CS448f: Image Processing For Photography and Vision

Deconvolution

Assignment 3

- Competition
- Lessons Learned

Project

- Proposals due Thursday
- Everyone should have a pretty good idea of what they plan to do at this stage
- Presentations begin next Tuesday
- Schedule?

Problems in Photography

	Linear Filters	Non-Linear Filters	Alignment	Wavelets	Gradient Domain
Misfocus or Lens Blur	Sharpening	**Sharpening**	Focal Stacks Panoramas	Sharpening	?
Motion Blur	Sharpening	Sharpening	?	?	?
Noise	Blurring	**Bilateral Nonlocal Means**	Aligned Averaging	Wavelet Shrinkage	?
Dynamic Range	?	Tone-Mapping	HDR Acquisition	?	Tone-Mapping
Composition	Multi-Band Blending	?	Panoramas	?	Poisson Blending

Problems in Photography

	Linear Filters	Non-Linear Filters	Alignment	Wavelets	Gradient Domain	Deconvolution
Misfocus or Lens Blur	Sharpening	**Sharpening**	Focal Stacks Panoramas	Sharpening	?	✓
Motion Blur	Sharpening	Sharpening	?	?	?	✓
Noise	Blurring	**Bilateral Nonlocal Means**	Aligned Averaging	Wavelet Shrinkage	?	✗
Dynamic Range	?	Tone-Mapping	HDR Acquisition	?	Tone-Mapping	?
Composition	Multi-Band Blending	?	Panoramas	?	Poisson Blending	?

Motion Blur (Handheld 200mm 1/50 s)

Motion Blur (Handheld 200mm, 1/50s)

Less Motion Blur (1/640s)

Motion Blur (Rolling the camera)

Motion Blur = Convolution

*

=

Convolution = Linear Operator

- I**m**age ∗ **k**ernel = **b**lurry
- Km = b
  - K = the blur (may or may not be known)
  - m = the unknown good image
  - b = the known blurry image
- K is known = nonblind deconvolution
- K is unknown = blind deconvolution

Estimating K

- Include an accelerometer
- Look for the path traced by bright points
- Bounce back and forth between estimating K and estimating m
  - Deconvolution using Natural Image Priors
    - Levin et al. 2007

Deconvolution = Least Squares

- Assuming we know K
- Find m such that Km = b
- Alternatively, minimize $(Km-b)^2$

Solution Methods: Input

Solution Methods: Gradient Descent

Solution Methods: Richardson-Lucy

Solution Methods: Richardson-Lucy

- $m *= K^T(b/(Km))$
- Like a multiplicative gradient descent
- Each step conserves average brightness in each region
- `ImageStack -load blurry.tmp -loop --dup --load kernel.tmp --pull 1 --convolve --pull 1 --pop --load blurry.tmp --divide --load kernel.tmp --flip x --flip y --pull 1 --convolve --pull 1 --pop --multiply --save rl.tmp --display`

