max planck institut
informatik

UNIVERSITÄT
DES
SAARLANDES

Probabilistic Graphical Models and Their Applications

Image Processing

Bernt Schiele

http://www.mpi-inf.mpg.de/gm

Image Processing & Stereo

- Today we shift gears and look at another problem domain: Image processing

- 4 applications of interest
  - Image denoising.
  - Image inpainting.
  - Super-resolution.
  - Stereo

- Acknowledgement
  - Majority of Slides (adapted) from **Stefan Roth @ TU Darmstadt**

Image Denoising

- Essentially all digital images exhibit image noise to some degree.
  - Even high quality images contain some noise.
  - Really noisy images may look like they are dominated by noise.

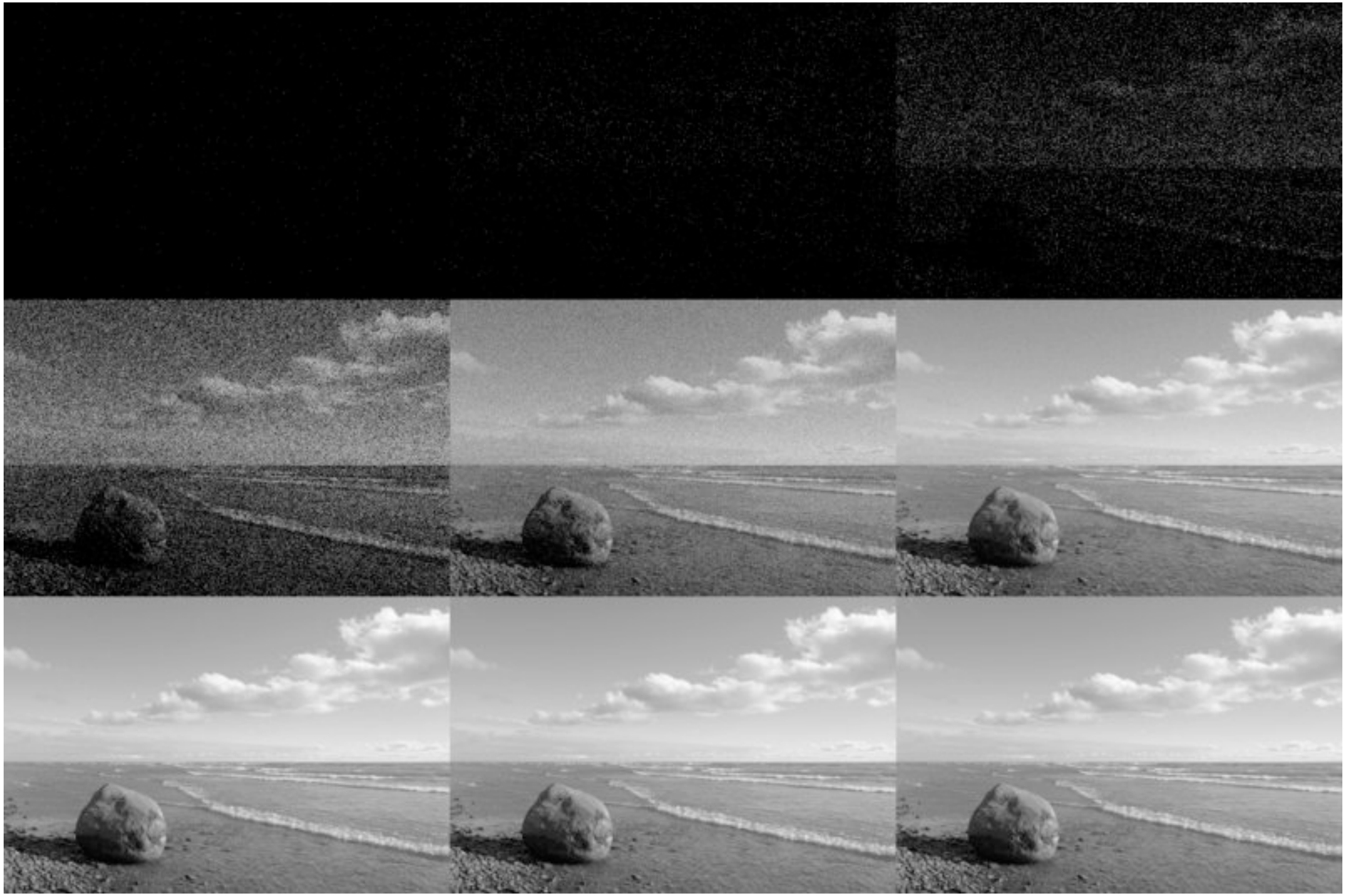
- Image noise is both:
  - Visually disturbing for the human viewer.
  - Distracting for vision algorithms.

