Deconvolution

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

15-463, 15-663, 15-862
Computational Photography
Fall 2020, Lecture 14

Course announcements

- Homework assignment 4 due November 2nd.
  - Generally shorter to accommodate final project proposals.
  - Two bonus parts.

- Project logistics on Piazza and the course website.
  - Project ideas due on Piazza on October 23rd (optional).
  - Project proposals due on Gradescope on October 30th.

- Office hour logistics for this week:
  - Yannis will have extra office hours on Friday (time TBD).

- Late submissions:
  - We are making an exception for homework assignment 3 and we won’t count late days for submissions that are a few minutes late due to uploading delays.
  - We will resume enforcing late days strictly for subsequent homeworks. One second late is one late day.

Computational photography talks this week @ CMU

- Ce Liu, Google Research (October 20th, 11 am – noon).

  - Title: Advancing the State of the Art of Computer Vision for Billions of Users

At Google, advancing the state of the art of computer vision is very impactful as there are billions of users of Google products, many of which require **high-quality, artifact-free images**. I will share what we learned from successfully launching core computer vision techniques for various Google products, including **PhotoScan (Photos), seamless Google Street View panorama stitching (Geo), Super Res Zoom (Pixel 4), Auto Pop-out & Uncrop (Display Ads)**, and Rendering4AI (Cloud AI). We also conduct academic research and publish at top-tier conferences. I will give an overview of several representative works, including **seeing through obstructions (Siggraph'15)**, learning the depth of moving people by watching frozen people (CVPR'19), GAN-based image uncrop (ICCV'19), and supervised contrastive learning (NeurIPS'20).

- Tali Dekel, Google Research (October 20th, noon – 1 pm).

  - Title: Learning to Retime People in Videos

By changing the speed of frames, or the speed of objects, we can enhance the way we perceive events or actions in videos. In this talk, I will present two of my recent works on retiming videos, and more specifically, manipulating the timings of people's actions. 1) "SpeedNet" (CVPR 2020 oral): a method for adaptively speeding up videos based on their content, allowing us to gracefully watch videos faster while avoiding jerky and unnatural motions. 2) "Layered Neural Rendering for Retiming People" (SIGGRAPH Asia): a method for speeding up, slowing down, or entirely freezing certain people in videos, while automatically re-rendering properly all the scene elements that are related to those people, like shadows, reflections, and loose clothing. Both methods are based on novel deep neural networks that learn concepts of natural motion and scene decomposition just by observing ordinary videos, without requiring any manual labels. I'll show adaptively sped-up videos of sports, of boring family events (that all of us want to watch faster), and I'll demonstrate various retiming effects of people dancing, groups running, and kids jumping on trampolines.

Overview of today’s lecture

- Sources of blur.
- Deconvolution.
- Blind deconvolution.

Slide credits

Most of these slides were adapted from:

- Fredo Durand (MIT).
- Gordon Wetzstein (Stanford).

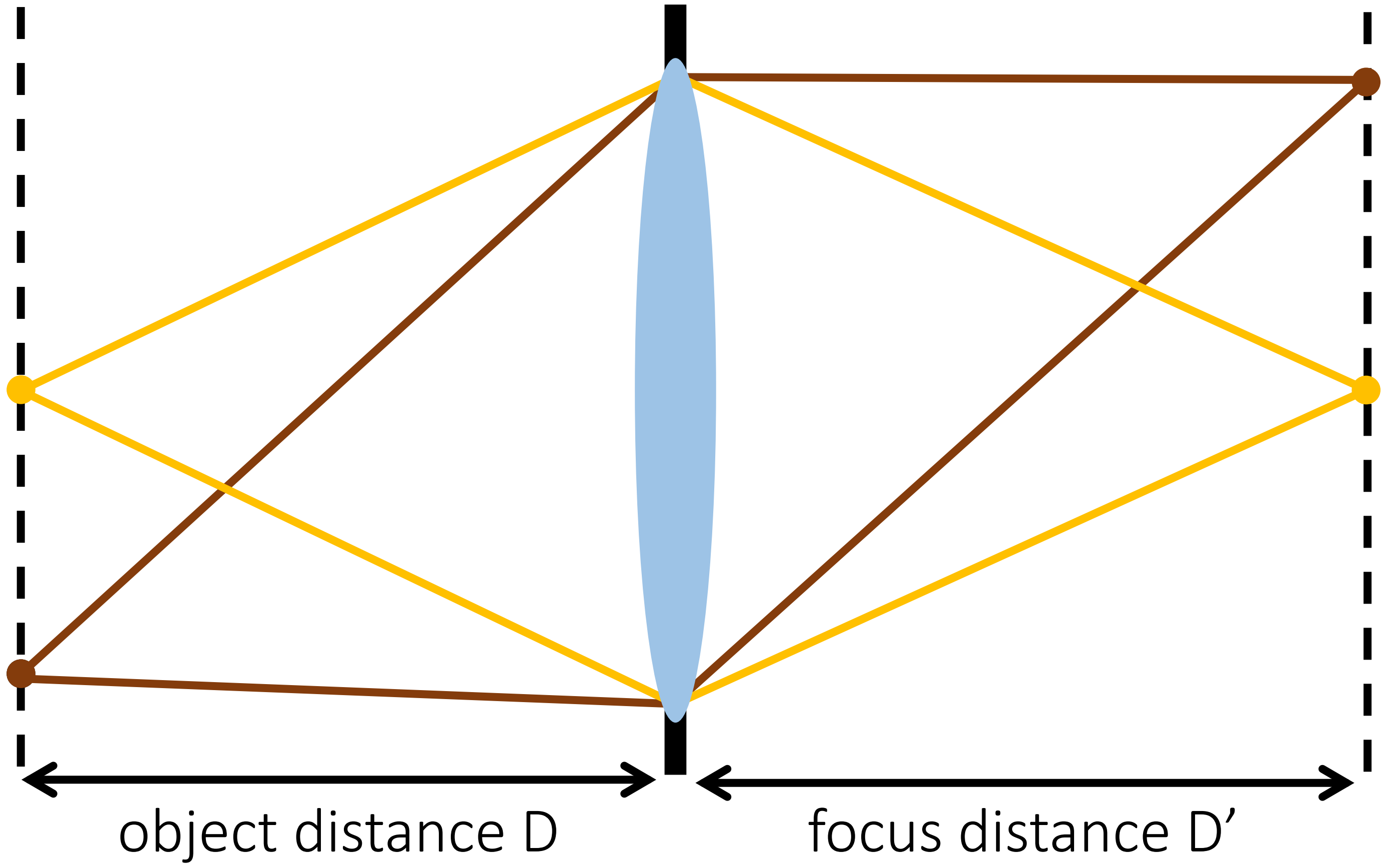
Why are our images blurry?

Why are our images blurry?

- Lens imperfections.
- Camera shake.
- Scene motion.
- Depth defocus.

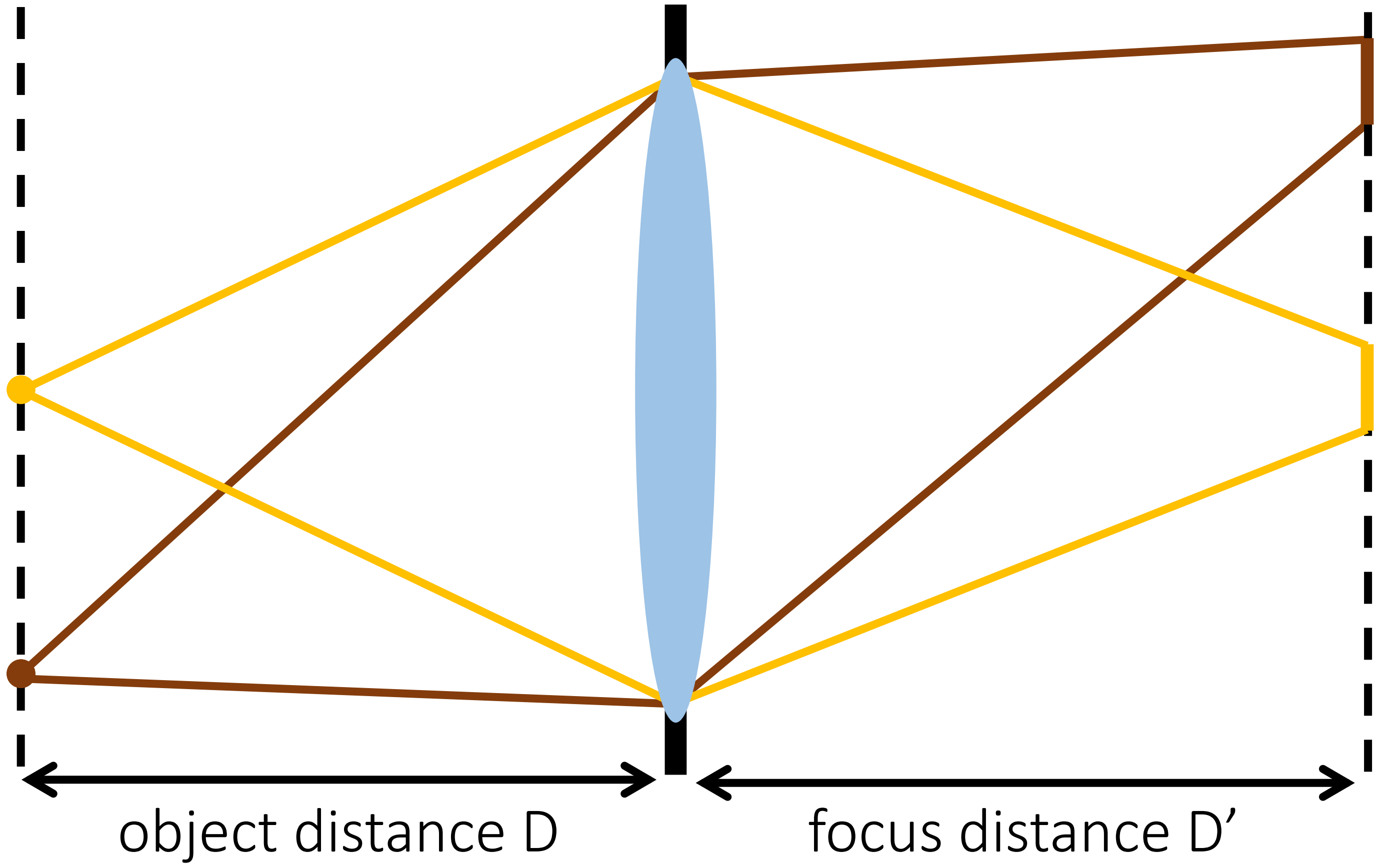
Lens imperfections

- Ideal lens: An point maps to a point at a certain plane.



Lens imperfections

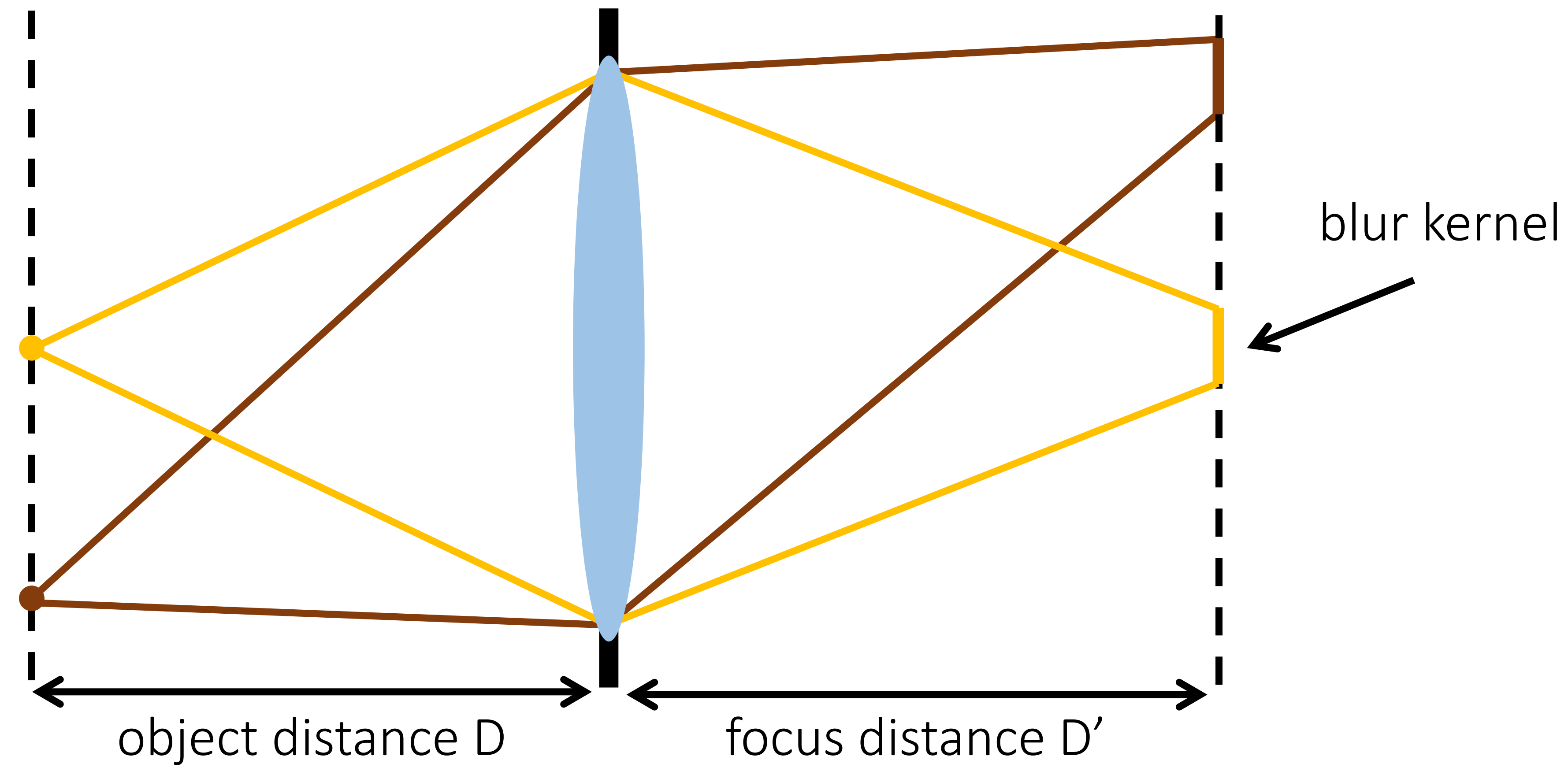
- Ideal lens: An point maps to a point at a certain plane.
- Real lens: A point maps to a circle that has non-zero minimum radius among all planes.



What is the effect of this on the images we capture?

Lens imperfections

- Ideal lens: An point maps to a point at a certain plane.
- Real lens: A point maps to a circle that has non-zero minimum radius among all planes.



Shift-invariant blur.

Lens imperfections

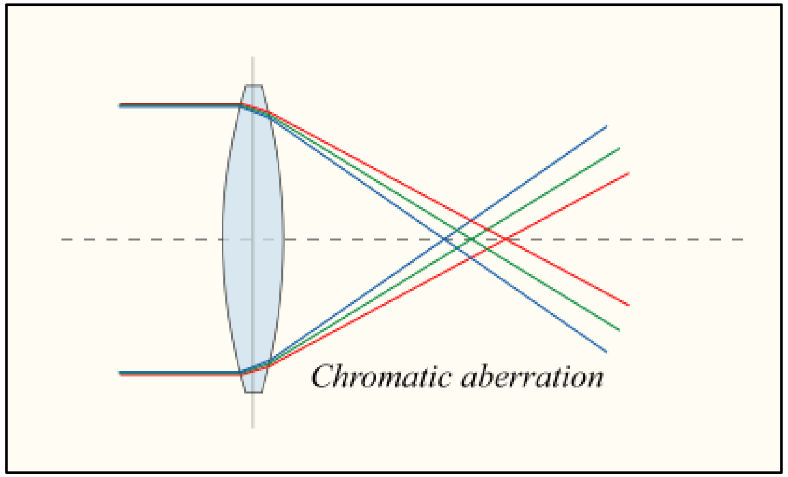
What causes lens imperfections?

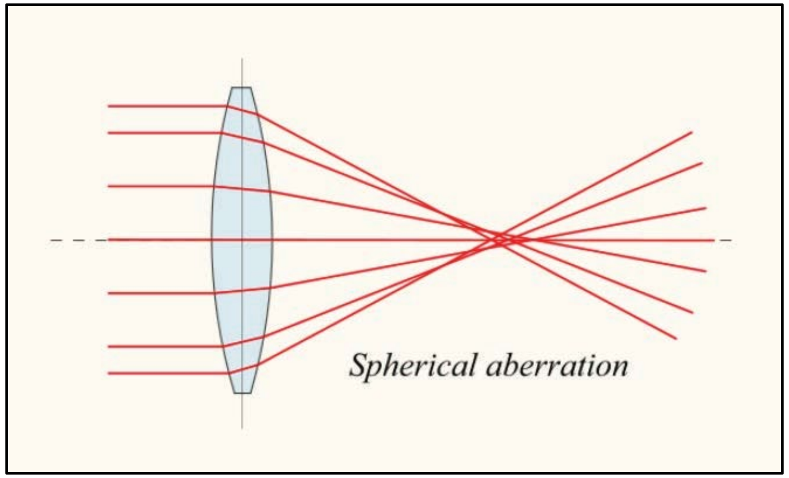
Lens imperfections

What causes lens imperfections?

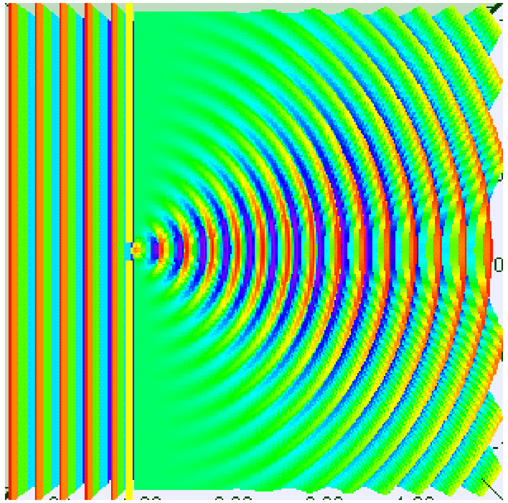
- Aberrations.

(Important note: Oblique aberrations like coma and distortion are not shift-invariant blur and we do not consider them here!)