High-Frequency Junk

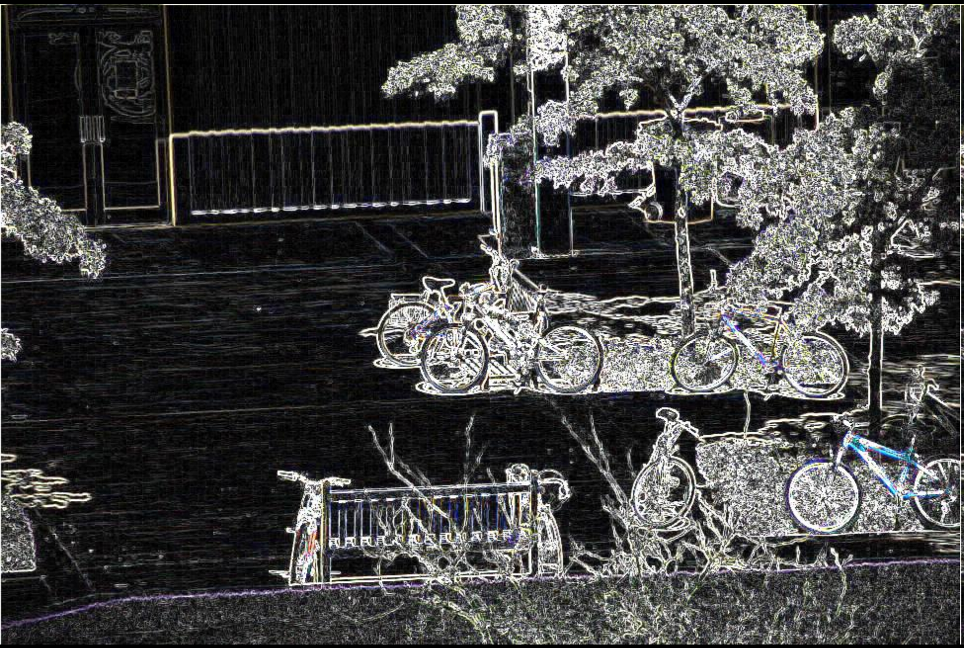
Priors

- The result image above satisfies the equation:
  - Km = b
- Why does it look bad?

Priors

- The result image above satisfies the equation:
  - $Km = b$
- Why does it look bad?
- There's extra high-frequency junk