- Image denoising aims at removing or reducing the amount of noise in the image.

Image Noise

- Image noise is a very common phenomenon that can come from a variety of sources:
  - Shot noise or photon noise due to the stochastic nature of the photons arriving at the sensor.
    - cameras actually count photons!
  - Thermal noise (“fake” photon detections).
  - Processing noise within CMOS or CCD, or in camera electronics.
  - If we use “analog” film, there are physical particles of a finite size that cause noise in a digital scan.

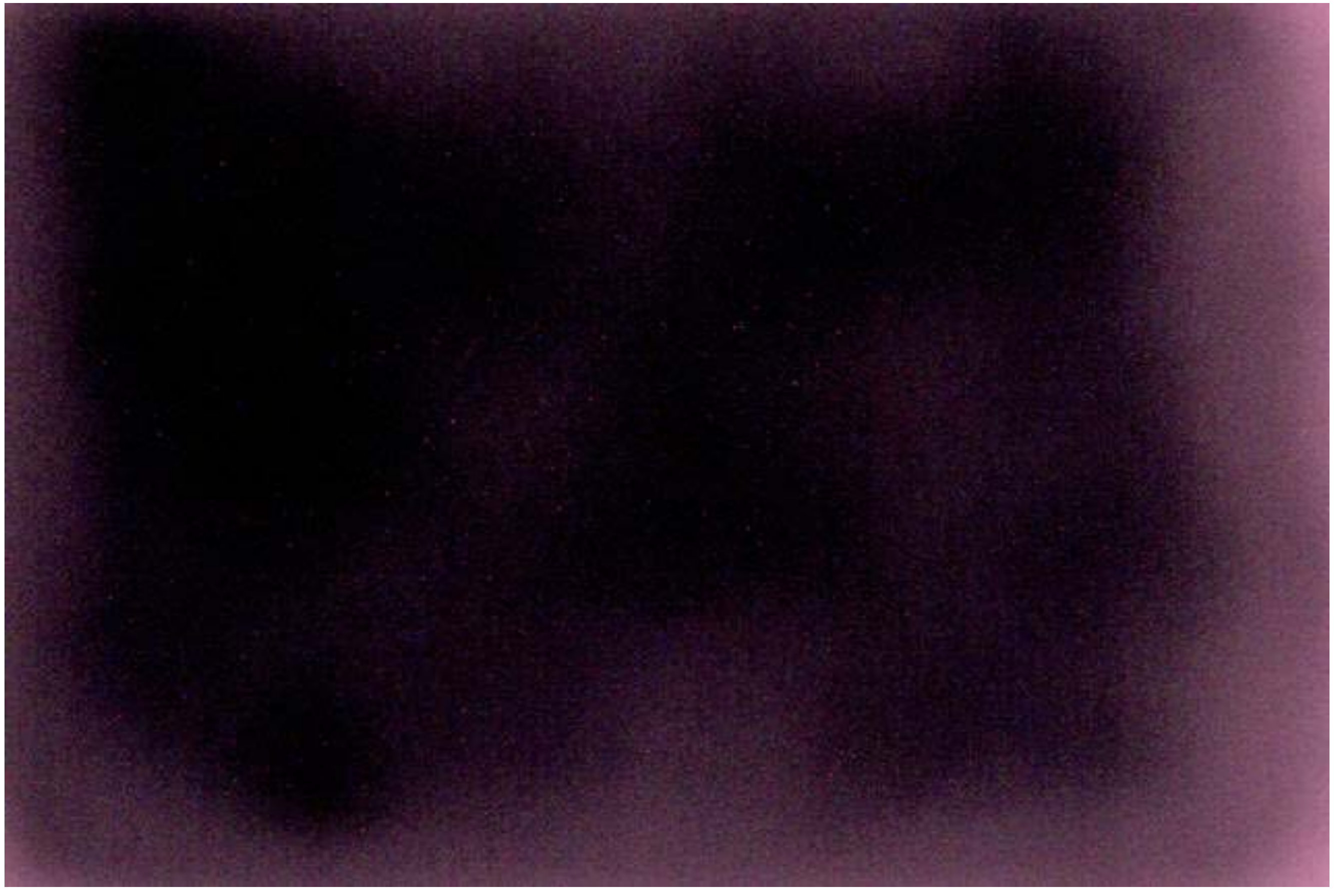
Shot Noise



Simulated shot noise (Poisson process)

From Wikipedia

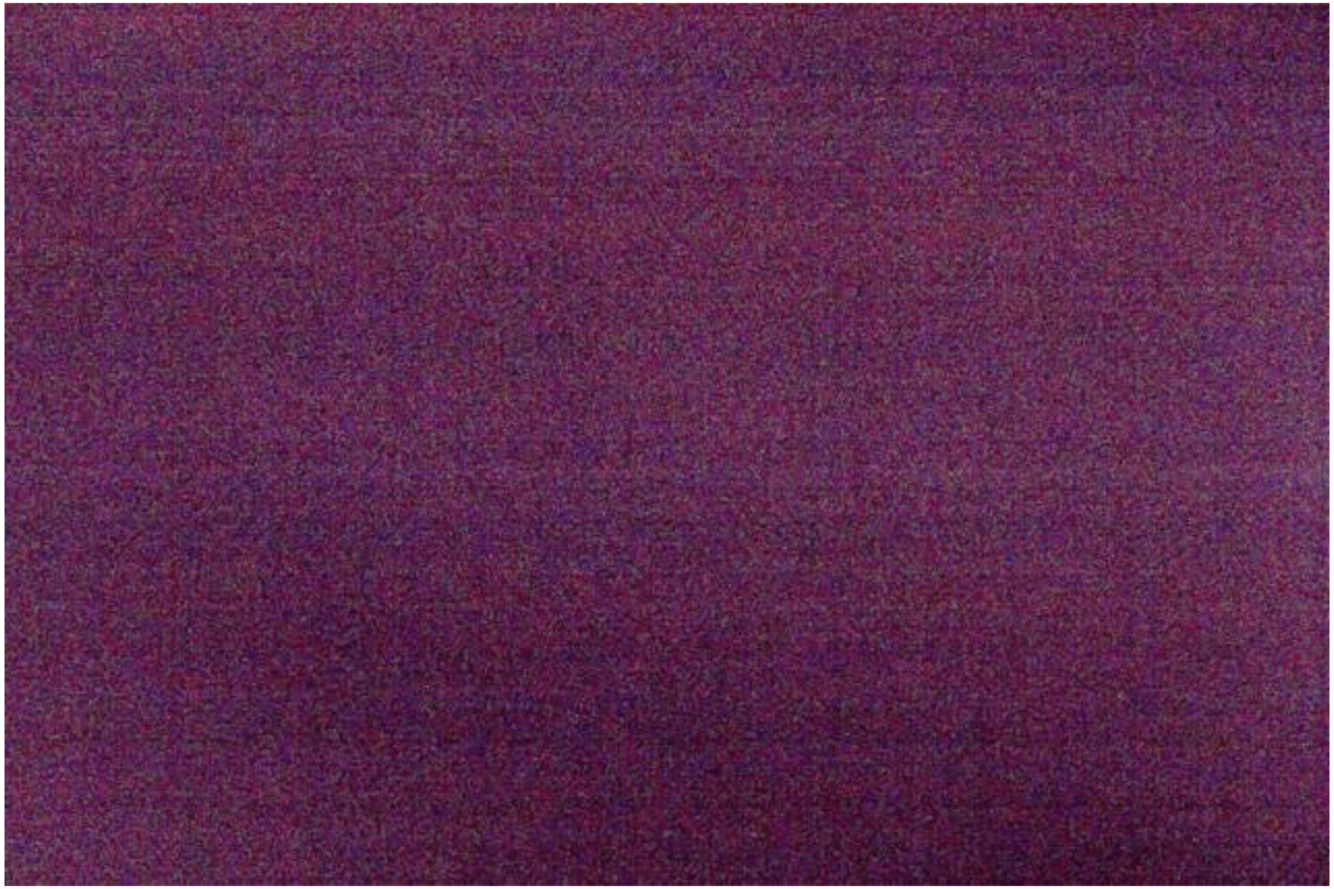
Thermal Noise or “Dark Noise”