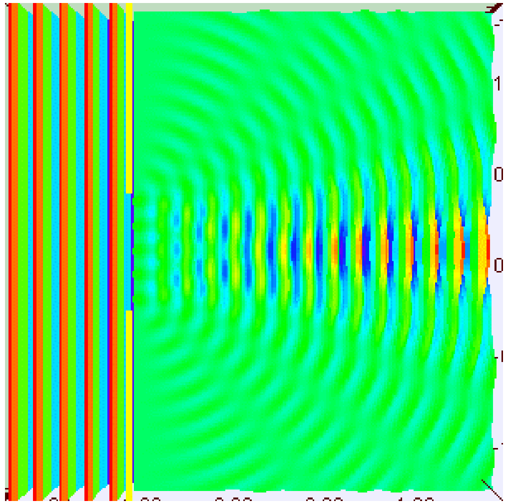




- Diffraction.



small aperture

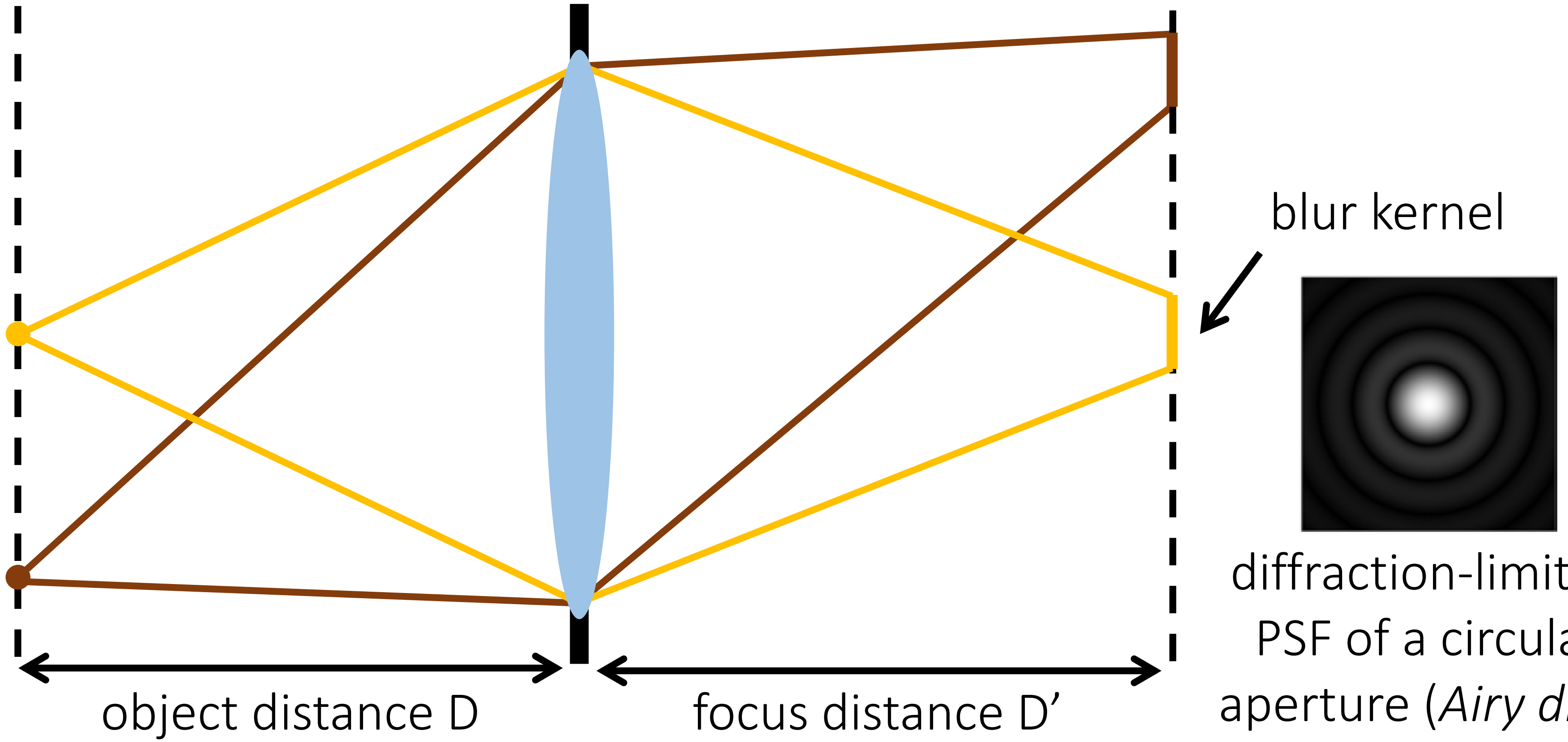


large aperture

Lens as an optical low-pass filter

Point spread function (PSF): The blur kernel of a lens.

- “Diffraction-limited” PSF: No aberrations, only diffraction. Determined by aperture shape.

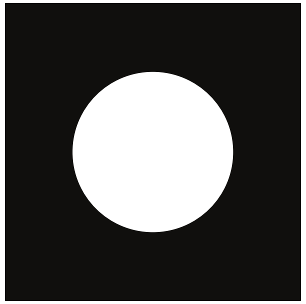


diffraction-limited PSF of a circular aperture (*Airy disk*)

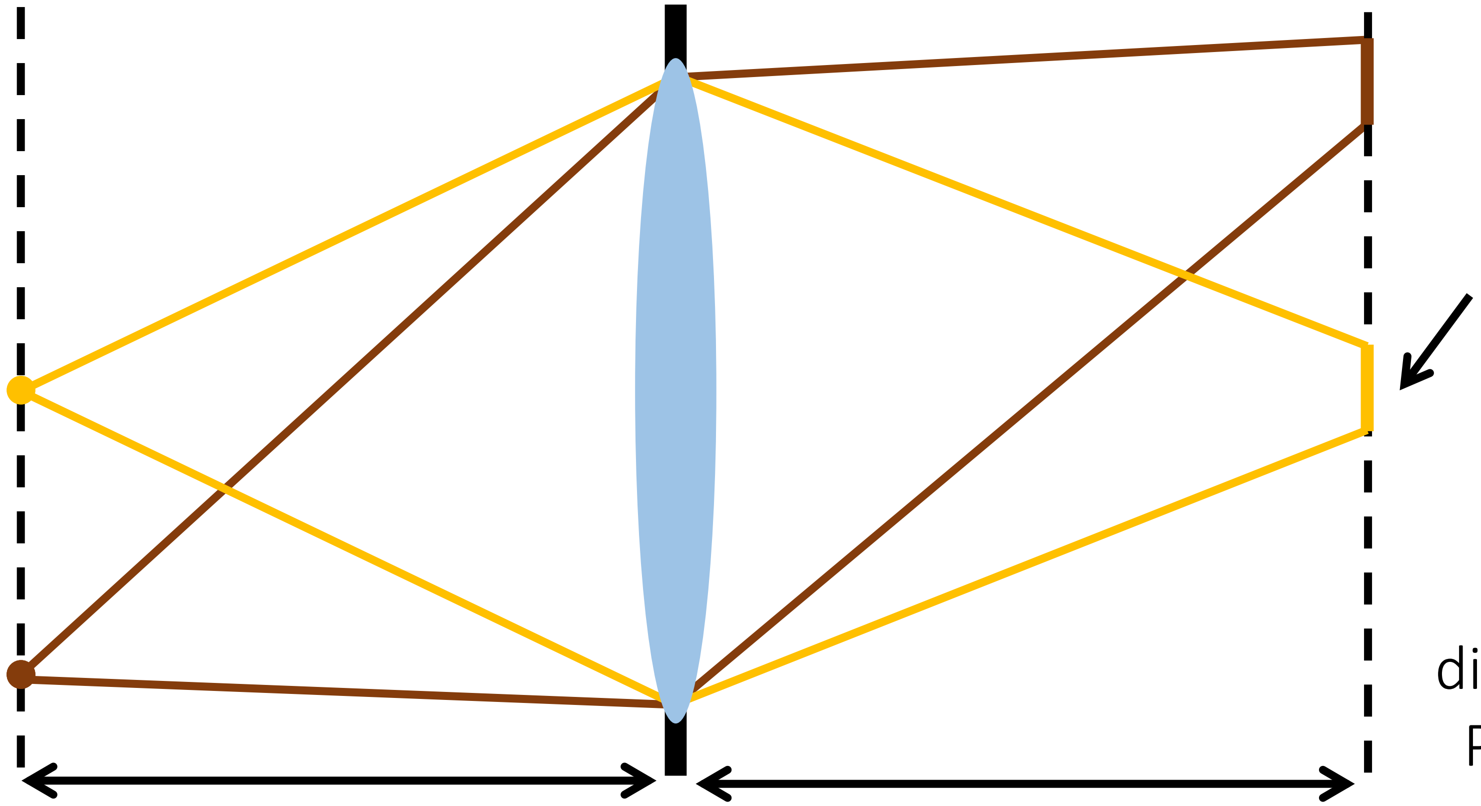
Lens as an optical low-pass filter

Point spread function (PSF): The blur kernel of a lens.

- "Diffraction-limited" PSF: No aberrations, only diffraction. Determined by aperture shape.



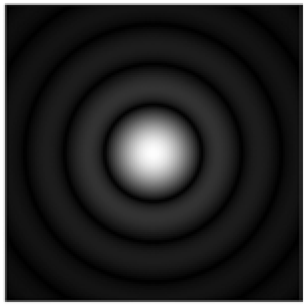
diffraction-limited OTF of a circular aperture



object distance D

focus distance D'

blur kernel



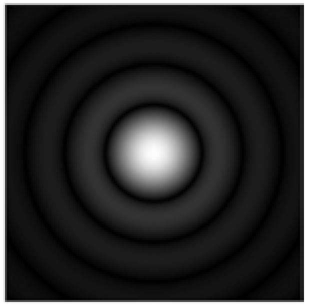
diffraction-limited PSF of a circular aperture

Optical transfer function (OTF): The Fourier transform of the PSF. Equal to aperture shape.

Lens as an optical low-pass filter

image from a perfect lens

*



imperfect lens PSF

=

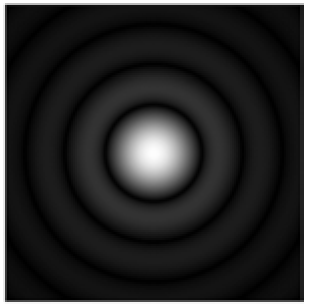
image from imperfect lens

$$x \quad * \quad c \quad = \quad b$$

Lens as an optical low-pass filter

If we know c and b, can we recover x?

$*$



$=$

image from a perfect lens

imperfect lens PSF

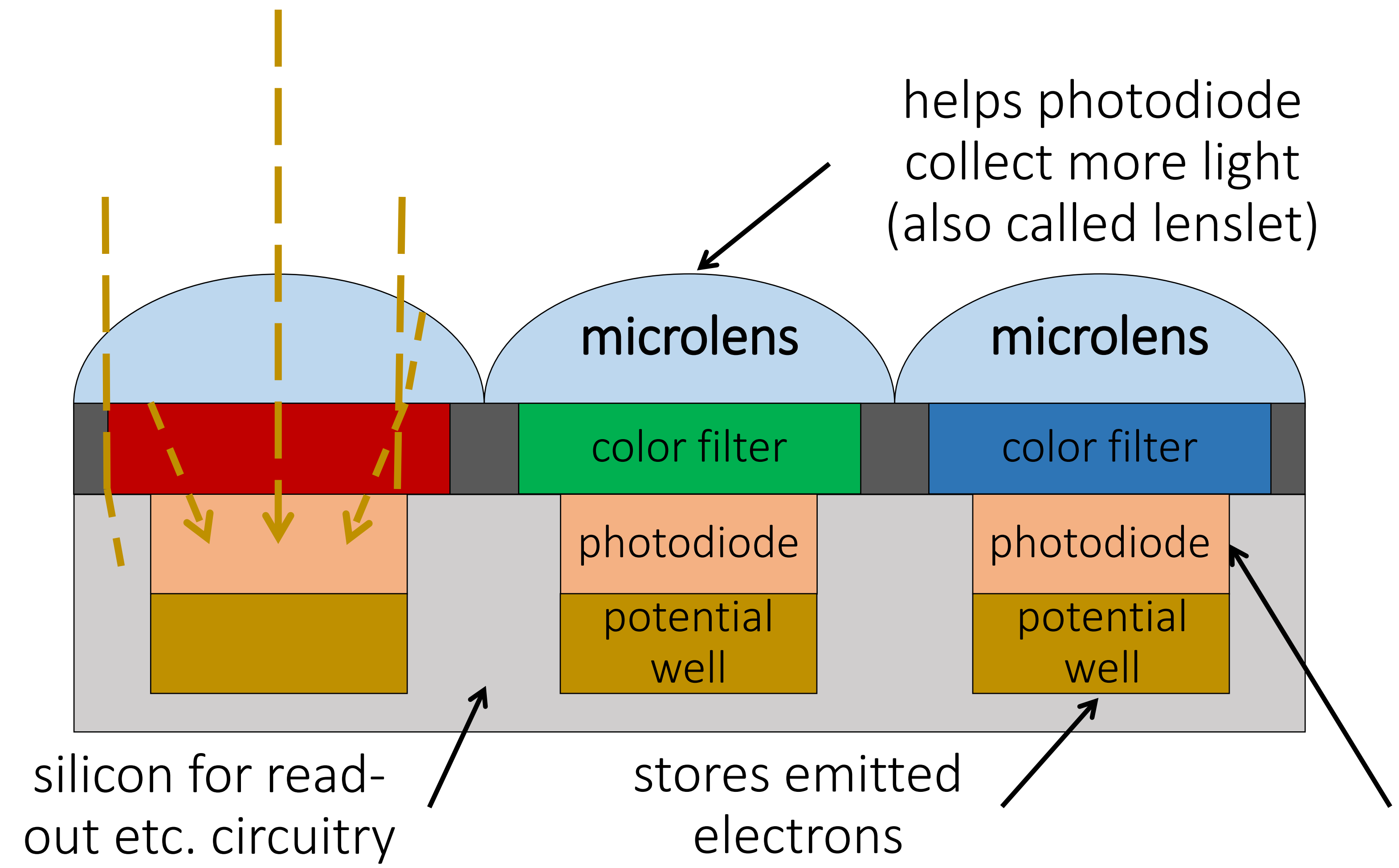
image from imperfect lens

$$x * c = b$$

Quick aside: optical anti-aliasing

Lenses act as (optical) low-pass filters.

Slide from lecture 2: Basic imaging sensor design



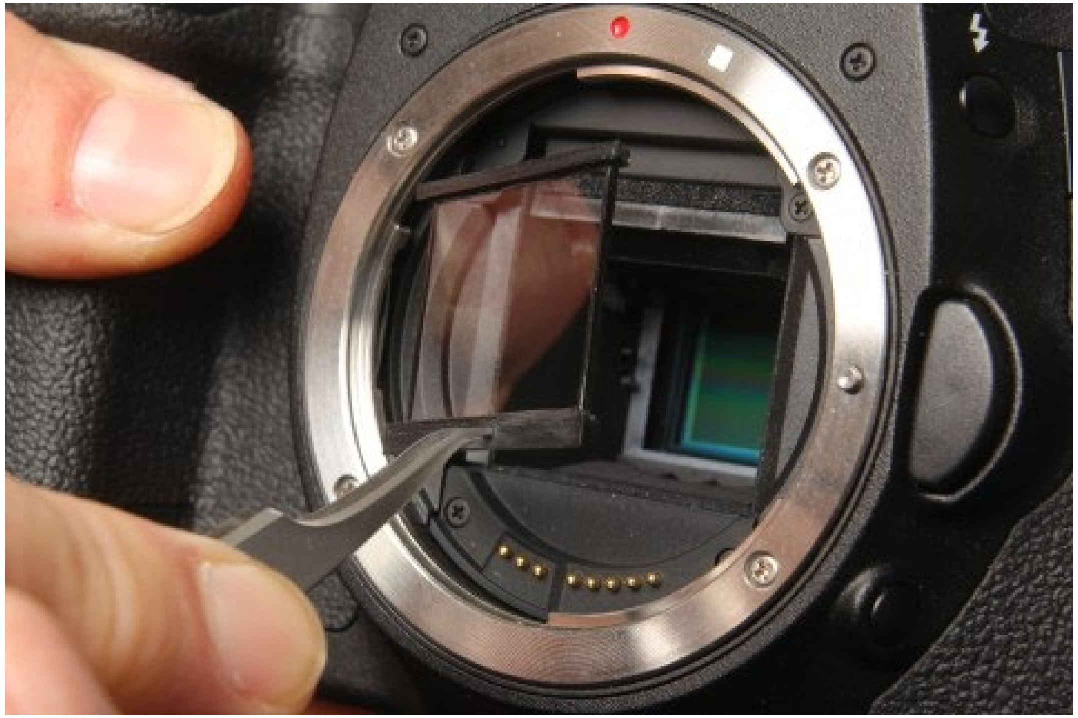
- **Lenslets also filter the image to avoid resolution artifacts.**
- Lenslets are problematic when working with coherent light.
- Many modern cameras do not have lenslet arrays.

We will discuss these issues in more detail at a later lecture.

Quick aside: optical anti-aliasing

Lenses act as (optical) smoothing filters.

- Sensors often have a lenslet array in front of them as an anti-aliasing (AA) filter.
- However, the AA filter means you also lose resolution.
- Nowadays, due the large number of sensor pixels, AA filters are becoming unnecessary.



Photographers often hack their cameras to remove the AA filter, in order to avoid the loss of resolution.

a.k.a. “hot rodding”

Quick aside: optical anti-aliasing

Example where AA filter is needed

without AA filter

with AA filter

Quick aside: optical anti-aliasing

Example where AA filter is unnecessary

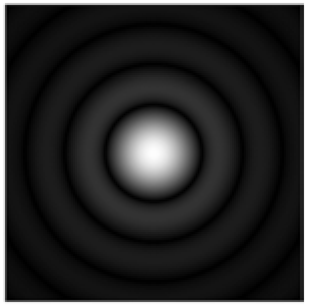
without AA filter

with AA filter

Lens as an optical low-pass filter

If we know c and b, can we recover x?

*



=

image from a perfect lens | imperfect lens PSF | image from imperfect lens

$$x \ast c = b$$

Deconvolution

$$x \quad * \quad c \quad = \quad b$$

If we know c and b, can we recover x?

Deconvolution

$$x \quad * \quad c \quad = \quad b$$

Reminder: convolution is multiplication in Fourier domain:

$$F(x) \quad \cdot \quad F(c) \quad = \quad F(b)$$

If we know c and b, can we recover x?

Deconvolution

$$x \quad * \quad c \quad = \quad b$$

Reminder: convolution is multiplication in Fourier domain:

$$F(x) \cdot F(c) = F(b)$$

Deconvolution is division in Fourier domain:

$$F(x_{est}) = F(b) \setminus F(c)$$

After division, just do inverse Fourier transform:

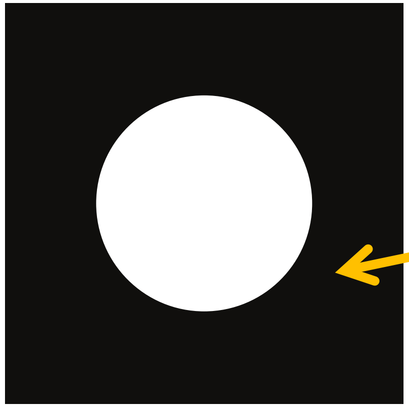
$$x_{est} = F^{-1}\left(F(b) \setminus F(c) \right)$$

Deconvolution

Any problems with this approach?