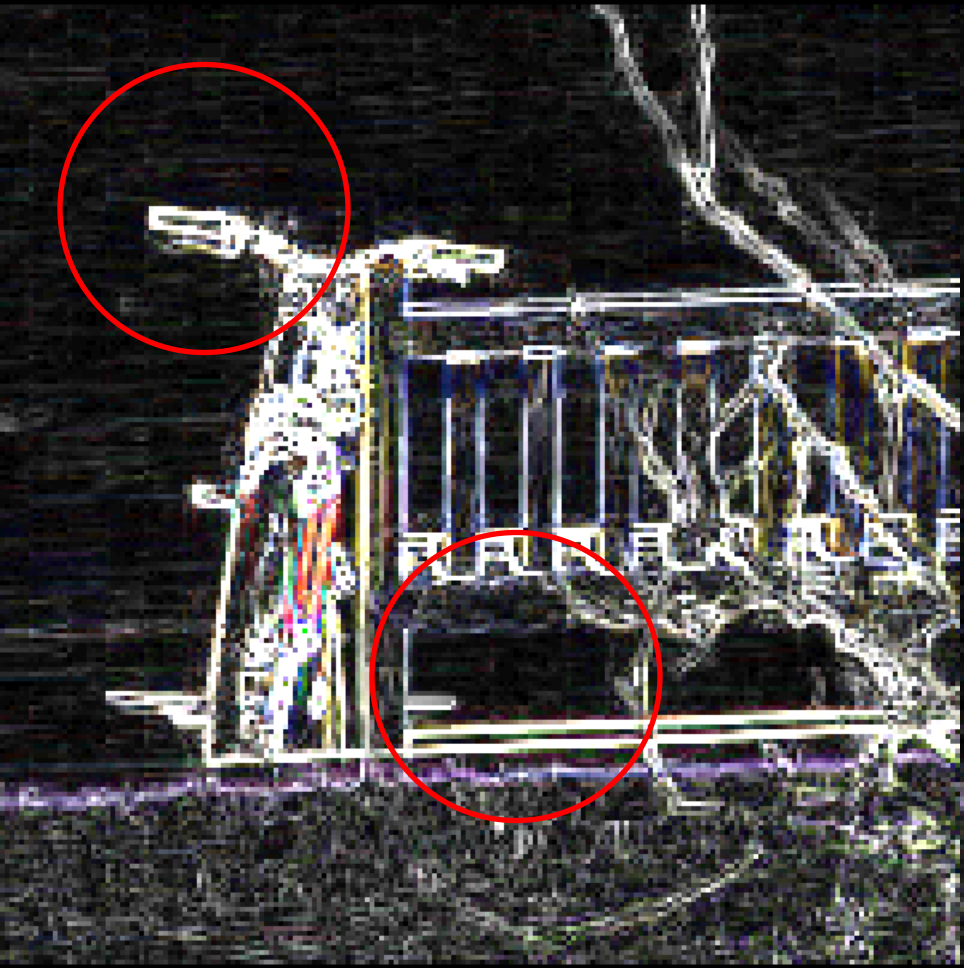
Gradient Magnitude



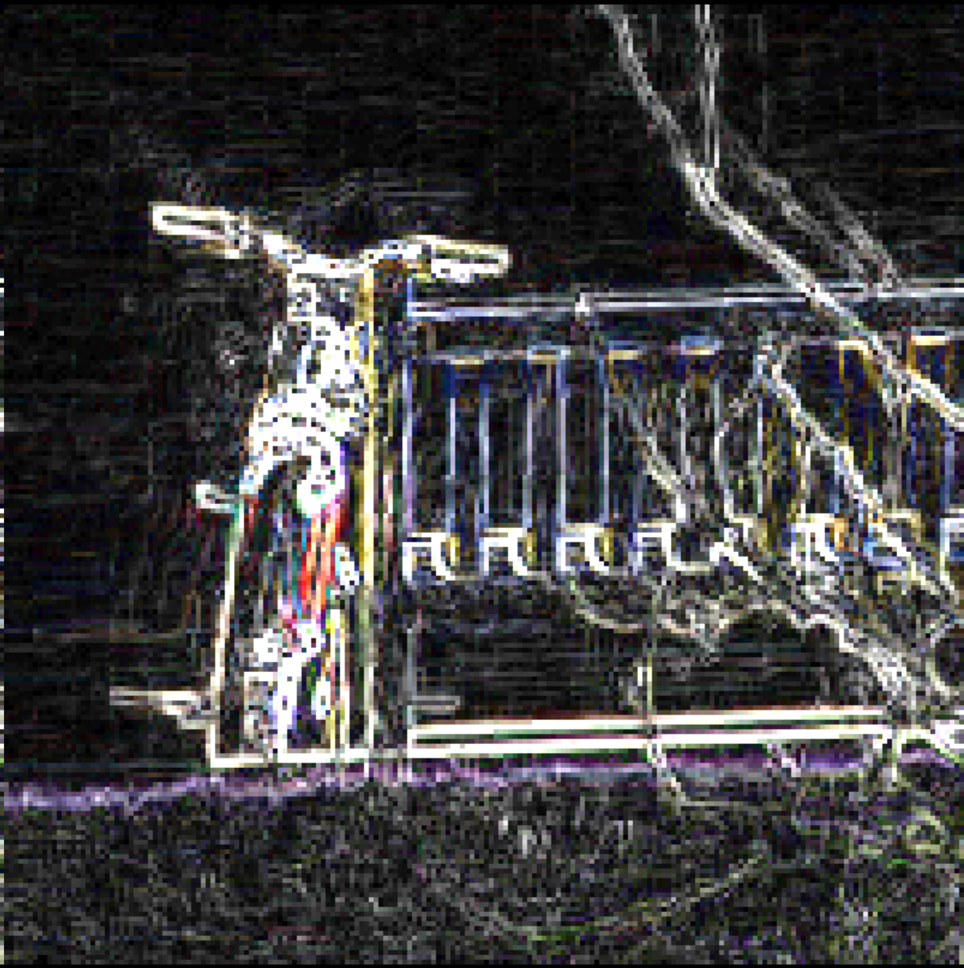
Original

Richardson Lucy Result

Gradient Magnitude



Original



Richardson Lucy Result

Let's also minimize gradients

- $Km = b$
- $D_x m = 0$
- $D_y m = 0$

- Solving this least-squares minimizes:

  $|Km-b|^2 + |D_x m|^2 + |D_y m|^2$

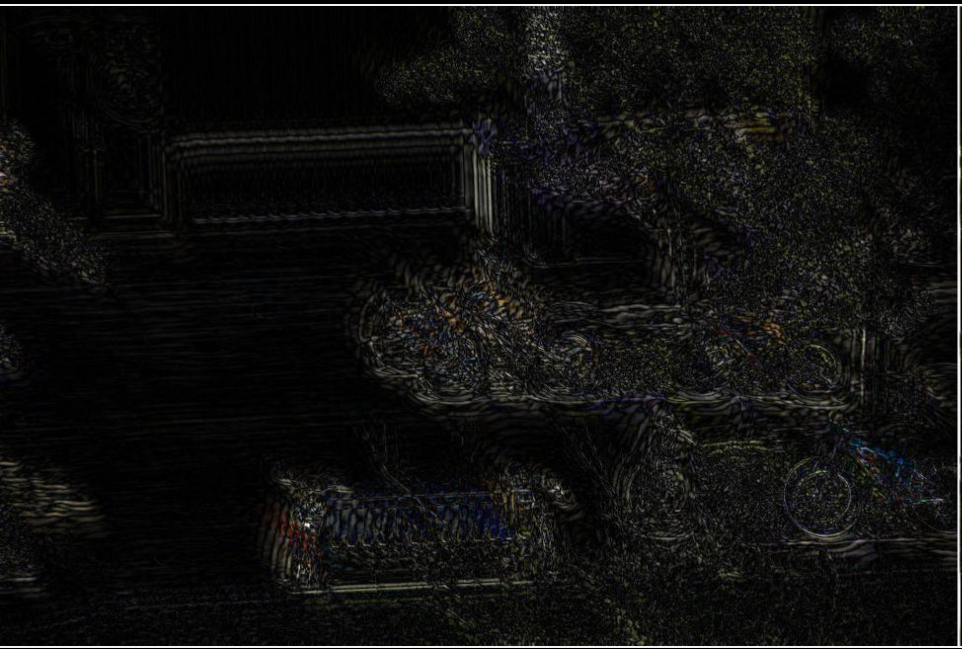
  = L2-norm of error + L2-norm of gradient field