62 minute exposure with no incident light

From Jeff Medkeff (photo.net)

Bias Noise from Amplifiers

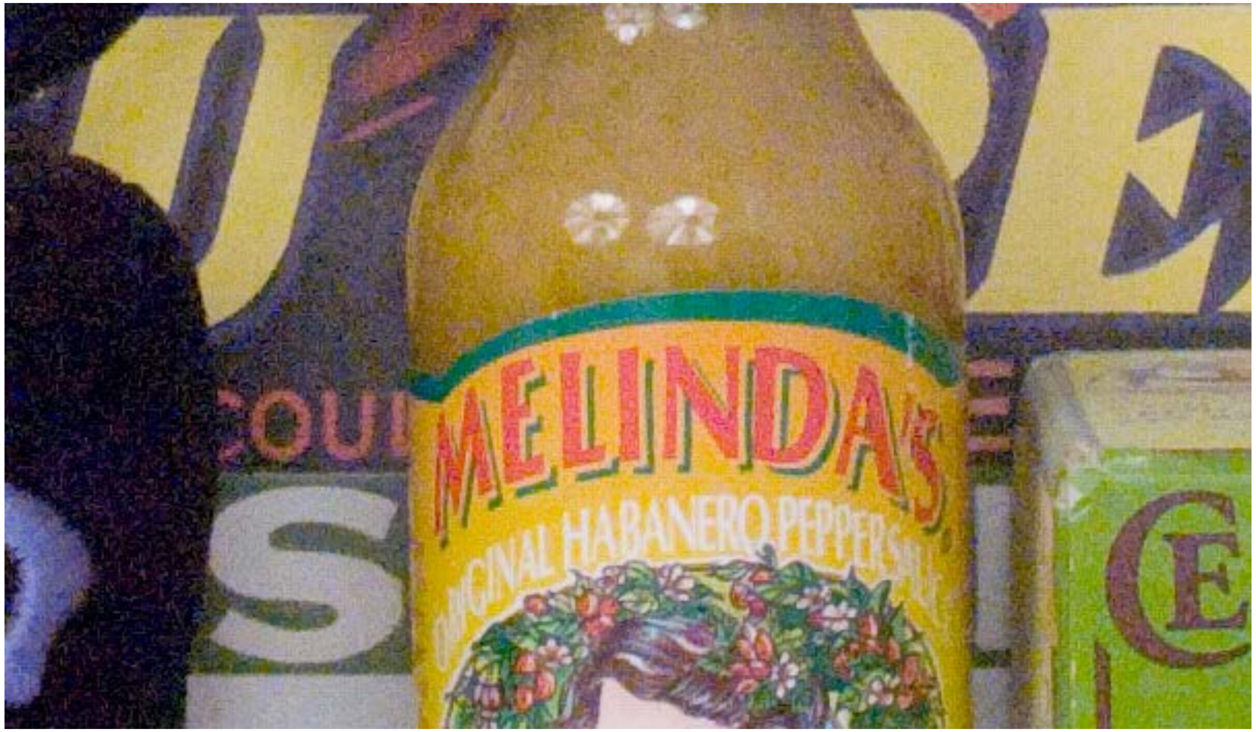


High ISO exposure (3200)