Deconvolution

- The OTF (Fourier of PSF) is a low-pass filter



zeros at high frequencies

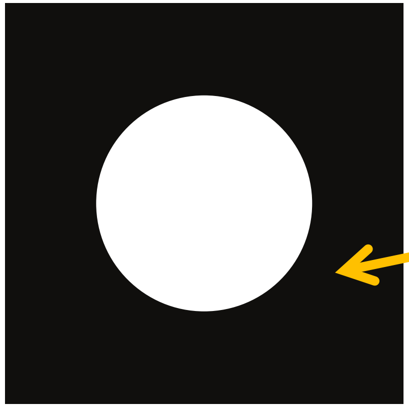
- The measured signal includes noise

$$b = c * x + n$$

noise term

Deconvolution

- The OTF (Fourier of PSF) is a low-pass filter



zeros at high frequencies

- The measured signal includes noise

$$b = c * x + n$$

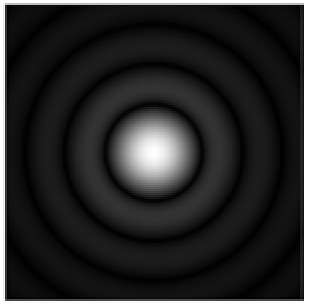
noise term

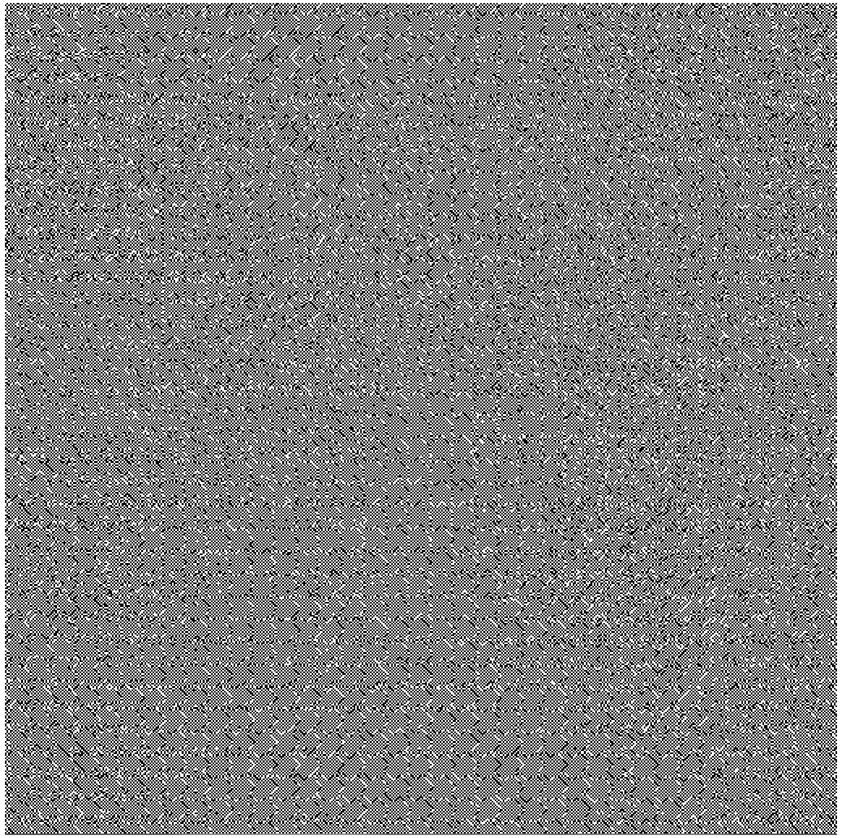
- When we divide by zero, we amplify the high frequency noise

Naïve deconvolution

Even tiny noise can make the results awful.

- Example for Gaussian of σ = 0.05

$*$ () $^{-1}$ $=$



$$b \ast c^{-1} = x_{est}$$

Wiener Deconvolution

Apply inverse kernel and do not divide by zero:

$$x_{est} = F^{-1}\left(\boxed{\frac{|F(c)|^2}{|F(c)|^2 + 1/SNR(\omega)}} \cdot \frac{F(b)}{F(c)} \right)$$

noise-dependent damping factor

- Derived as solution to maximum-likelihood problem under Gaussian noise assumption
- Requires noise of signal-to-noise ratio at each frequency

$$SNR(\omega) = \frac{\text{signal variance at } \omega}{\text{noise variance at } \omega}$$

Wiener Deconvolution

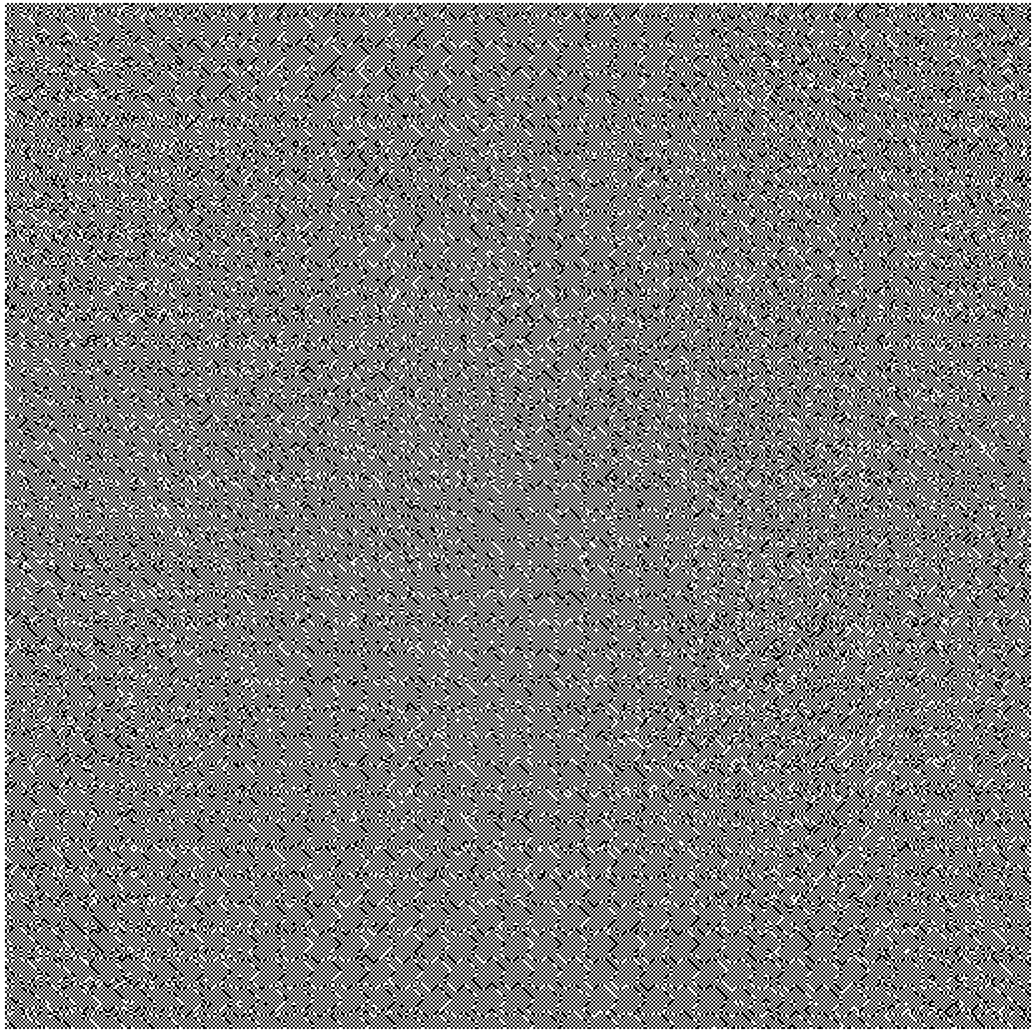
Apply inverse kernel and do not divide by zero:

$$x_{est} = F^{-1} \left(\underbrace{\boxed{\frac{|F(c)|^2}{|F(c)|^2 + 1/SNR(\omega)}}}_{\text{noise-dependent damping factor}} \cdot \frac{F(b)}{F(c)} \right)$$

Intuitively:

- When SNR is high (low or no noise), just divide by kernel.
- When SNR is low (high noise), just set to zero.

Deconvolution comparisons



naïve deconvolution

Wiener deconvolution

Deconvolution comparisons

σ = 0.01

σ = 0.05

σ = 0.01

Derivation

Sensing model:

$$\tilde{x} = c * x + n$$

Noise $\mathbf{n}$ is assumed to be zero-mean and independent of signal $\mathbf{x}$.

Derivation

Sensing model:

$$b = c * x + n$$

Noise $\mathbf{n}$ is assumed to be zero-mean and independent of signal $\mathbf{x}$.

Fourier transform:

$$B = C \cdot X + N$$

Why multiplication?

Derivation

Sensing model:

$$b = c * x + n$$

Noise $\mathbf{n}$ is assumed to be *zero-mean* and *independent* of signal $\mathbf{x}$.

Fourier transform:

$$B = C \cdot X + N$$

Convolution becomes multiplication.

Problem statement: Find function $\mathrm{H}(\omega)$ that minimizes *expected* error *in Fourier domain*.

$$\min_{H} E[\|X - HB\|^2]$$

Derivation

Replace B and re-arrange loss:

$$\min_H E[\|(1 + HC)X - HN\|^2]$$

Expand the squares:

$$\min_H \|1 - HC\|^2 E[\|X\|^2] - 2(1 - HC)E[XN] + \|H\|^2 E[\|N\|^2]$$

Derivation

When handling the cross terms:

- Can I write the following?

$$E[XN] = E[X]E[N]$$

Derivation

When handling the cross terms:

- Can I write the following?

$$E[XN] = E[X]E[N]$$

Yes, because X and N are assumed independent.

- What is this expectation product equal to?

Derivation

When handling the cross terms:

- Can I write the following?

$$E[XN] = E[X]E[N]$$

Yes, because X and N are assumed independent.

- What is this expectation product equal to?

Zero, because N has zero mean.

Derivation

Replace B and re-arrange loss:

$$\min_H E[\|(1+HC)X - HN\|^2]$$

Expand the squares:

$$\min_H \|1 - HC\|^2 E[\|X\|^2] - 2(1 - HC)E[XN] + \|H\|^2 E[\|N\|^2]$$

cross-term is zero

Simplify:

$$\min_H \|1 - HC\|^2 E[\|X\|^2] + \|H\|^2 E[\|N\|^2]$$

How do we solve this optimization problem?

Derivation

Differentiate loss with respect to H, set to zero, and solve for H:

$$\frac{\partial \mathrm{loss}}{\partial H} = 0$$

$$\Rightarrow -2(1 - HC)E[\|X\|^2] + 2HE[\|N\|^2] = 0$$

$$\Rightarrow H = \frac{CE[\|X\|^2]}{C^2E[\|X\|^2] + E[\|N\|^2]}$$

Divide both numerator and denominator with $E[\|X\|^2]$, extract factor $1/\mathrm{C}$, and done!

Wiener Deconvolution

Apply inverse kernel and do not divide by zero:

$$x_{est} = F^{-1}\left(\frac{|F(c)|^2}{|F(c)|^2 + 1/SNR(\omega)} \cdot \frac{F(b)}{F(c)}\right)$$

noise-dependent damping factor

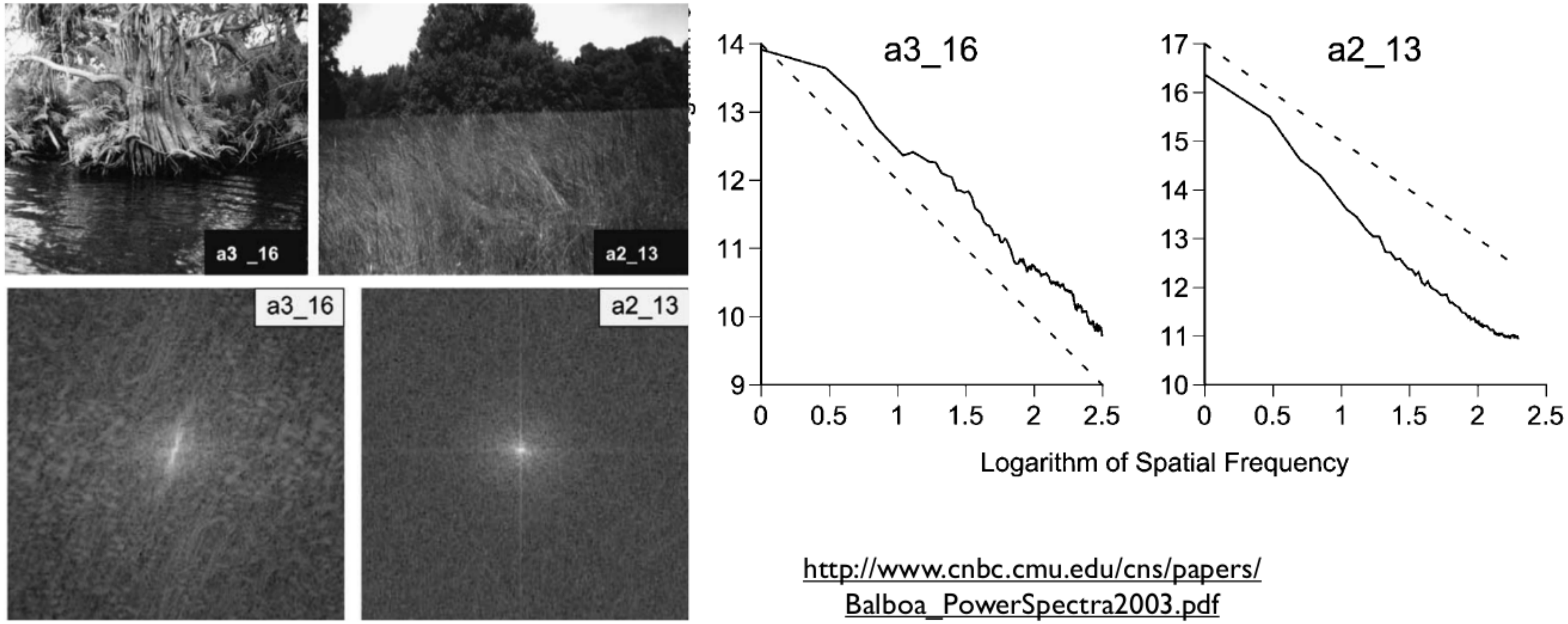
- Derived as solution to maximum-likelihood problem under Gaussian noise assumption
- Requires estimate of signal-to-noise ratio at each frequency

$$SNR(\omega) = \frac{\text{signal variance at } \omega}{\text{noise variance at } \omega}$$

Natural image and noise spectra

Natural images *tend* to have spectrum that scales as $1 / \omega^2$

- This is a *natural image statistic*

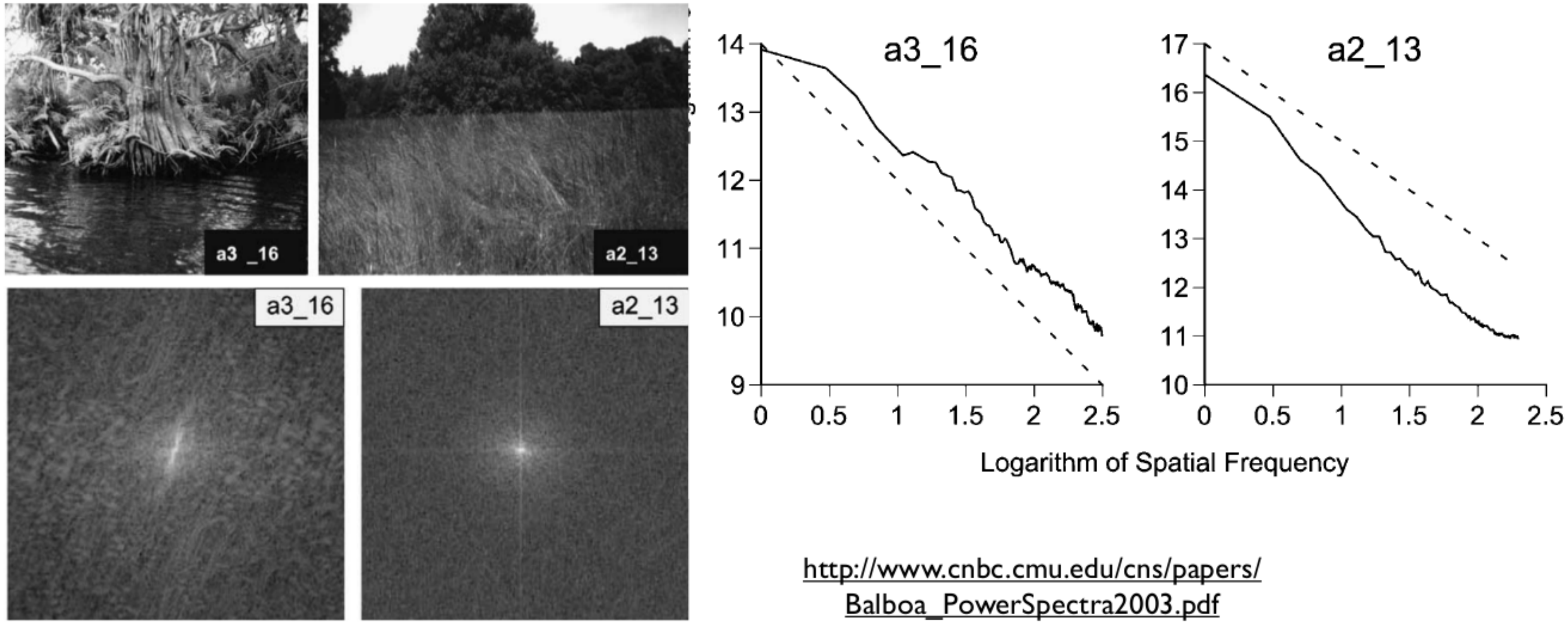


http://www.cnbc.cmu.edu/cns/papers/Balboa_PowerSpectra2003.pdf

Natural image and noise spectra

Natural images *tend* to have spectrum that scales as $1 / \omega^2$

- This is a *natural image statistic*



http://www.cnbc.cmu.edu/cns/papers/Balboa_PowerSpectra2003.pdf

Noise tends to have flat spectrum, $\sigma(\omega)$ = constant

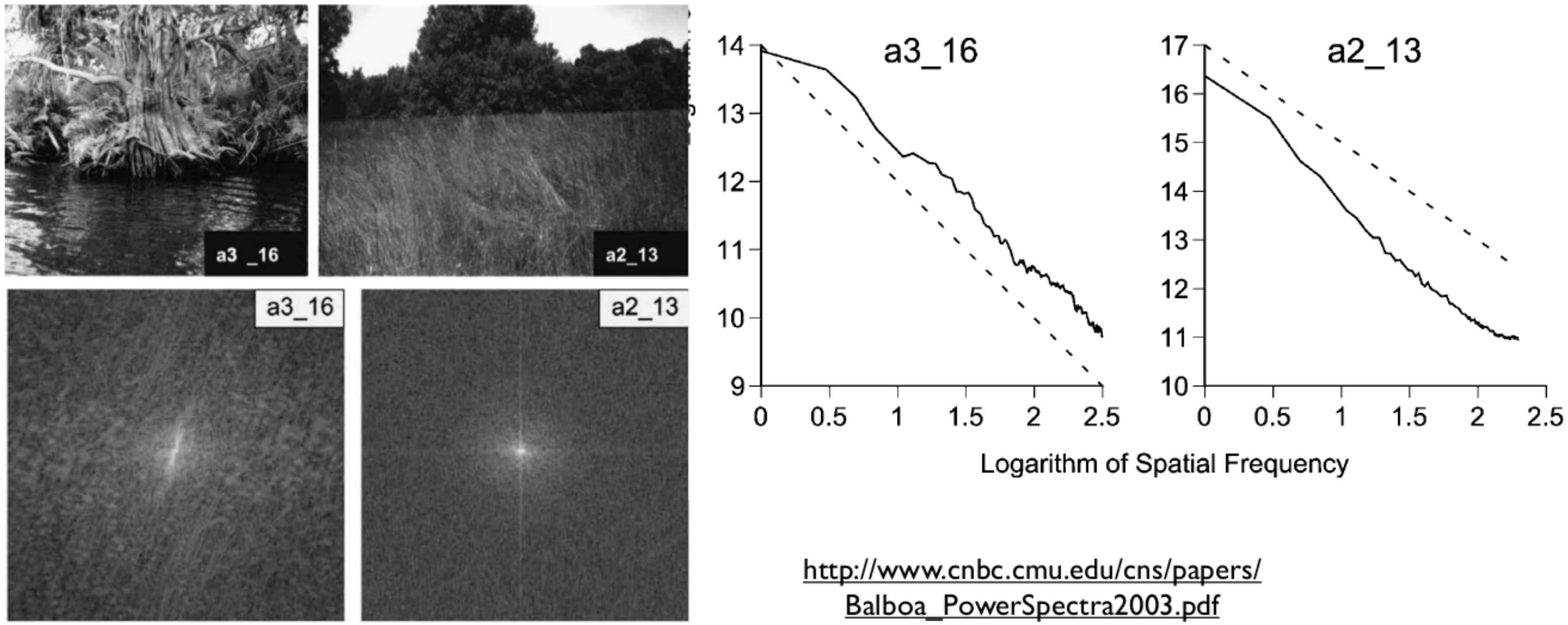
- We call this white noise

What is the SNR?