Let m = correct answer

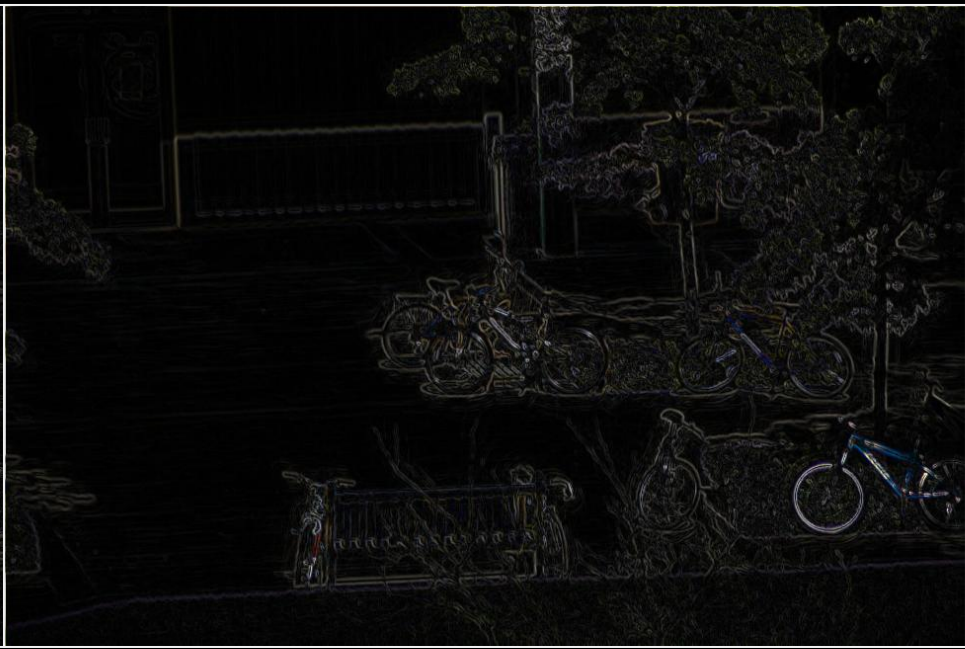
$|Km - b|^2$

$|D_x m|^2 + |D_y m|^2$

Let m = Richardson Lucy

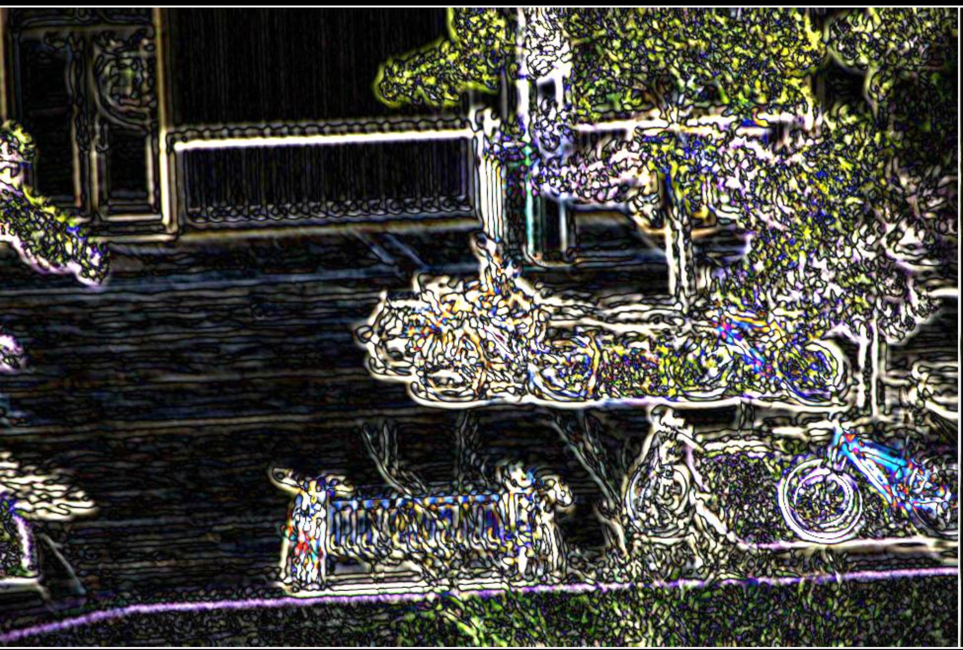


$|Km - b|^2$

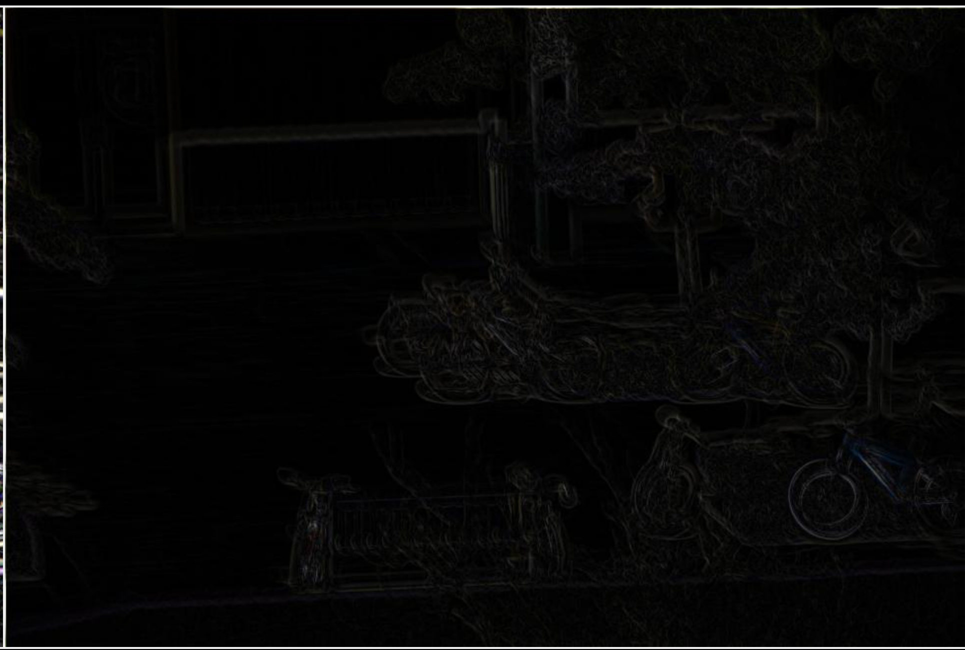