From Jeff Medkeff (photo.net)

Noise in Real Digital Photographs

- Combination of many different noise sources:



From dcresource.com

Film Grain

- If we make a digital scan of a film, we get noise from the small silver particles on the film strip:

Das Testament des Dr. Mabuse, 1933

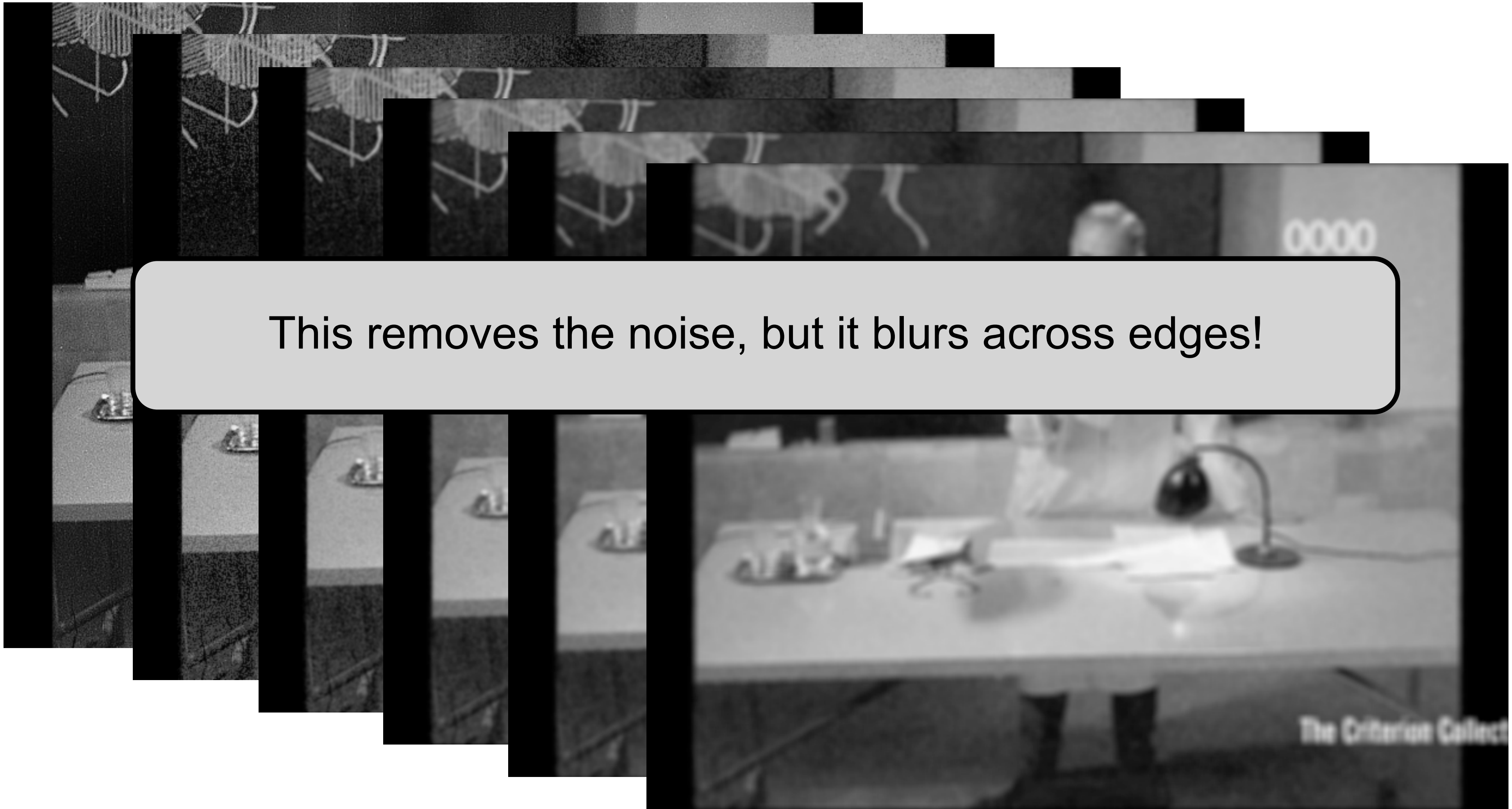
How do we remove image noise?