Natural image and noise spectra

Natural images *tend* to have spectrum that scales as $1 / \omega^2$

- This is a *natural image statistic*



http://www.cnbc.cmu.edu/cns/papers/Balboa_PowerSpectra2003.pdf

Noise tends to have flat spectrum, $\sigma(\omega)$ = constant

- We call this white noise

Therefore, we have that: $\mathrm{SNR}(\omega) = 1 / \omega^2$

Wiener Deconvolution

Apply inverse kernel and do not divide by zero:

$$x_{est} = F^{-1}\left(\frac{|F(c)|^2}{|F(c)|^2 + \omega^2} \cdot \frac{F(b)}{F(c)}\right)$$

amplitude-dependent damping factor

- Derived as solution to maximum-likelihood problem under Gaussian noise assumption
- $\boxed{\text{Requires noise of signal-to-noise ratio at each frequency}}$

$$\mathrm{SNR}(\omega) = \frac{1}{\omega^2}$$

Wiener Deconvolution

For natural images and white noise, equivalent to the minimization problem:

$$\min_x \|b - c * x\|^2 + \|\nabla x\|^2$$

gradient *regularization*

How can you prove this equivalence?

Wiener Deconvolution

For natural images and white noise, it can be re-written as the minimization problem

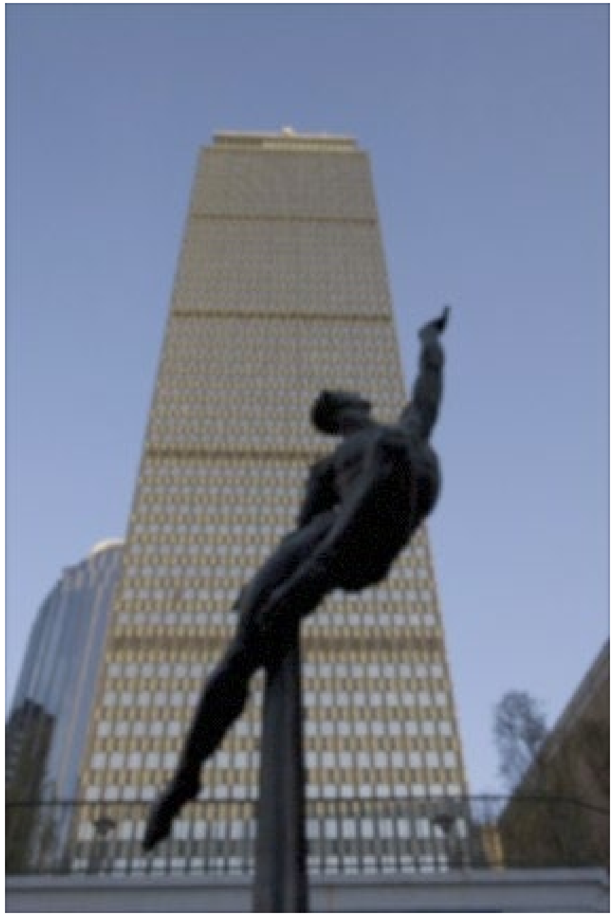
$$\min_x \|b - c * x\|^2 + \|\nabla x\|^2$$

gradient *regularization*

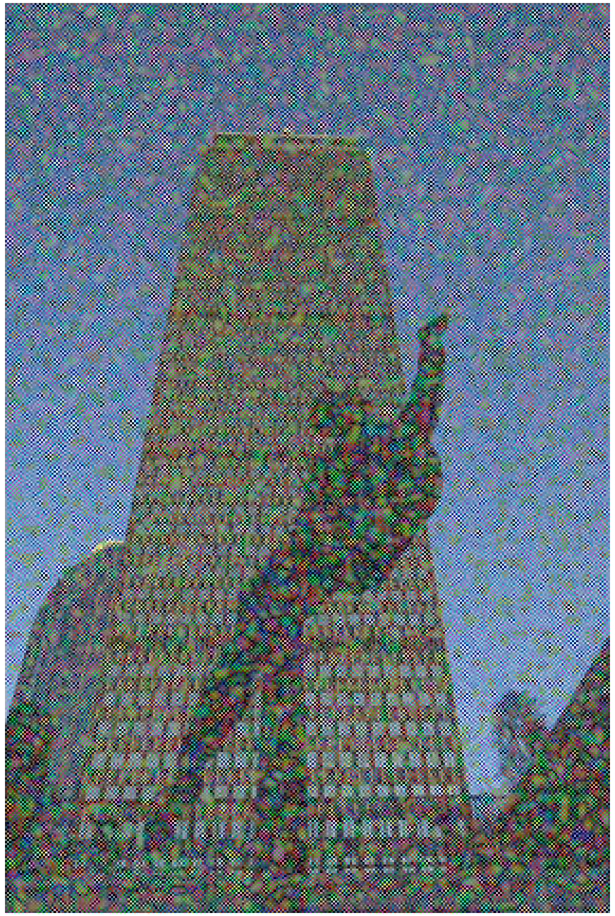
How can you prove this equivalence?

- Convert to Fourier domain and repeat the proof for Wiener deconvolution.
- Intuitively: The ω^2 term in the denominator of the special Wiener filter is the square of the Fourier transform of $\nabla \mathbf{x}$, which is $\mathbf{i} \cdot \omega$.