$|D_x m|^2 + |D_y m|^2$

Let m = blurry input



$|Km - b|^2$



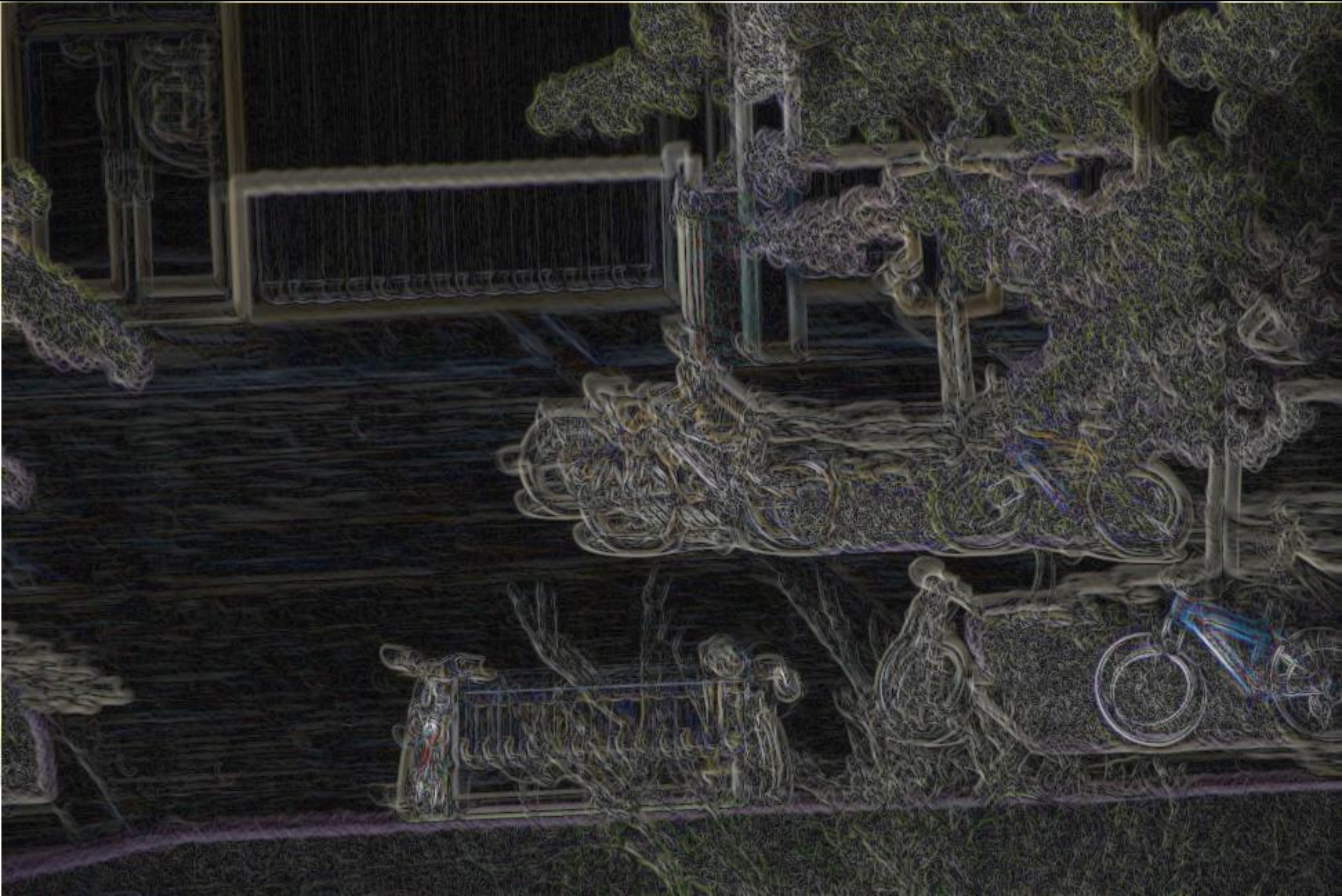
$|D_x m|^2 + |D_y m|^2$

Gradient Magnitude is a Bad Prior

- It strongly prefers blurry output if at all possible