- Classical techniques:
  - Linear filtering, e.g. Gauss filtering.
  - Median filtering
  - Wiener filtering
  - Etc.

- Modern techniques:
  - PDE-based techniques
  - Wavelet techniques
  - MRF-based techniques (application of graphical models :)
  - Etc.

Linear Filtering

- The simplest idea is to use a Gaussian filter:



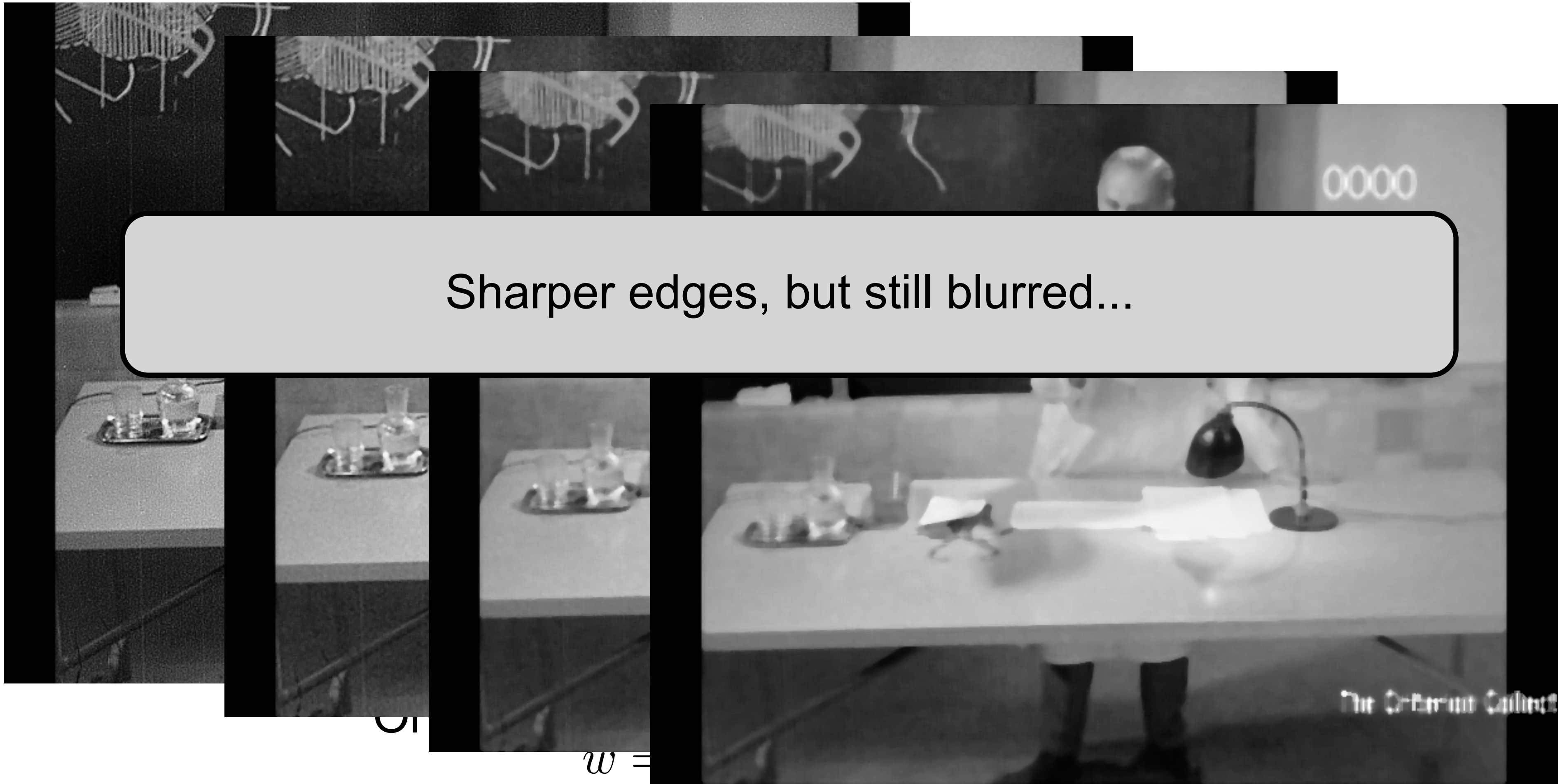
This removes the noise, but it blurs across edges!

