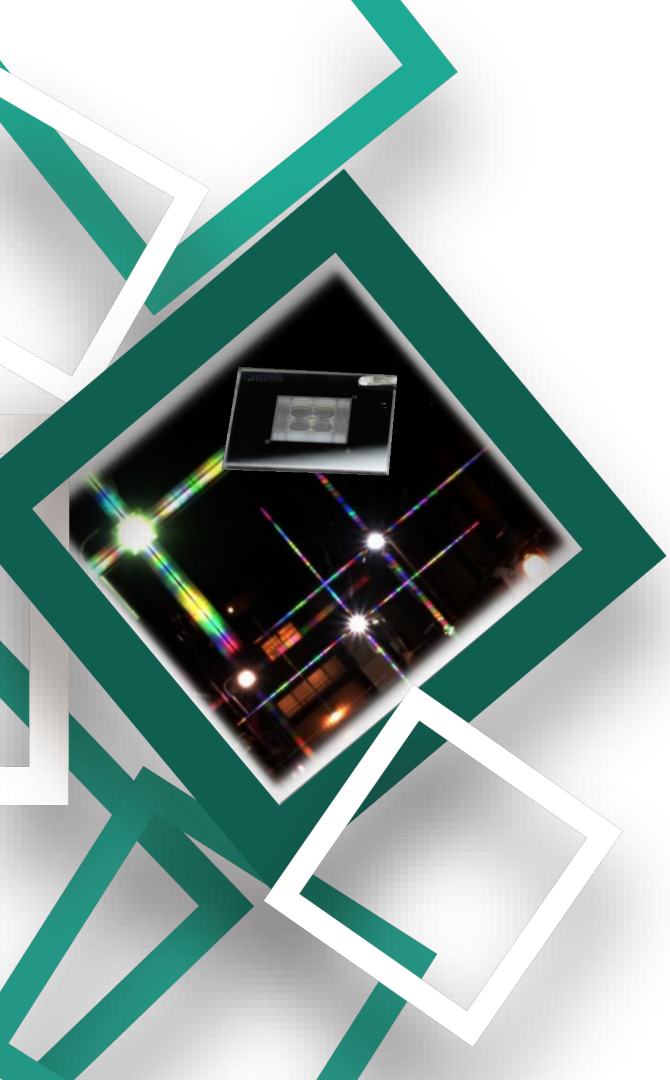
Basic reading:

- 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.
- Agrawal et al., "Removing Photography Artifacts Using Gradient Projection and Flash-Exposure Sampling," SIGGRAPH 2005.
  A 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.
- Frankot and Chellappa, "A method for enforcing integrability in shape from shading algorithms," PAMI 1988.
- Bhat et al., "Fourier Analysis of the 2D Screened Poisson Equation for Gradient Domain Problems," ECCV 2008.
  A couple of papers discussing the (Fourier) basis projection approach for solving the Poisson integration problem.
- Agrawal et al., "What Is the Range of Surface Reconstructions from a Gradient Field?," ECCV 2006.
- Quéau et al., "Normal Integration: A Survey," JMIV 2017.
  Two papers reviewing various gradient (and surface normal) integration techniques, including Poisson solvers.
- Szeliski, "Locally adapted hierarchical basis preconditioning," SIGGRAPH 2006.
- Krishnan and Szeliski, "Multigrid and multilevel preconditioners for computational photography," SIGGRAPH 2011.
- Krishnan et al., "Efficient Preconditioning of Laplacian Matrices for Computer Graphics," SIGGRAPH 2013.
  A few well-known references on multi-grid and preconditioning techniques for accelerating the Poisson solver, with a specific focus on computational photography applications..
- Shewchuk, "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain," CMU TR 1994, http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf
  A great reference on (preconditioned) conjugate gradient solvers for large linear systems.
- Briggs et al., "A multigrid tutorial," SIAM 2000.
  A great reference book on multi-grid approaches.
- 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.
- Krishnan and Fergus, "Dark Flash Photography," SIGGRAPH 2009.
  A paper proposing doing flash/no-flash photography using infrared flash lights.
- Kazhdan et al., "Poisson surface reconstruction," SGP 2006.
- Kazhdan and Hoppe, "Screened Poisson surface reconstruction," TOG 2013.
  Two papers discussing Poisson problems for reconstructing meshes from point clouds and normals. This is arguably the most commonly used surface reconstruction algorithm.
- Lehtinen et al., "Gradient-domain metropolis light transport," SIGGRAPH 2013.
- Kettunen et al., "Gradient-domain path tracing," SIGGRAPH 2015.
- Hua et al., "Light transport simulation in the gradient domain," SIGGRAPH Asia 2018 course, http://beltegeuse.s3-website-ap-northeast-1.amazonaws.com/research/2018_GradientCourse/
  In addition to *editing* images in the gradient-domain, we can *render* them directly 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*.
- Callenberg et al., "Snapshot difference imaging using correlation time-of-flight sensors," SIGGRAPH Asia 2017.
  A form of camera with differential pixels.
- Koppal et al., "Toward wide-angle microvision sensors", PAMI 2013.
  Gradient cameras using optical filtering.
- Chen et al., "ASP vision: Optically computing the first layer of convolutional neural networks using angle sensitive pixels," CVPR 2016.
  Gradient cameras using angle-sensitive pixels.
- Kim et al., "Real-time 3D reconstruction and 6-DoF tracking with an event camera," ECCV 2016.
  A paper on using evet-based cameras for computer vision applications in very fast frame rates (best paper award at ECCV 2016!).

- Efficient Poisson Solver
- Poisson Image Editing
- Flash/no-flash Photography
- Gradient Cameras
  - Diff Sensor
  - Diff Optics
  - Diff Time



Thank You!

Qilin Sun（孙启霖）

School of Data Science

The Chinese University of Hong Kong, Shenzhen