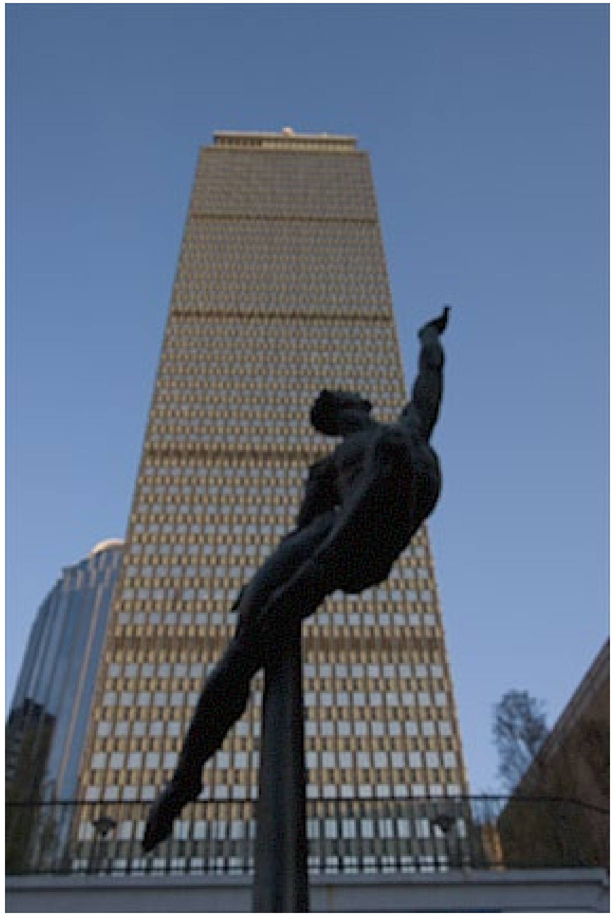
Deconvolution comparisons



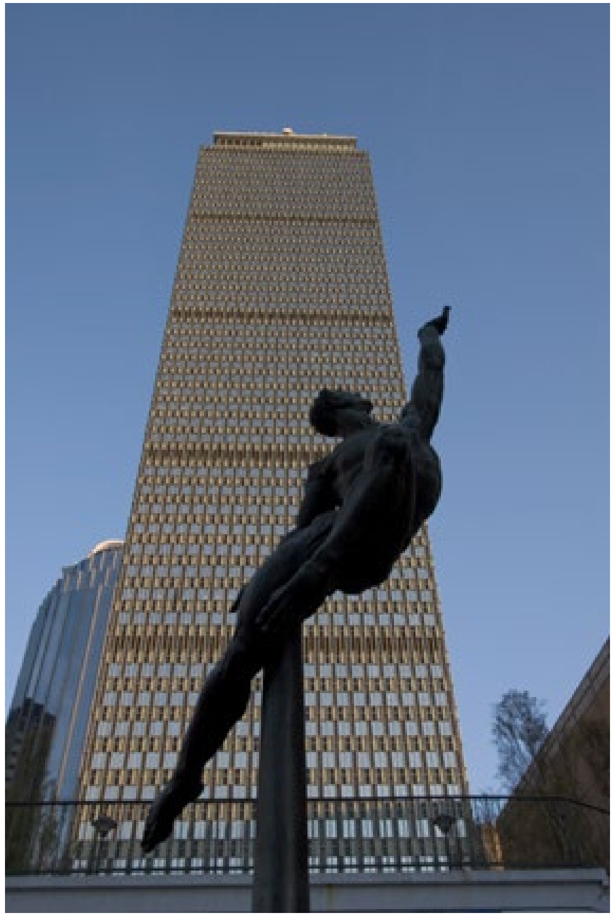
blurry input



naive deconvolution

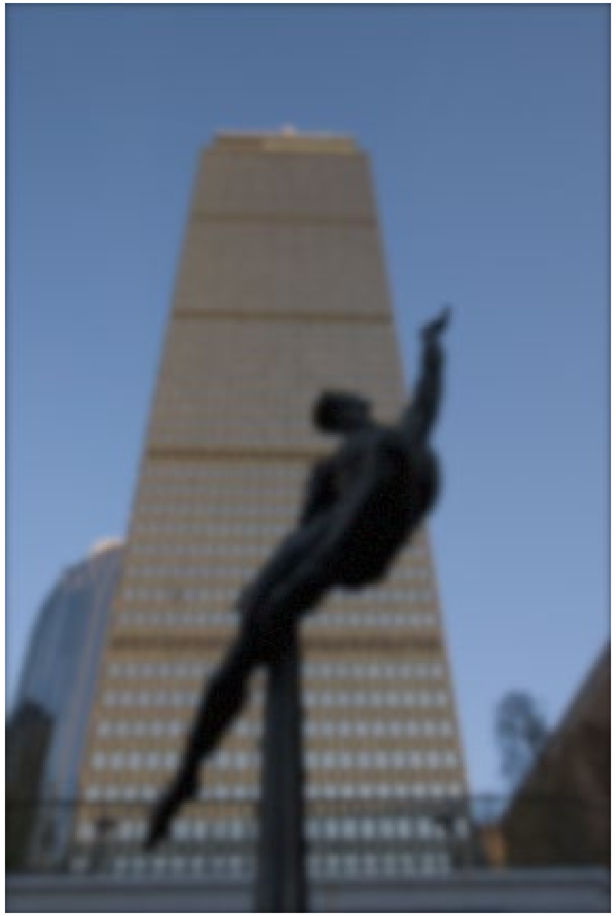


gradient regularization

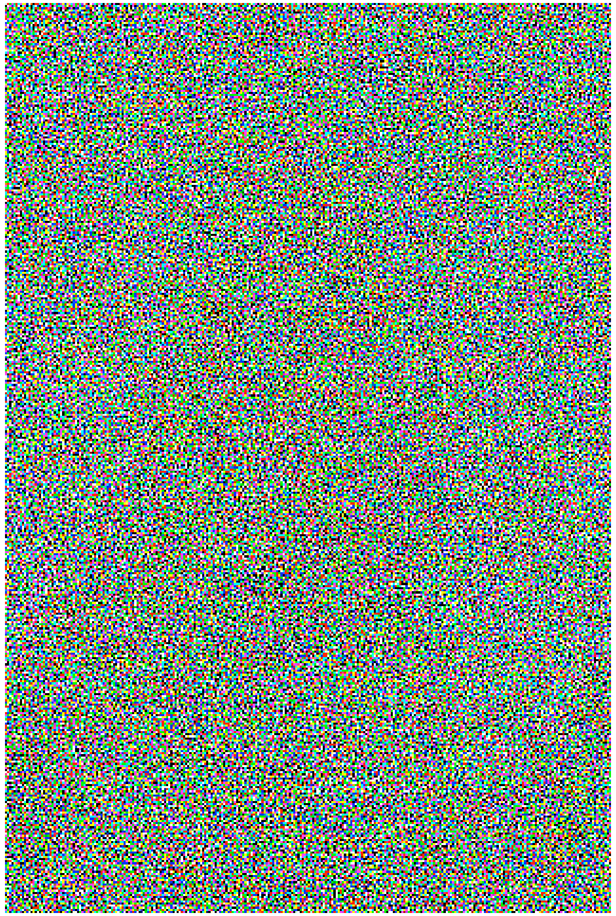


original

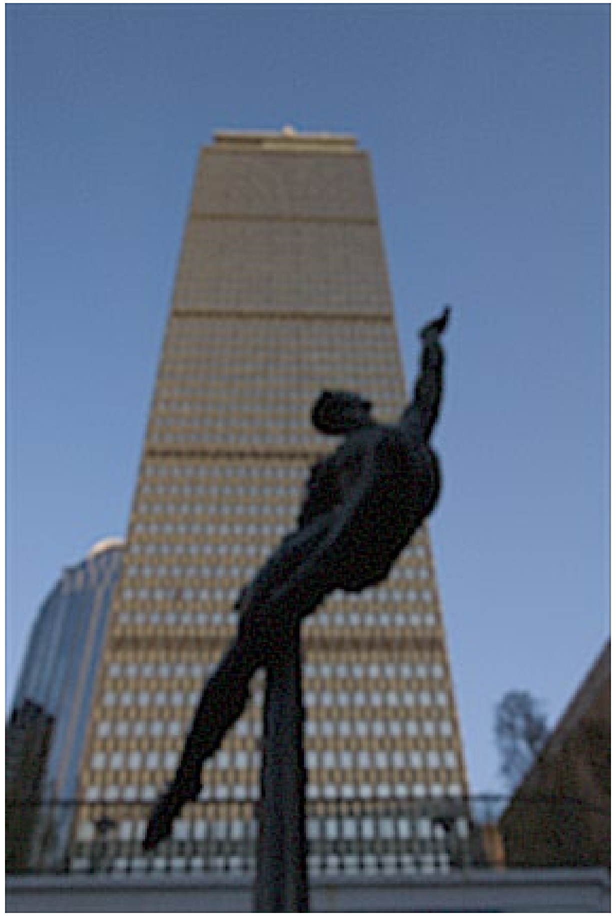
Deconvolution comparisons



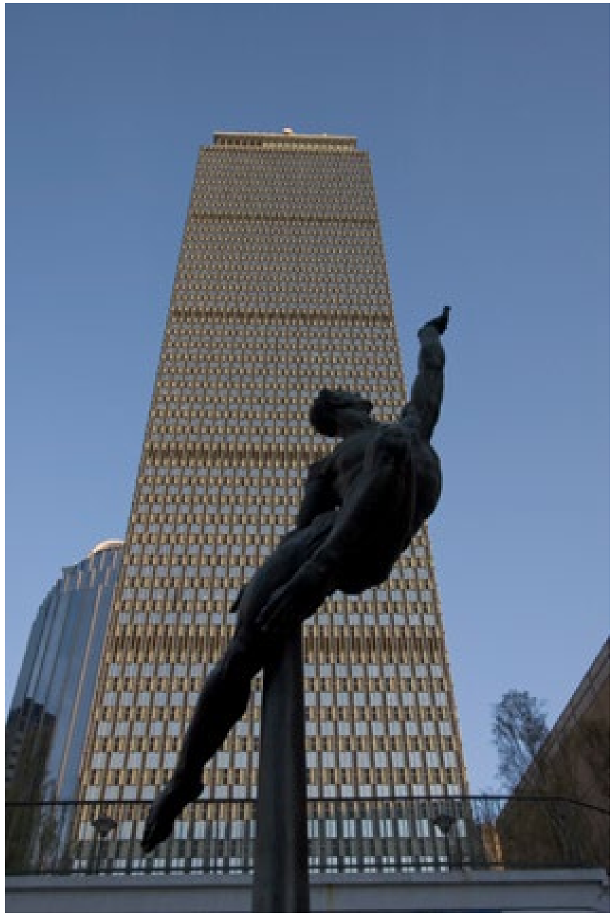
blurry input



naive deconvolution

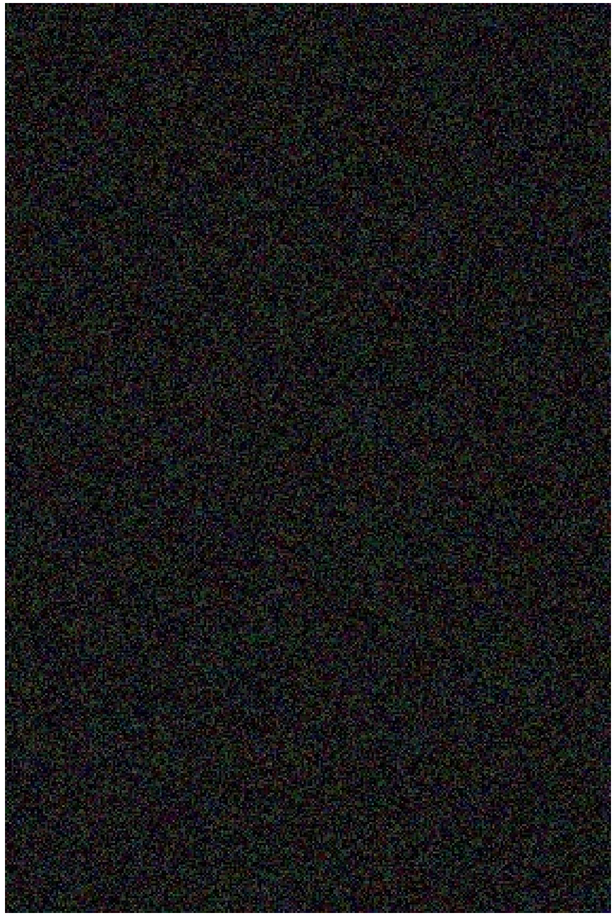


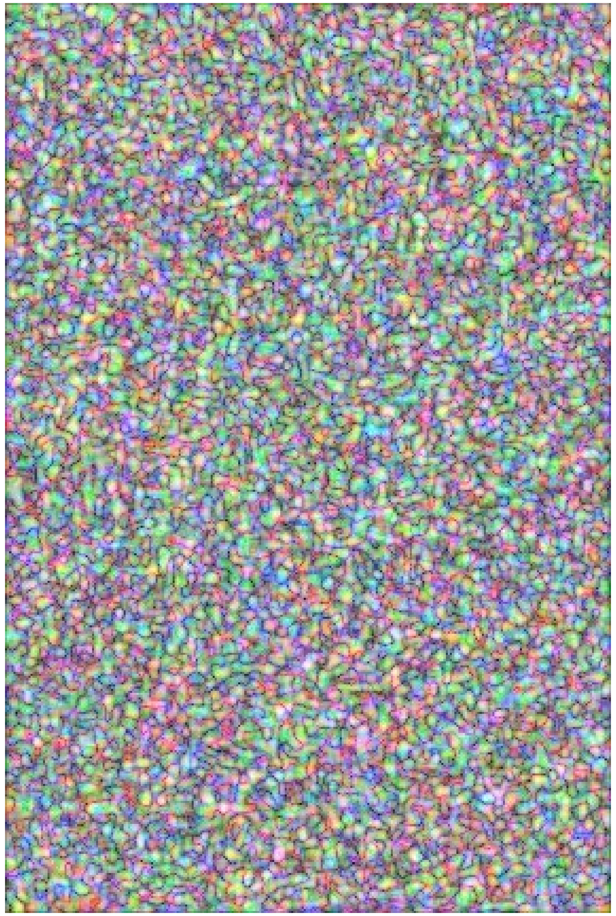
gradient regularization

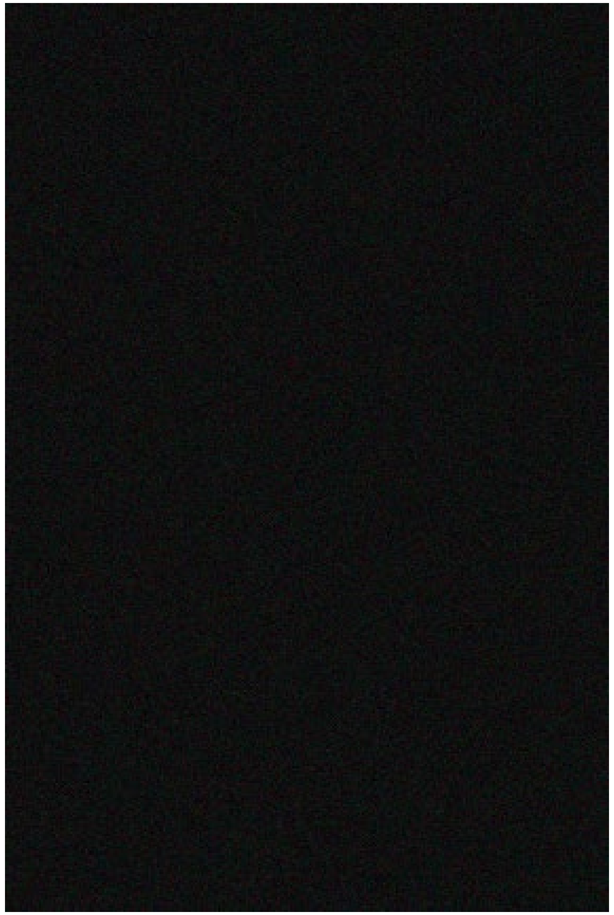


original

… and a proof-of-concept demonstration


noisy input


naive deconvolution


gradient regularization

Question

Can we undo lens blur by deconvolving a PNG or JPEG image without any preprocessing?

Question

Can we undo lens blur by deconvolving a PNG or JPEG image without any preprocessing?

- All the blur processes we discuss today happen *optically* (before capture by the sensor).
- Blur model is accurate only if our images are *linear.*

Are PNG or JPEG images linear?

Question

Can we undo lens blur by deconvolving a PNG or JPEG image without any preprocessing?

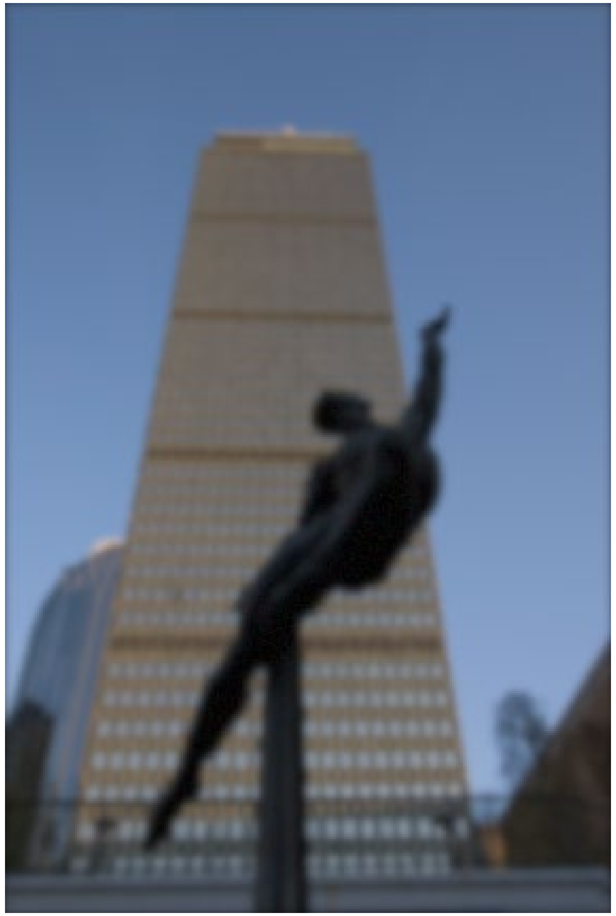
- All the blur processes we discuss today happen *optically* (before capture by the sensor).
- Blur model is accurate only if our images are *linear.*

Are PNG or JPEG images linear?

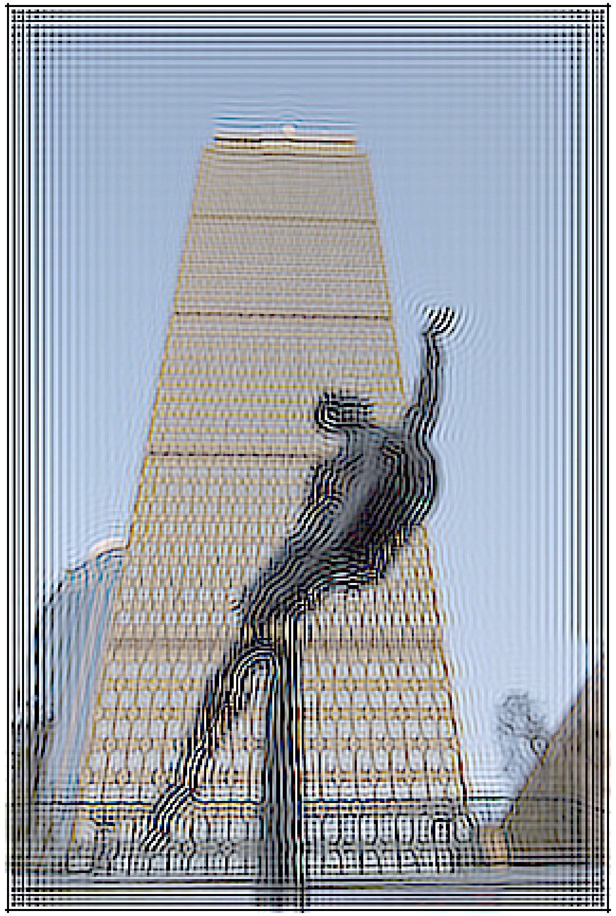
- No, because of gamma encoding.
- Before deblurring, you must linearize your images.

How do we linearize PNG or JPEG images?

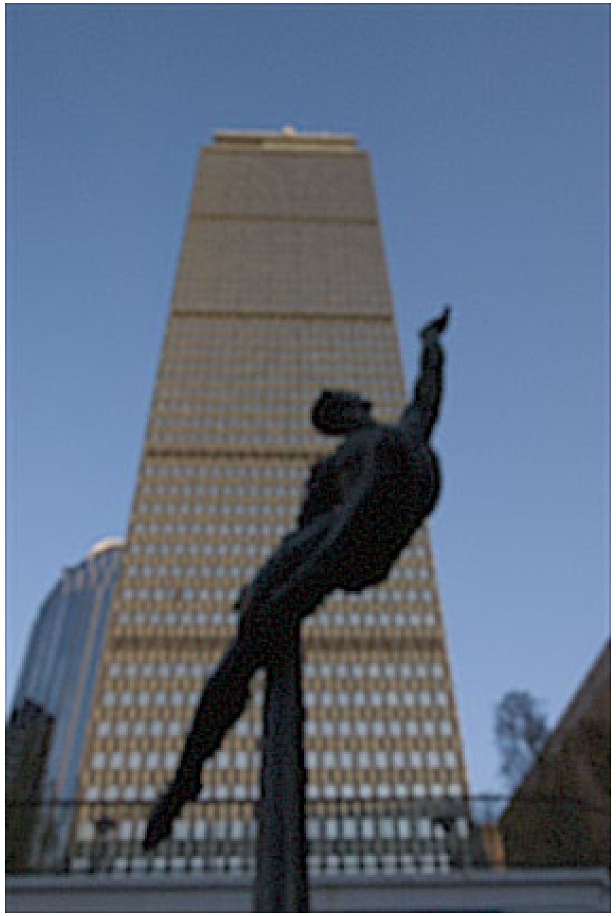
The importance of linearity



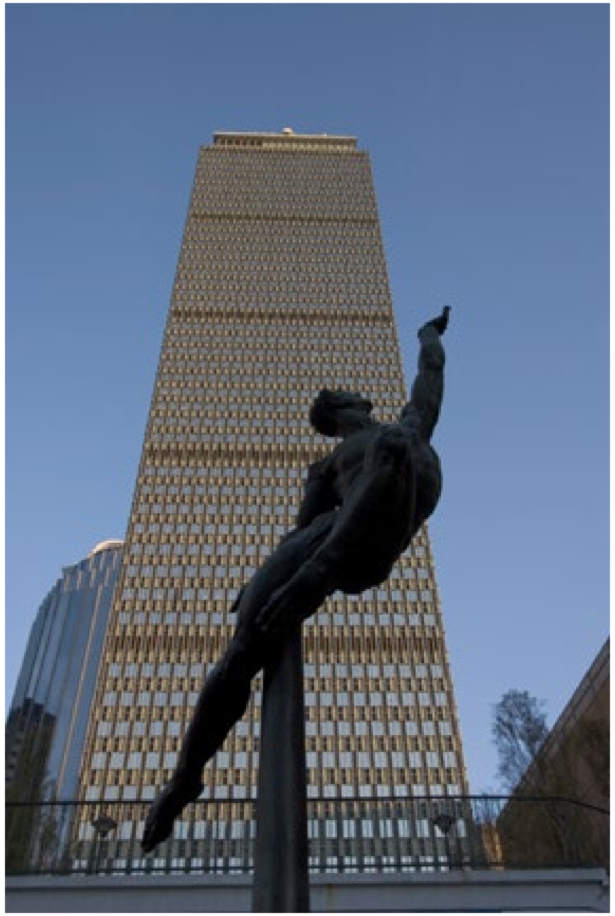
blurry input



deconvolution without linearization



deconvolution after linearization



original

Can we do better than that?

Can we do better than that?

Use different gradient regularizations:

- L_2 gradient regularization (Tikhonov regularization, same as Wiener deconvolution)

$$\min_x ||b - c * x||^2 + ||\nabla x||_2{}^2$$

- L_1 gradient regularization (sparsity regularization, *isotropic total variation*)

$$\min_x ||b - c * x||^2 + ||\nabla x||_1{}^1$$

- *Anisotropic total variation*

$$\min_x ||b - c * x||^2 + ||\nabla x||_2$$

How are these two different?

All of these are motivated by natural image statistics. Active research area.

Total Variation

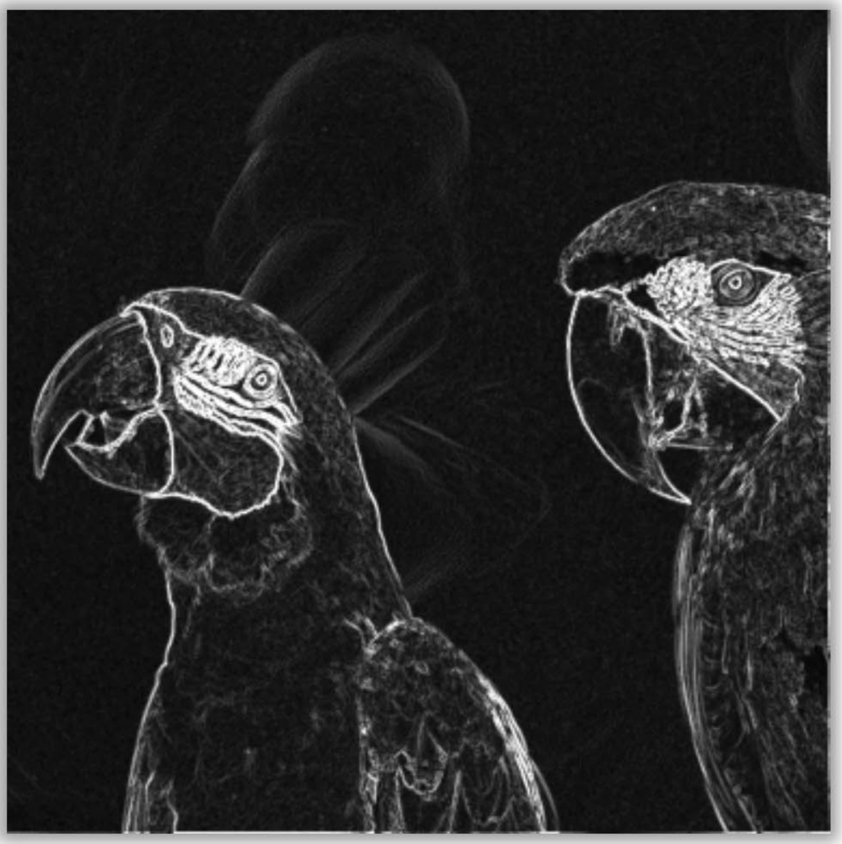
x

better: isotropic

$$\sqrt{(\nabla_x x)^2 + (\nabla_y x)^2}$$

easier: anisotropic

$$\sqrt{(\nabla_x x)^2} + \sqrt{(\nabla_y x)^2}$$



Total Variation

$$\underset{x}{\text{minimize}} \|Cx-b\|_2^2 + \lambda TV(x) = \underset{x}{\text{minimize}} \|Cx-b\|_2^2 + \lambda \|\nabla x\|_1$$

$$\|x\|_1 = \sum_i |x_i|$$

- idea: promote sparse gradients (edges)

- ∇ is finite differences operator, i.e. matrix $\begin{bmatrix} -1 & 1 & & \\ & -1 & 1 & \\ & & \ddots & \\ & & & -1 \end{bmatrix}$

Rudin et al. 1992

Total Variation

- for simplicity, this lecture only discusses anisotropic TV:

$$TV(x) = \left\| \nabla_x x \right\|_1 + \left\| \nabla_y x \right\|_1 = \left\| \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} x \right\|_1$$

- problem: l1-norm is not differentiable, can't use inverse filtering

- however: simple solution for data fitting along and simple solution for TV alone → split problem!

Deconvolution with ADMM

- split deconvolution with TV prior:

$$\begin{array}{ll} \text{minimize} & \|Cx-b\|_2^2 + \lambda\|z\|_1 \\ \text{subject to} & \nabla x = z \end{array}$$

- general form of ADMM (alternating direction method of multiplies):

$$\begin{array}{ll} \text{minimize} & f(x)+g(z) \\ \text{subject to} & Ax+Bz=c \end{array}$$

$$f(x)=\|Cx-b\|_2^2$$

$$g(z)=\lambda\|z\|_1$$

$$A=\nabla,\ B=-I,c=0$$

ADMM

$$\begin{array}{ll} \text{minimize} & f(x)+g(z) \\ \text{subject to} & Ax+Bz=c \end{array}$$

- Lagrangian (bring constraints into objective = penalty method):

$$L(x,y,z)=f(x)+g(z)+y^T(Ax+Bz-c)$$

dual variable or Lagrange multiplier

ADMM

$$\begin{array}{ll} \text{minimize} & f(x)+g(z) \\ \text{subject to} & Ax+Bz=c \end{array}$$

- augmented Lagrangian is differentiable under mild conditions (usually better convergence etc.)

$$L_\rho(x,y,z) = f(x)+g(z)+y^T(Ax+Bz-c)+(\rho/2)\|Ax+Bz-c\|_2^2$$

ADMM

$$\begin{array}{ll} \text{minimize} & f(x)+g(z) \\ \text{subject to} & Ax+Bz=c \end{array}$$

- ADMM consists of 3 steps per iteration k:

$$\begin{aligned} x^{k+1} &:= \arg\min_x L_\rho(x, z^k, y^k) \\ z^{k+1} &:= \arg\min_z L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} &:= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

ADMM

$$\begin{array}{ll} \text{minimize} & f(x)+g(z) \\ \text{subject to} & Ax+Bz=c \end{array}$$

- ADMM consists of 3 steps per iteration k:

constant

$$x^{k+1} := \arg\min_x \left(f(x) + (\rho/2) \left\| Ax + \boxed{Bz^k - c + u^k} \right\| \right)$$

$$z^{k+1} := \arg\min_z \left(g(z) + (\rho/2) \left\| \boxed{Ax^{k+1}} + Bz \boxed{- c + u^k} \right\| \right)$$

$$u^{k+1} := u^k + Ax^{k+1} + Bz^{k+1} - c$$

scaled dual variable: $u = (1/\rho)y$

ADMM

$$\begin{array}{ll} \text{minimize} & f(x)+g(z) \\ \text{subject to} & Ax+Bz=c \end{array}$$

- ADMM consists of 3 steps per iteration k:

split f(x) and g(x) into independent problems!

(u connects them)

$$x^{k+1} := \arg\min_x \left(f(x) + (\rho/2) \left\| Ax + Bz^k - c + u^k \right\|_2^2 \right)$$

$$z^{k+1} := \arg\min_z \left(g(z) + (\rho/2) \left\| Ax^{k+1} + Bz - c + u^k \right\|_2^2 \right)$$

$$u^{k+1} := u^k + Ax^{k+1} + Bz^{k+1} - c$$

scaled dual variable: $u = (1/\rho)y$

Deconvolution with ADMM

$$\text{minimize} \quad \frac{1}{2}\|Cx-b\|_2^2+\lambda\|z\|_1$$

$$\text{subject to} \quad \nabla x - z = 0$$

- ADMM consists of 3 steps per iteration k:

$$x^{k+1} := \arg\min_x \left(\frac{1}{2}\|Cx-b\|_2^2+(\rho/2)\left\|\nabla x - z^k + u^k\right\|_2^2\right)$$

$$z^{k+1} := \arg\min_z \left(\lambda\|z\|_1+(\rho/2)\left\|\nabla x^{k+1} - z + u^k\right\|_2^2\right)$$

$$u^{k+1} := u^k + \nabla x^{k+1} - z^{k+1}$$

Deconvolution with ADMM

$$\text{minimize} \quad \frac{1}{2}\|Cx-b\|_2^2 + \lambda\|z\|_1$$

$$\text{subject to} \qquad \nabla x - z = 0$$

constant, say $v = z^k - u^k$

1. x-update:

$$x^{k+1} := \arg\min_x \left(\frac{1}{2}\|Cx-b\|_2^2 + (\rho/2)\left\|\nabla x \boxed{- z^k + u^k}\right\|_2^2 \right)$$

solve normal equations $\left(C^T C + \rho \nabla^T \nabla\right) x = \left(C^T b + \rho \nabla^T v\right)$

$$\nabla^T v = \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix}^T v = \nabla_x^T v_1 + \nabla_y^T v_2$$

Deconvolution with ADMM

$$\text{minimize} \quad \frac{1}{2}\|Cx-b\|_2^2 + \lambda\|z\|_1$$

$$\text{subject to} \quad \nabla x - z = 0$$

constant, say $v = z^k - u^k$

1. x-update:

$$x^{k+1} := \arg\min_x \left(\frac{1}{2}\|Cx-b\|_2^2 + (\rho/2)\left\|\nabla x \boxed{- z^k + u}^k\right\|_2^2 \right)$$

$$x = \left(C^T C + \rho\nabla^T\nabla\right)^{-1}\left(C^T b + \rho\nabla^T v\right)$$

- inverse filtering: $x^{k+1} = F^{-1}\left\{ \frac{\boxed{F\{c\}^* \cdot F\{b\}} + \rho\left(\boxed{F\{\nabla_x\}}^* \cdot F\{v_1\} + \boxed{F\{\nabla_y\}}^* \cdot F\{v_2\}\right)}{\boxed{F\{c\}^* \cdot F\{c\} + \rho\left(F\{\nabla_x\}^* \cdot F\{\nabla_x\} + F\{\nabla_y\}^* \cdot F\{\nabla_y\}\right)}} \right\}$

precompute!