$\sigma = 2.0$

$\sigma = 3.0$

Median Filter

- Replace each pixel by the median of the pixel values in a window around it:



Sharper edges, but still blurred...

$w = [5, 5]$

$w = [7, 7]$

How do we improve on this?

- We need denoising techniques that better preserve boundaries in images and do not blur across them.

- There is a whole host of modern denoising techniques:
  - We would need a whole semester to go through the important ones in detail.
  - So we will do the area of denoising some major injustice and restrict ourselves to what we can easily understand with what we have learned so far.

Denoising as Probabilistic Inference

- We formulate the problem of image denoising in a Bayesian fashion as a problem of probabilistic inference.
- For that we model denoising using a suitable posterior distribution:

$$p(\text{true image}|\text{noisy image}) = p(\mathbf{T}|\mathbf{N})$$

- Idea:
  - derive a graphical model that models this posterior appropriately
  - use standard inference techniques (such as sum-product rule or max-product rule) to estimate the true image that we want to recover

Modeling the Posterior

- For this, we can apply Bayes' rule and obtain:

likelihood of noisy given true image (observation model)

image prior for all true images

$$p(\mathbf{T}|\mathbf{N}) = \frac{p(\mathbf{N}|\mathbf{T}) \cdot p(\mathbf{T})}{p(\mathbf{N})}$$

posterior

normalization term (constant)

Modeling the Likelihood $p(\mathbf{N}|\mathbf{T})$

- The likelihood expresses a model of the observation:
  - Given the true, noise free image T, we assess how likely it is to observe a particular noisy image N.
  - If we wanted to model particular real noise phenomena, we could model the likelihood based on real physical properties of the world.
  - Here, we will simplify things and only use a simple, generic noise model.
  - Nevertheless, our formulation allows us to easily adapt the noise model without having to change everything...

Modeling the Likelihood $p(\mathbf{N}|\mathbf{T})$

- Simplification: assume that the noise at one pixel is independent of the others.

$$p(\mathbf{N}|\mathbf{T}) = \prod_{i,j} p(N_{i,j}|T_{i,j})$$

  - often reasonable assumption, for example since sites of a CCD sensor are relatively independent.

- Then we will assume that the noise at each pixel is additive and Gaussian distributed:

$$p(N_{i,j}|T_{i,j}) = \mathcal{N}(N_{i,j} - T_{i,j}|0, \sigma^2)$$

  - The variance σ^2 controls the amount of noise.

Gaussian Image Likelihood $p(\mathbf{N}|\mathbf{T})$

- We can thus write the Gaussian image likelihood as:

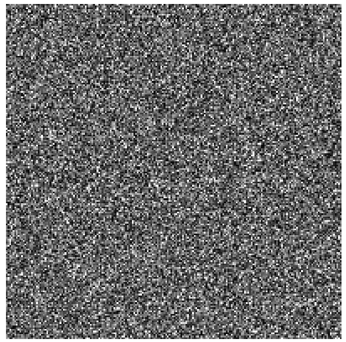
$$p(\mathbf{N}|\mathbf{T}) = \prod_{i,j} \mathcal{N}(N_{i,j} - T_{i,j}|0, \sigma^2)$$

- While this may seem a bit hacky, it works well in many applications.
  - It is suboptimal when the noise is not really independent, such as in some high definition (HD) images.
  - It also is suboptimal when the noise is non-additive, or not really Gaussian, for example as with film grain noise.