- The prior and the error fight each other
- What’s a better prior?

Strong Gradients are Sparse

Strong Gradients are Sparse

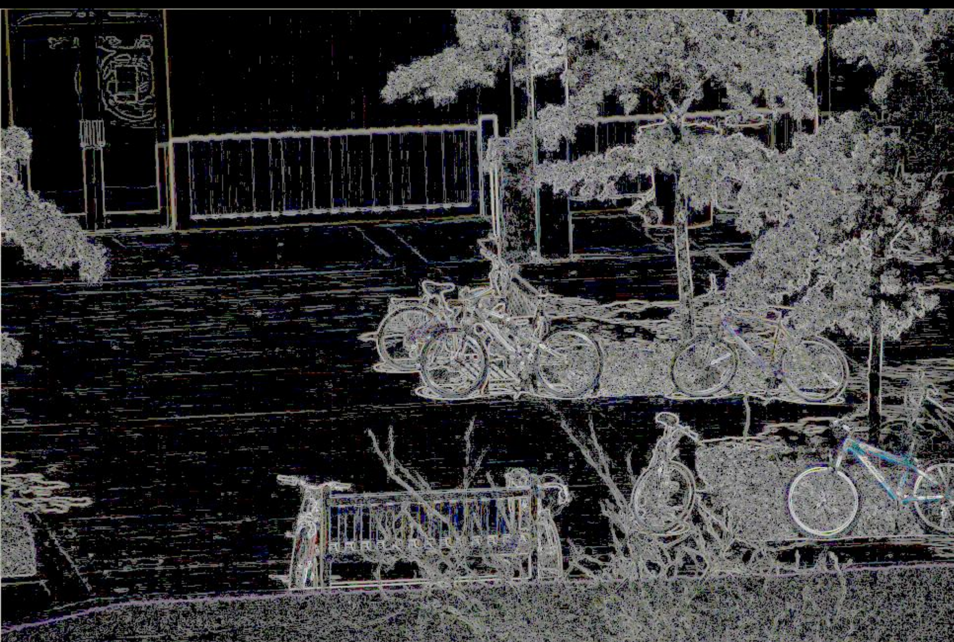


Our old prior:

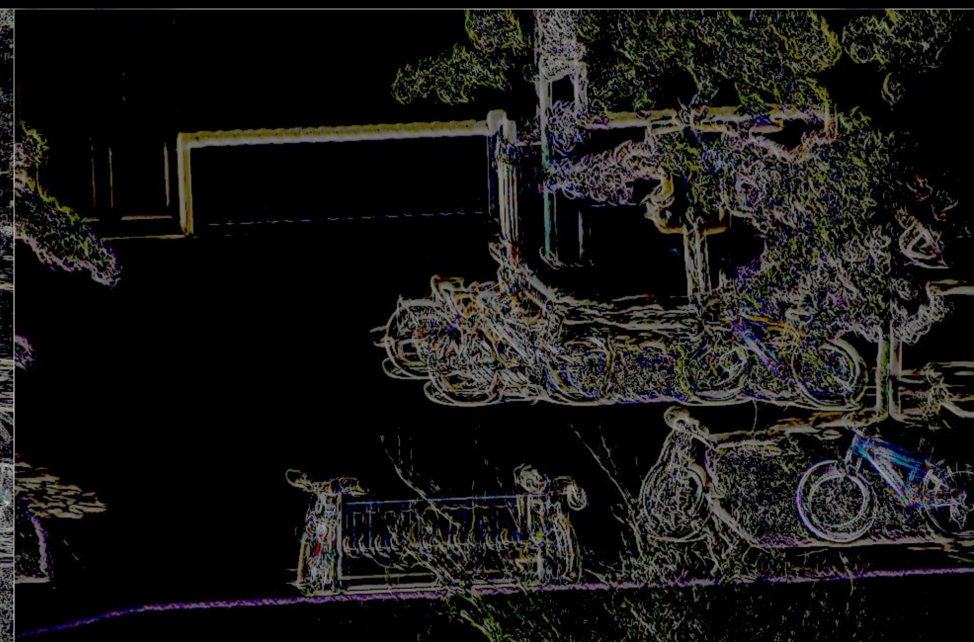
Original Grad ^ 2

Motion-Blurred Grad ^ 2

Slightly better to count the number of large edges, and minimize that



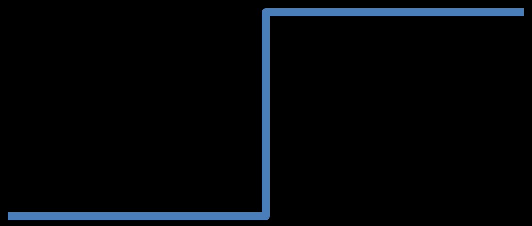
Original Grad ^ 0.125



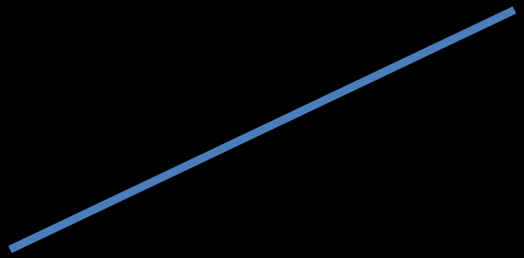
Motion-Blurred Grad ^ 0.125

Given a black-white transition...

Sum of gradients raised to power < 1 prefers sharp edges:



Sum of gradients raised to power > 1 prefers smooth edges:



Optimization

- Solving this least-squares minimizes:
  - $|Km-b|^2 + |D_x m|^2 + |D_y m|^2$
- We want to minimize something like this:
  - $|Km-b|^2 + |D_x m|^{1/2} + |D_y m|^{1/2}$
- No longer a convex optimization problem...
- Can still use gradient descent to find a local minima
  - it picks a sensible looking place for each edge

Some results

- http://graphics.ucsd.edu/~neel/dissertation/chapter5results/

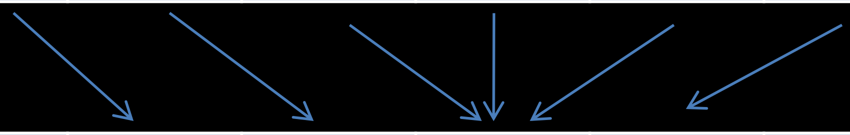
More Fun in the Gradient Domain

- So if gradients should be sparse, and we see a gradient that looks like this:

0	0	1	3	5	6	4	1	0	0

- Why not convert it to this:

0	0	1	3	5	6	4	1	0	0



0	0	0	1	3	15	1	0	0	0

More Fun in the Gradient Domain

- If it works: call it deblurring
- If it doesn't: call it a "painterly effect"


Input

Output