Deconvolution with ADMM

$$\text{minimize} \quad \frac{1}{2}\|Cx-b\|_2^2 + \lambda\|z\|_1$$

$$\text{subject to} \quad \nabla x - z = 0$$

constant, say $a = \nabla x^{k+1} + u^k$

2. z-update: $$z^{k+1} := \arg\min_z \left(\lambda\|z\|_1 + (\rho/2)\|\boxed{\nabla x^{k+1}} - z + \boxed{u^k}\|_2^2 \right)$$

Deconvolution with ADMM

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|Cx-b\|_2^2+\lambda\|z\|_1 \\ \text{subject to} & \nabla x - z = 0 \end{array}$$

for k=1:max_iters

$$x^{k+1} := \arg\min_x \left(\frac{1}{2} \left\| \begin{bmatrix} C \\ \rho\nabla \end{bmatrix} x - \begin{bmatrix} b \\ \rho v \end{bmatrix} \right\|_2^2 \right)$$ inverse filtering

$$z^{k+1} := S_{\lambda/\rho}(\nabla x^{k+1} + u^k)$$ element-wise threshold

$$u^{k+1} := u^k + \nabla x^{k+1} - z^{k+1}$$ trivial

Deconvolution comparisons

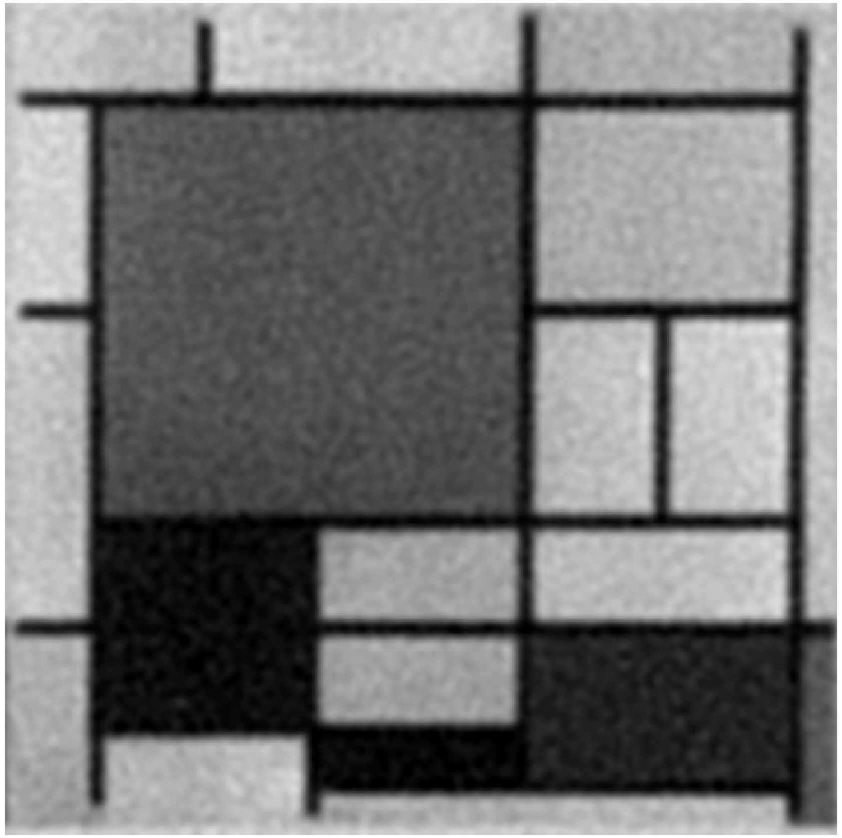
Wiener deconvolution

ADMM + TV, λ = 0.01

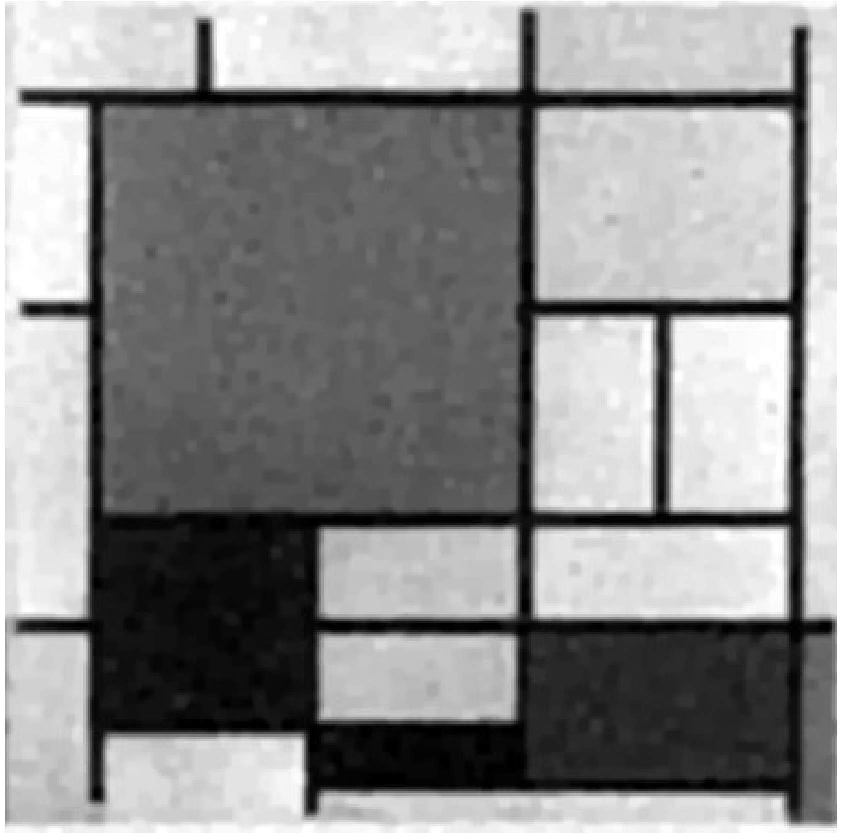
ADMM + TV, λ = 0.1

- image becomes too flat as we increase weight of TV prior
- Image becomes too noisy as we decrease weight of TV prior

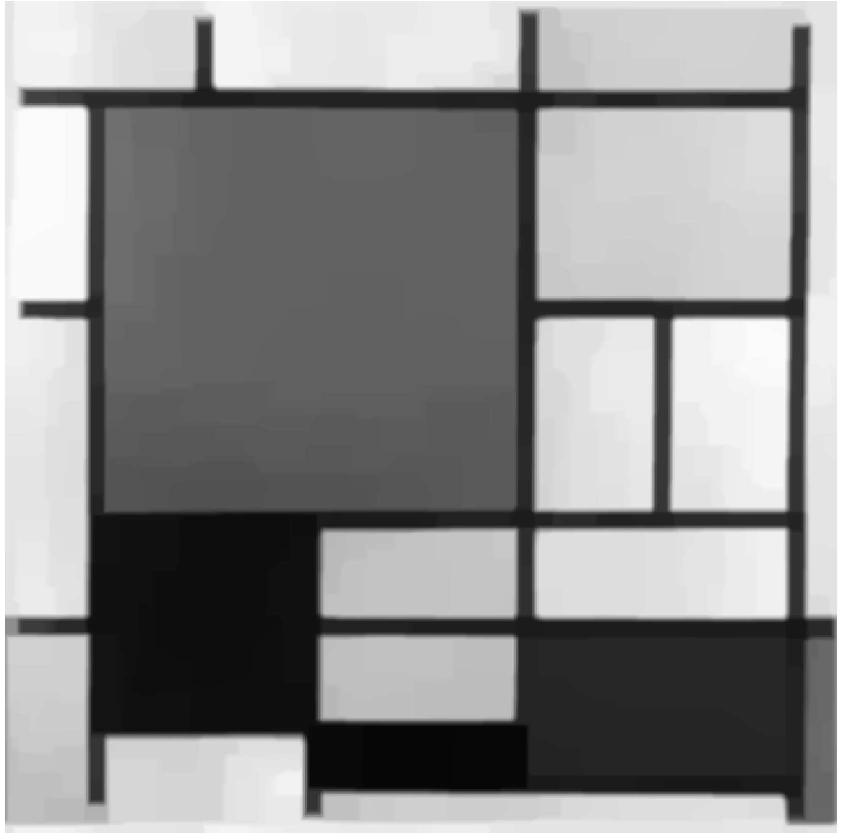
Deconvolution comparisons



Wiener deconvolution



ADMM + TV, λ = 0.01



ADMM + TV, λ = 0.1

- image becomes too flat as we increase weight of TV prior
- Image becomes too noisy as we decrease weight of TV prior

Outlook ADMM

- powerful tool for many computational imaging problems
- include generic prior in g(z), just need to derive proximal operator

$$\underset{x}{\text{minimize}} \underbrace{\frac{1}{2}\|Ax-b\|_2^2}_{\text{data fidelity}} + \underbrace{\Gamma(x)}_{\text{regularization}} \quad \longrightarrow \quad \begin{array}{ll} \underset{\{x,z\}}{\text{minimize}} & f(x)+g(z) \\ \text{subject to} & Ax = z \end{array}$$

- example priors: noise statistics, sparse gradient, smoothness, …
- weighted sum of different priors also possible
- anisotropic TV is one of the easiest priors

Can we do better than that?

Use different gradient regularizations:

- L_2 gradient regularization (Tikhonov regularization, same as Wiener deconvolution)

$$\min_x ||b - c * x||^2 + ||\nabla x||_2^{\,2}$$

- L_1 gradient regularization (sparsity regularization, same as *total variation*)

$$\min_x ||b - c * x||^2 + ||\nabla x||_1^{\,1}$$

- $L_{n<1}$ gradient regularization (fractional regularization)

$$\min_x ||b - c * x||^2 + ||\nabla x||_{0.8}^{\,0.8}$$

All of these are motivated by natural image statistics. Active research area.

Comparison of gradient regularizations

input

squared gradient regularization

fractional gradient regularization

Derivation

Sensing model:

$$\tilde{x} = c * x + n$$

Noise $\mathbf{n}$ is assumed to be zero-mean and independent of signal $\mathbf{x}$.

Is this a reasonable noise model?

Richardson-Lucy Algorithm + TV

- log-likelihood function:

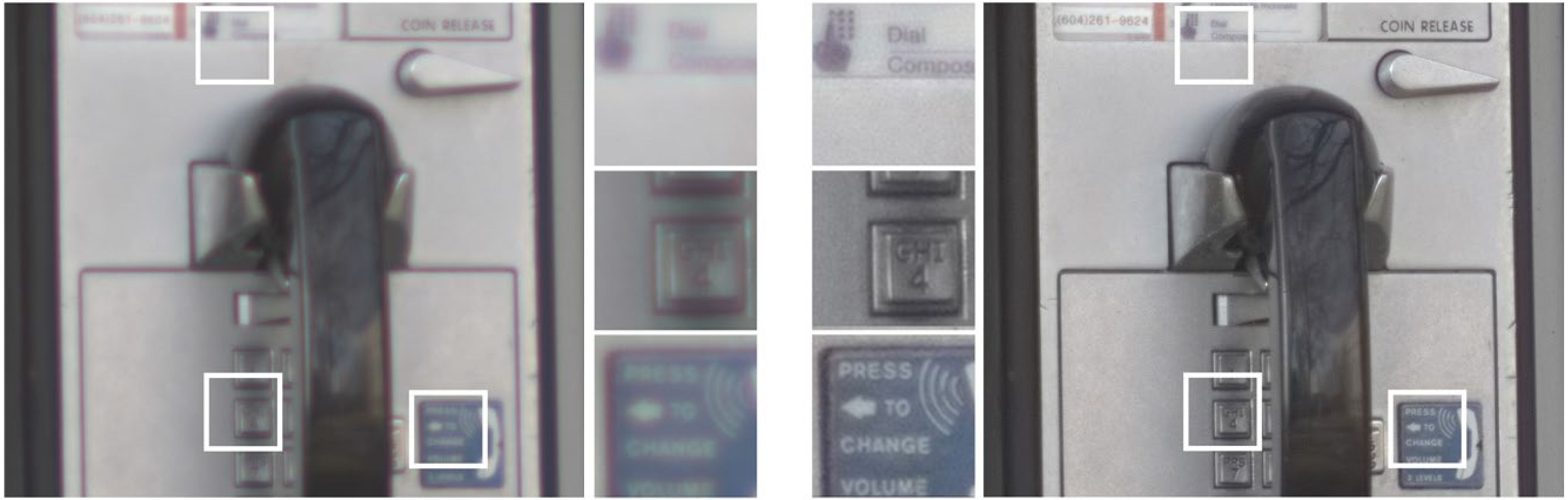
$$\log\left(L_{TV}\left(\mathbf{x}\right)\right) = \log\left(p\left(\mathbf{b}|\mathbf{x}\right)\right) + \log\left(p\left(\mathbf{x}\right)\right) = \log\left(\mathbf{A}\mathbf{x}\right)^T \mathbf{b} - \left(\mathbf{A}\mathbf{x}\right)^T \mathbf{1} - \sum_{i=1}^{M} \log\left(\mathbf{b}_i!\right) - \lambda \left\|\mathbf{D}\mathbf{x}\right\|_1$$

- gradient:

$$\nabla \log\left(L_{TV}\left(\mathbf{x}\right)\right) = \mathbf{A}^T \mathrm{diag}\left(\mathbf{A}\mathbf{x}\right)^{-1} \mathbf{b} - \mathbf{A}^T \mathbf{1} + \nabla\lambda \left\|\nabla \mathbf{x}\right\|_1 = \mathbf{A}^T \left(\frac{\mathbf{b}}{\mathbf{A}\mathbf{x}}\right) - \mathbf{A}^T \mathbf{1} - \nabla\lambda \left\|\mathbf{D}\mathbf{x}\right\|_1$$

- recover signal by setting gradient to zero
- generally challenging

High quality images using cheap lenses



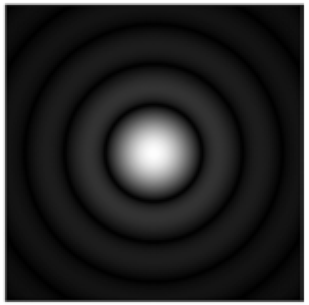
[Heide et al., “High-Quality Computational Imaging Through Simple Lenses,” TOG 2013]

Deconvolution

If we know b and c, can we recover x?

$*$

How do we measure this?



$=$

$$x \quad * \quad c \quad = \quad b$$

PSF calibration

Take a photo of a point source

Image of PSF

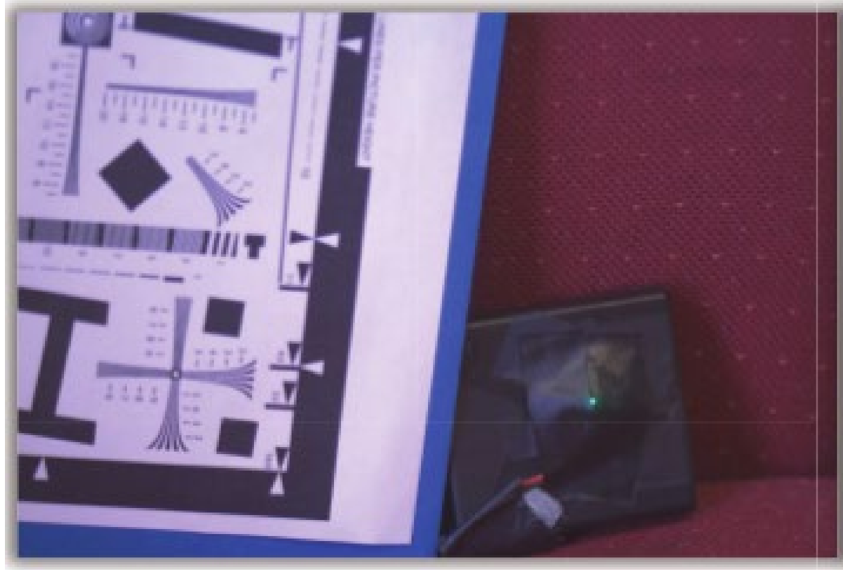


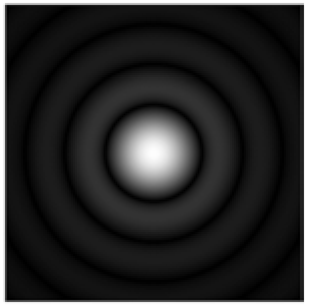
Image with sharp lens

Image with cheap lens

Deconvolution

If we know b and c, can we recover x?

$*$



$=$

$$x * c = b$$

Blind deconvolution

If we know b, can we recover x and c?

*

=

$$x * c = b$$

Camera shake

Removing Camera Shake from a Single Photograph

Rob Fergus[1] Barun Singh[1] Aaron Hertzmann[2] Sam T. Roweis[2] William T. Freeman[1]

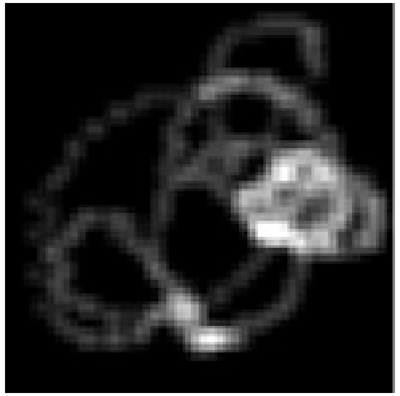
[1]MIT CSAIL [2]University of Toronto

Figure 1: *Left*: An image spoiled by camera shake. *Middle*: result from Photoshop "unsharp mask". *Right*: result from our algorithm.

Camera shake as a filter

If we know b, can we recover x and c?

$*$



$=$

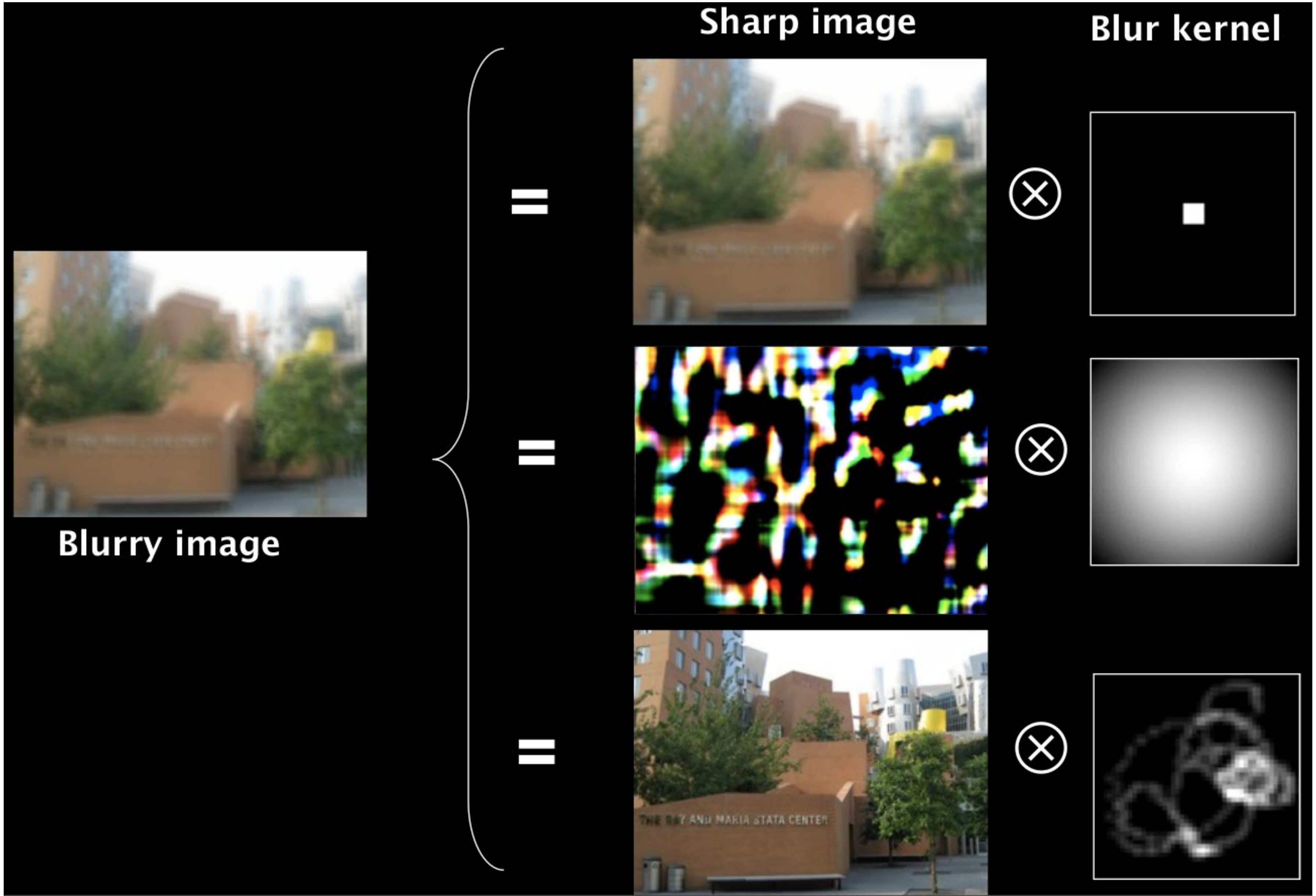
image from static camera

PSF from camera motion

image from shaky camera

$$x \quad * \quad c \quad = \quad b$$

Multiple possible solutions



How do we detect this one?

Use prior information

Among all the possible pairs of images and blur kernels, select the ones where:

- $\boxed{\text{The image “looks like” a natural image.}}$

- The kernel “looks like” a motion PSF.

Use prior information

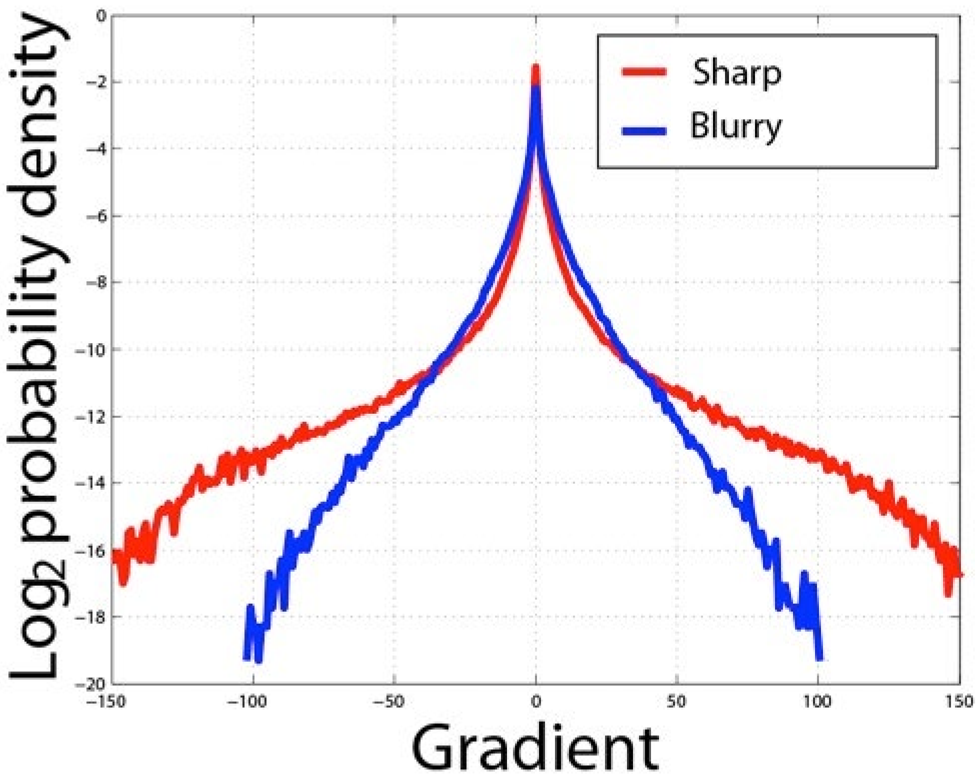
Among all the possible pairs of images and blur kernels, select the ones where:

- The image “looks like” a natural image.

- The kernel “looks like” a motion PSF.

Shake kernel statistics

Gradients in natural images follow a characteristic “heavy-tail” distribution.

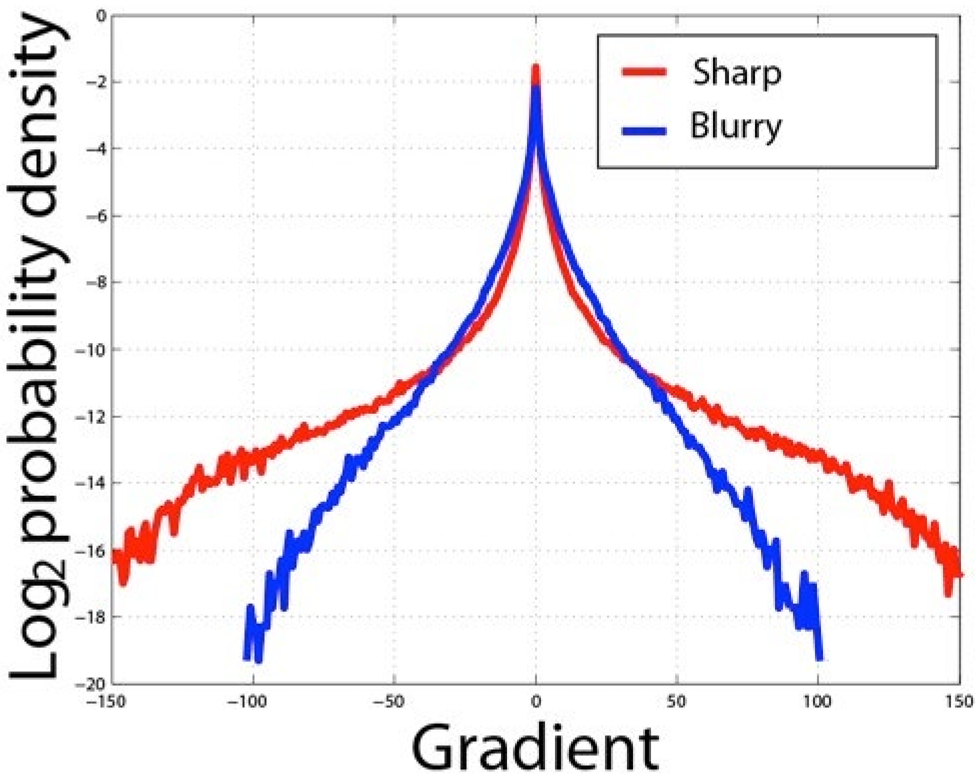


sharp natural image

blurry natural image

Shake kernel statistics

Gradients in natural images follow a characteristic “heavy-tail” distribution.



Can be approximated by $\|\nabla x\|^{0.8}$

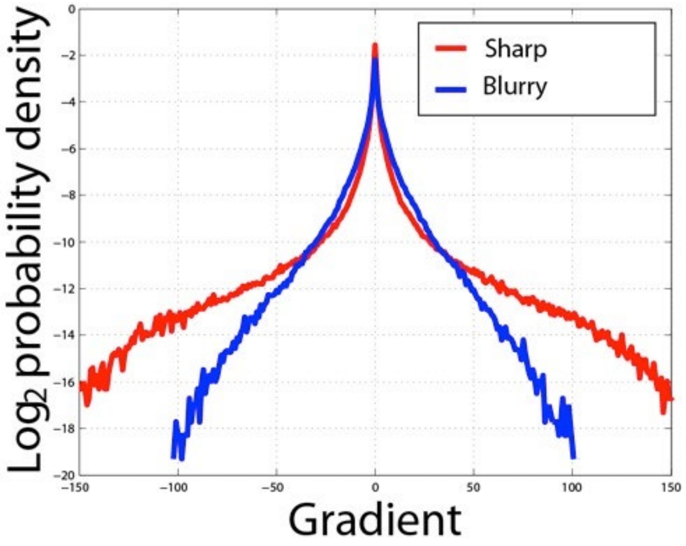
sharp natural image

blurry natural image

Use prior information

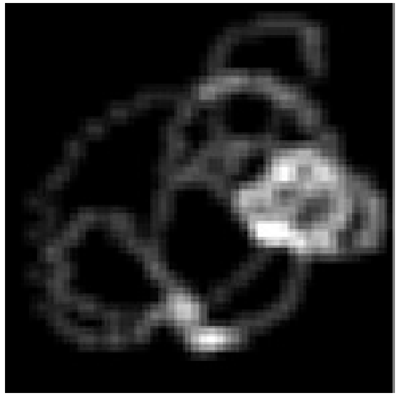
Among all the possible pairs of images and blur kernels, select the ones where:

- The image "looks like" a natural image.



Gradients in natural images follow a characteristic "heavy-tail" distribution.

- The kernel "looks like" a motion PSF.



Shake kernels are very sparse, have continuous contours, and are always positive

How do we use this information for blind deconvolution?

Regularized blind deconvolution

Solve regularized least-squares optimization

$$\min_{x,b} ||b - c * x||^2 + ||\nabla x||^{0.8} + ||c||_1$$

What does each term in this summation correspond to?

Regularized blind deconvolution

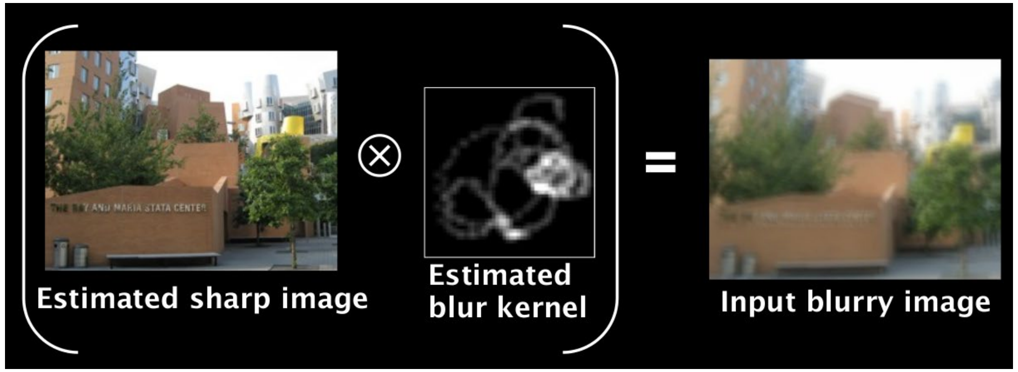
Solve regularized least-squares optimization

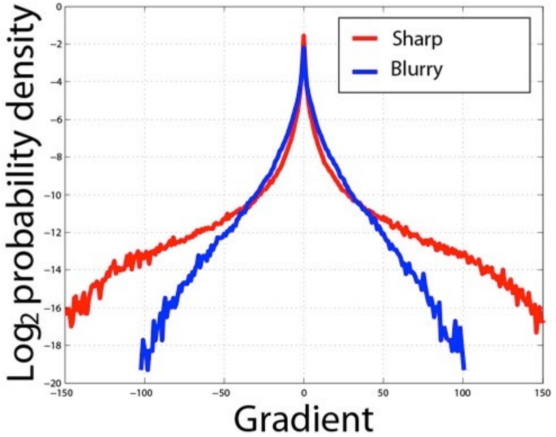
$$\min_{x,b} ||b - c * x||^2 + ||\nabla x||^{0.8} + ||c||_1$$

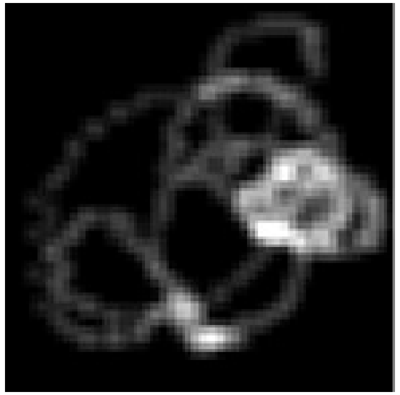
data term

natural image prior

shake kernel prior





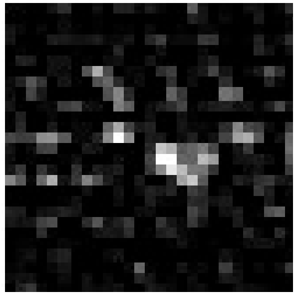


Note: Solving such optimization problems is complicated (no longer *linear* least squares).

A demonstration

input

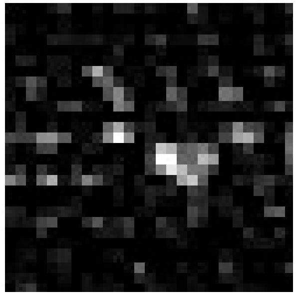
deconvolved image and kernel



A demonstration

input

deconvolved image and kernel



This image looks worse than the original…

This doesn't look like a plausible shake kernel…

Regularized blind deconvolution

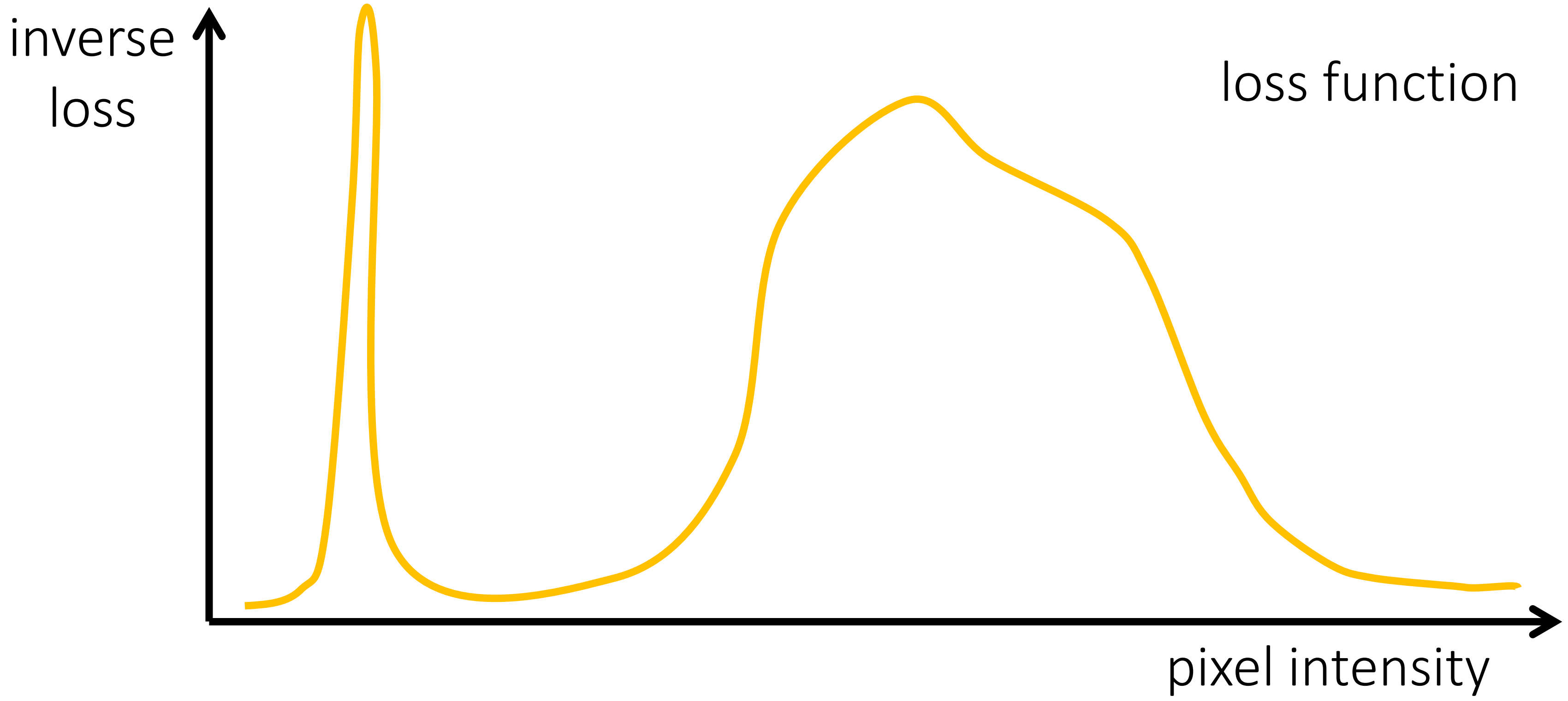
Solve regularized least-squares optimization

$$\min_{x,b} \underbrace{||b - c * x||^2 + ||\nabla x||^{0.8} + ||c||_1}_{\text{loss function}}$$

Regularized blind deconvolution

Solve regularized least-squares optimization

$$\min_{x,b} \underbrace{\|b - c * x\|^2 + \|\nabla x\|^{0.8} + \|c\|_1}_{\text{loss function}}$$

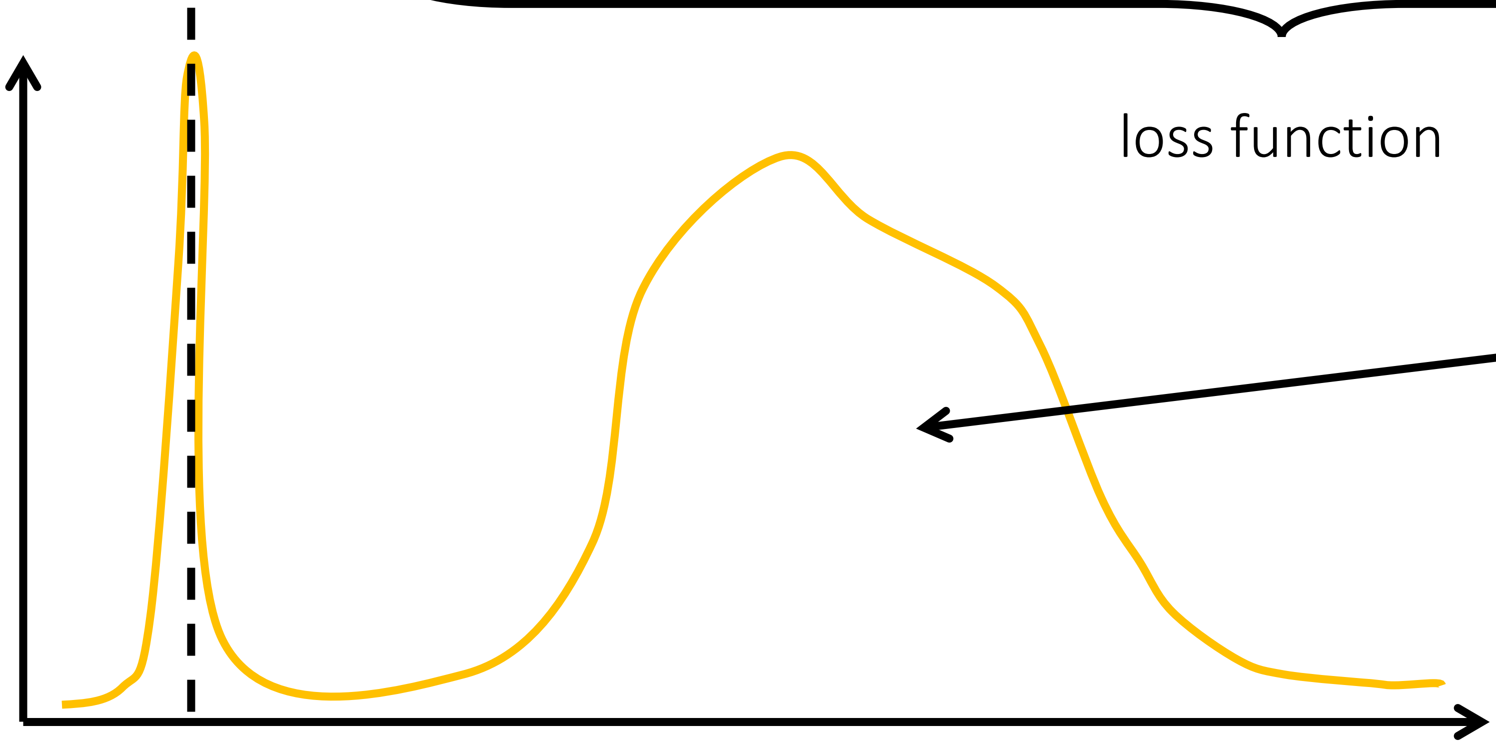


Where in this graph is the solution we find?

Regularized blind deconvolution

Solve regularized least-squares optimization

$$\min_{x,b} \underbrace{||b - c * x||^2 + ||\nabla x||^{0.8} + ||c||_1}_{\text{loss function}}$$

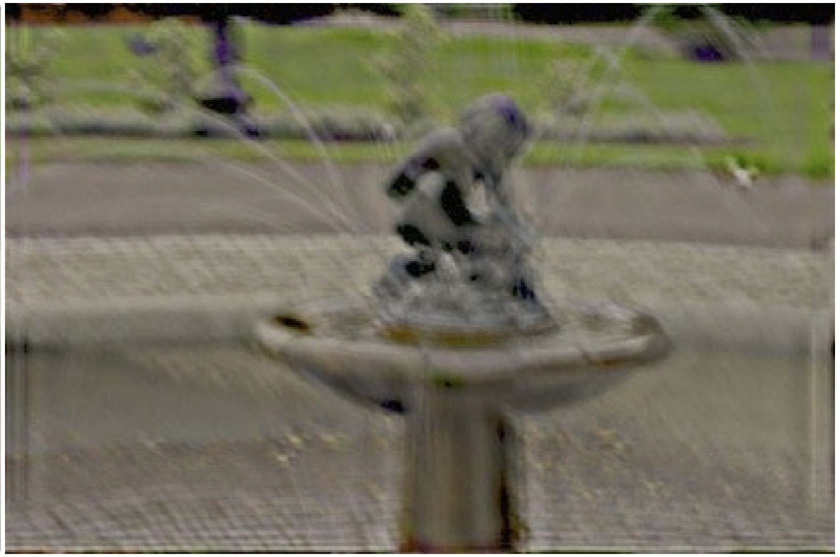


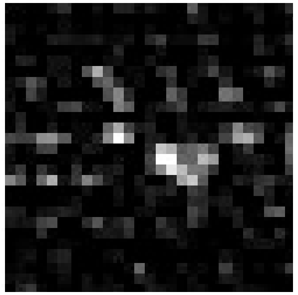
Rather than keep just maximum, do a weighted average of all solutions

A demonstration

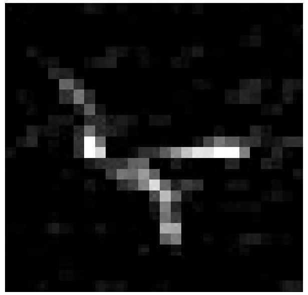
input

maximum-only

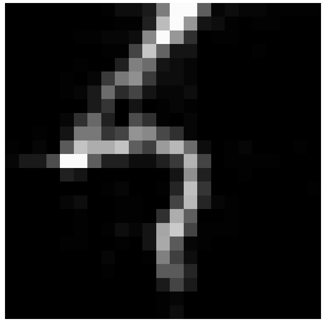


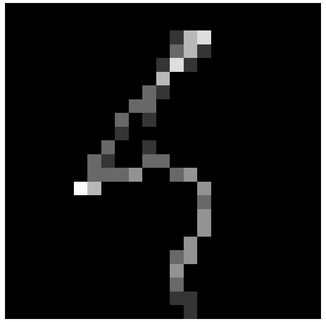


average

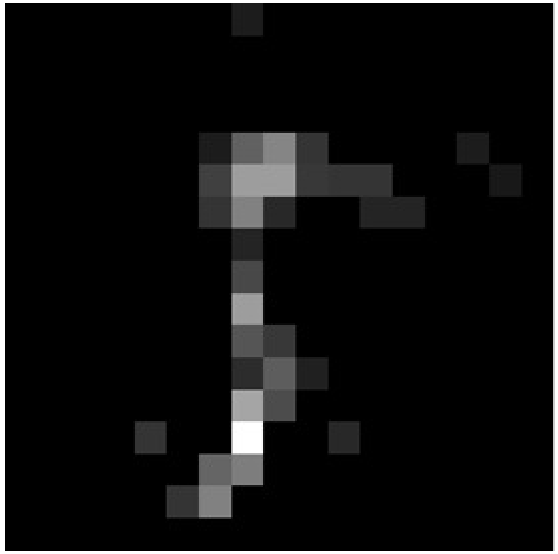


More examples



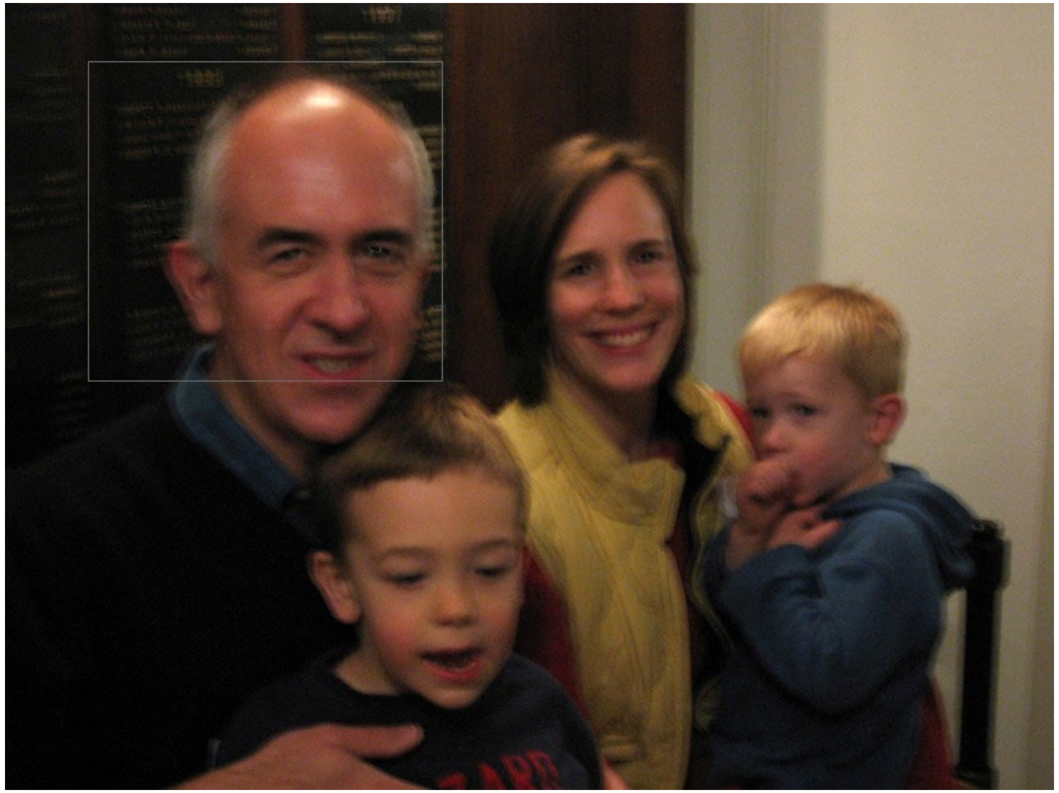


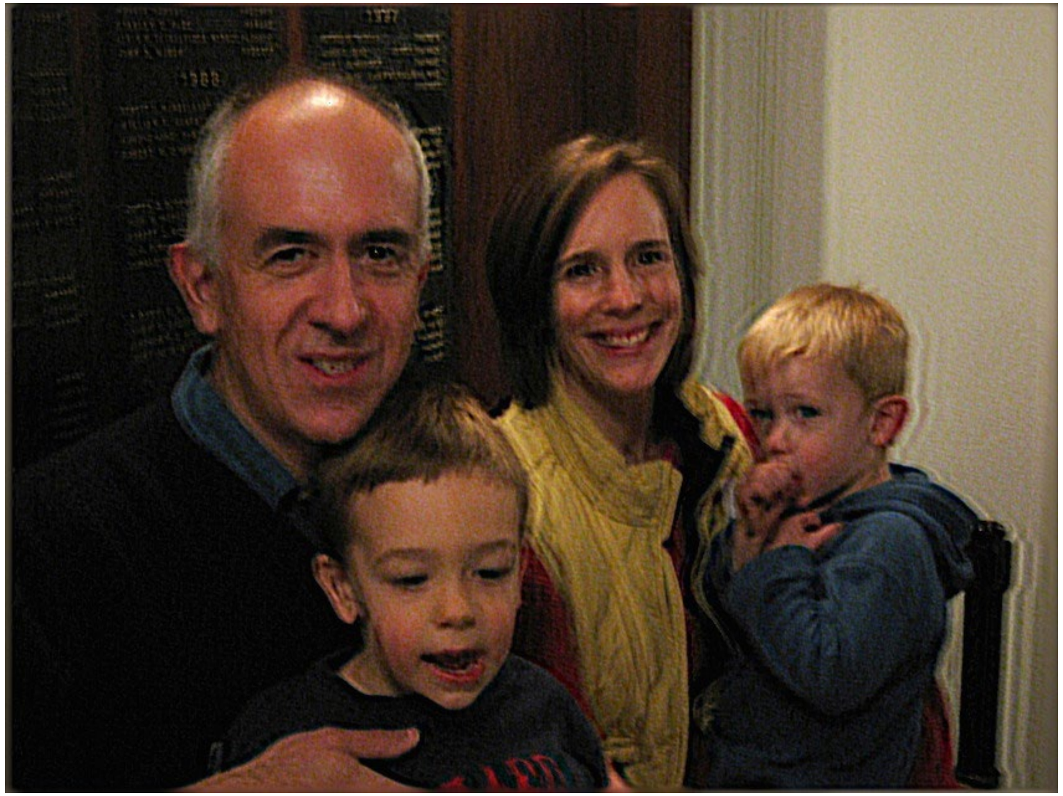
Results on real shaky images

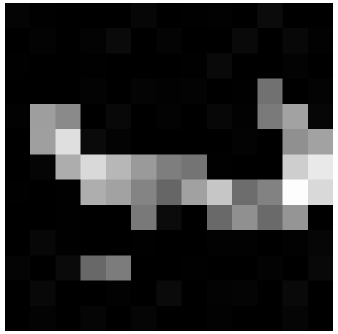


Results on real shaky images

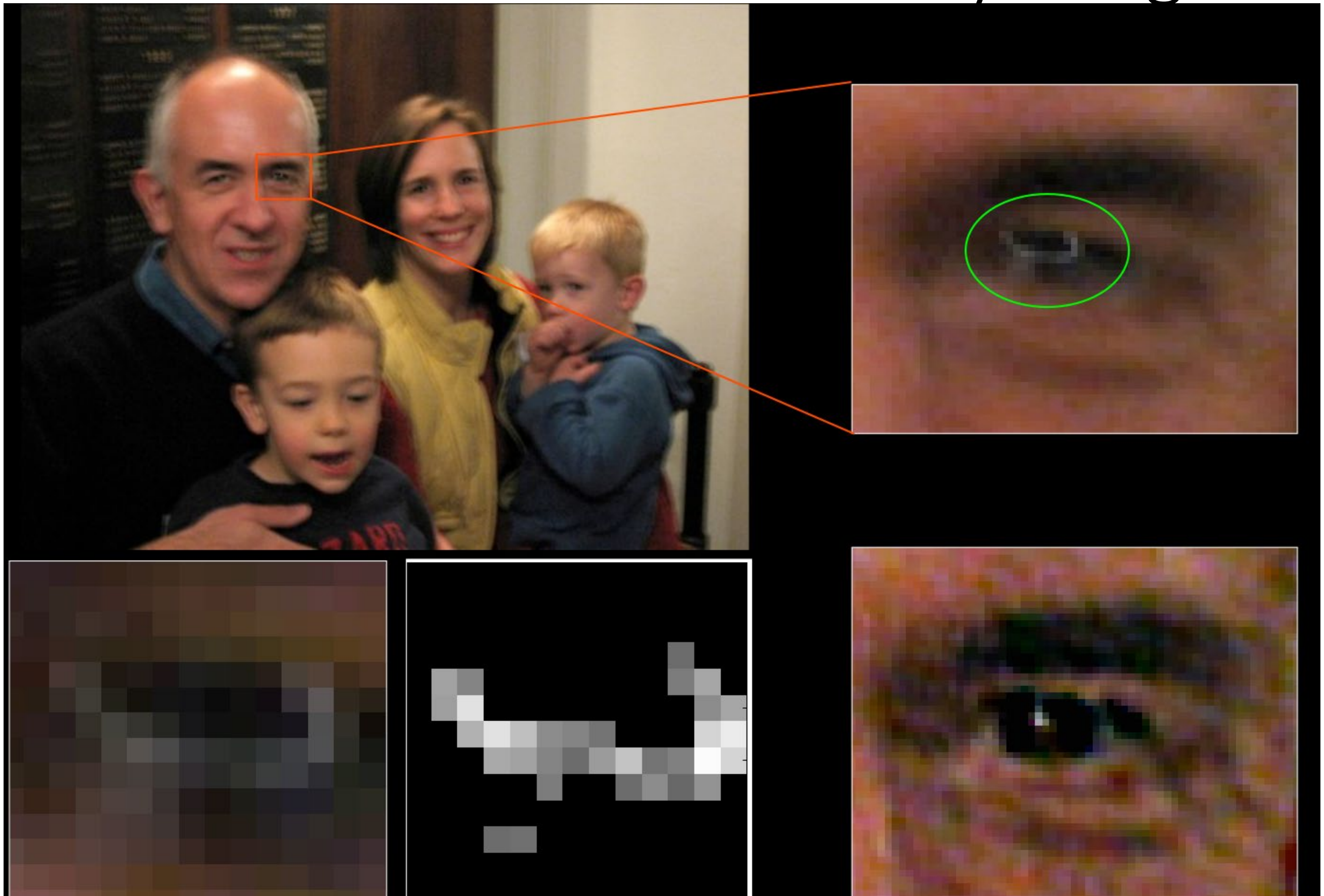
Results on real shaky images

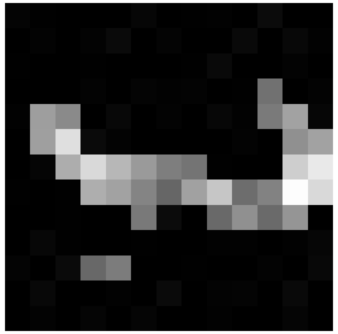






Results on real shaky images





More advanced motion deblurring

[Shah et al., High-quality Motion Deblurring from a Single Image, SIGGRAPH 2008]

Why are our images blurry?

- Lens imperfections.
- Camera shake.
- Scene motion.
- Depth defocus.

Can we solve all of these problems using (blind) deconvolution?

Why are our images blurry?

- Lens imperfections.
- Camera shake.
- Scene motion.
- Depth defocus.

Can we solve all of these problems using (blind) deconvolution?

- We can deal with (some) lens imperfections and camera shake, because their blur is shift invariant.
- We cannot deal with scene motion and depth defocus, because their blur is not shift invariant.
- See coded photography lecture.

References

Basic reading:
- Szeliski textbook, Sections 3.4.3, 3.4.4, 10.1.4, 10.3.
- Fergus et al., “Removing camera shake from a single image,” SIGGRAPH 2006.
  the main motion deblurring and blind deconvolution paper we covered in this lecture.

Additional reading:
- Heide et al., “High-Quality Computational Imaging Through Simple Lenses,” TOG 2013.
  the paper on high-quality imaging using cheap lenses, which also has a great discussion of all matters relating to blurring from lens aberrations and modern deconvolution algorithms.
- Levin, “Blind Motion Deblurring Using Image Statistics,” NIPS 2006.
- Levin et al., “Image and depth from a conventional camera with a coded aperture,” SIGGRAPH 2007.
- Levin et al., “Understanding and evaluating blind deconvolution algorithms,” CVPR 2009 and PAMI 2011.
- Krishnan and Fergus, “Fast Image Deconvolution using Hyper-Laplacian Priors,” NIPS 2009.
- Levin et al., “Efficient Marginal Likelihood Optimization in Blind Deconvolution,” CVPR 2011.
  a sequence of papers developing the state of the art in blind deconvolution of natural images, including the use Laplacian (sparsity) and hyper-Laplacian priors on gradients, analysis of different loss functions and maximum a-posteriori versus Bayesian estimates, the use of variational inference, and efficient optimization algorithms.
- Minskin and MacKay, “Ensemble Learning for Blind Image Separation and Deconvolution,” AICA 2000.
  the paper explaining the mathematics of how to compute Bayesian estimators using variational inference.
- Shah et al., “High-quality Motion Deblurring from a Single Image,” SIGGRAPH 2008.
  a more recent paper on motion deblurring.