Modeling the Prior $p(\mathbf{T})$

- How do we model the prior distribution of true images?
- What does that even mean?
  - We want the prior to describe how probable it is (a-priori) to have a particular true image among the set of all possible images.

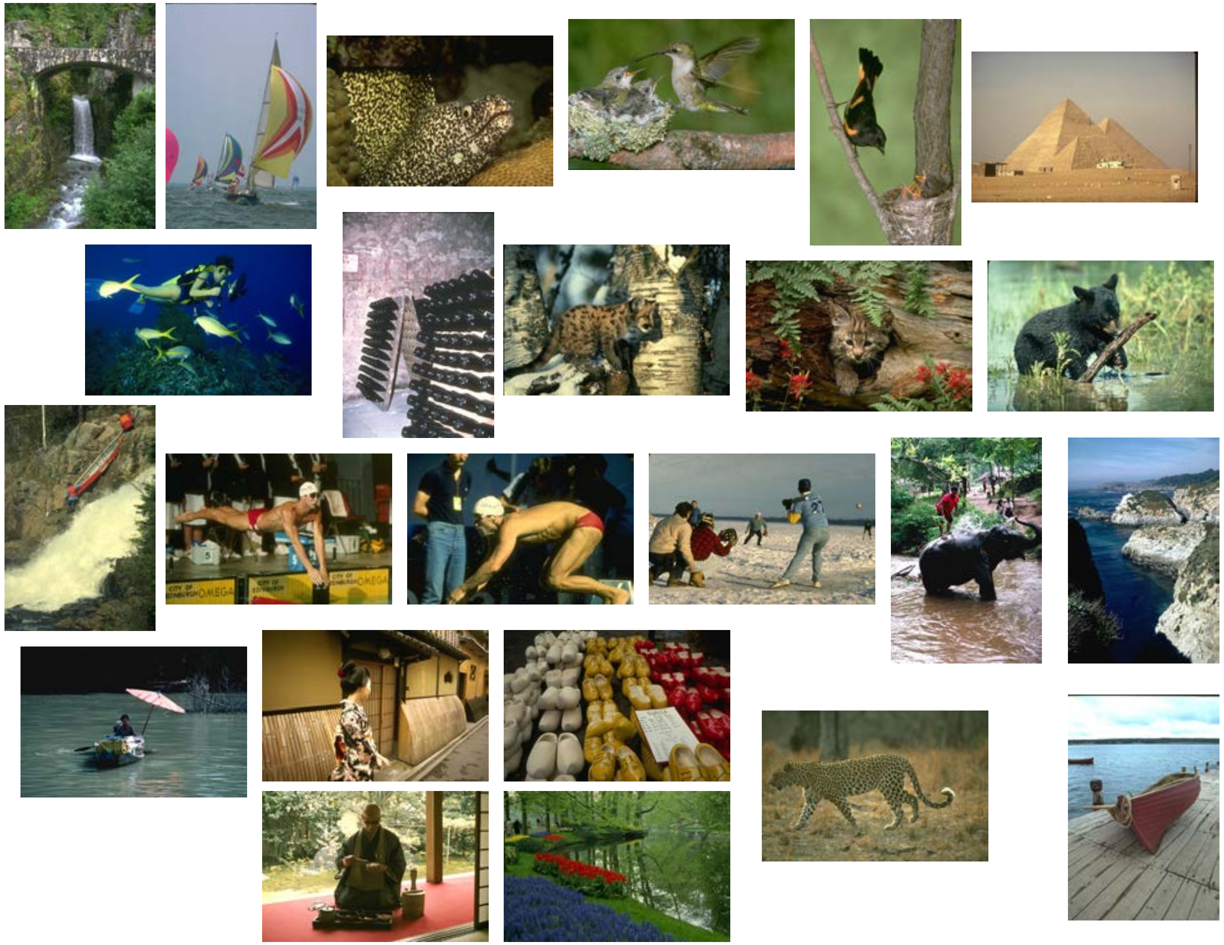
probable



improbable

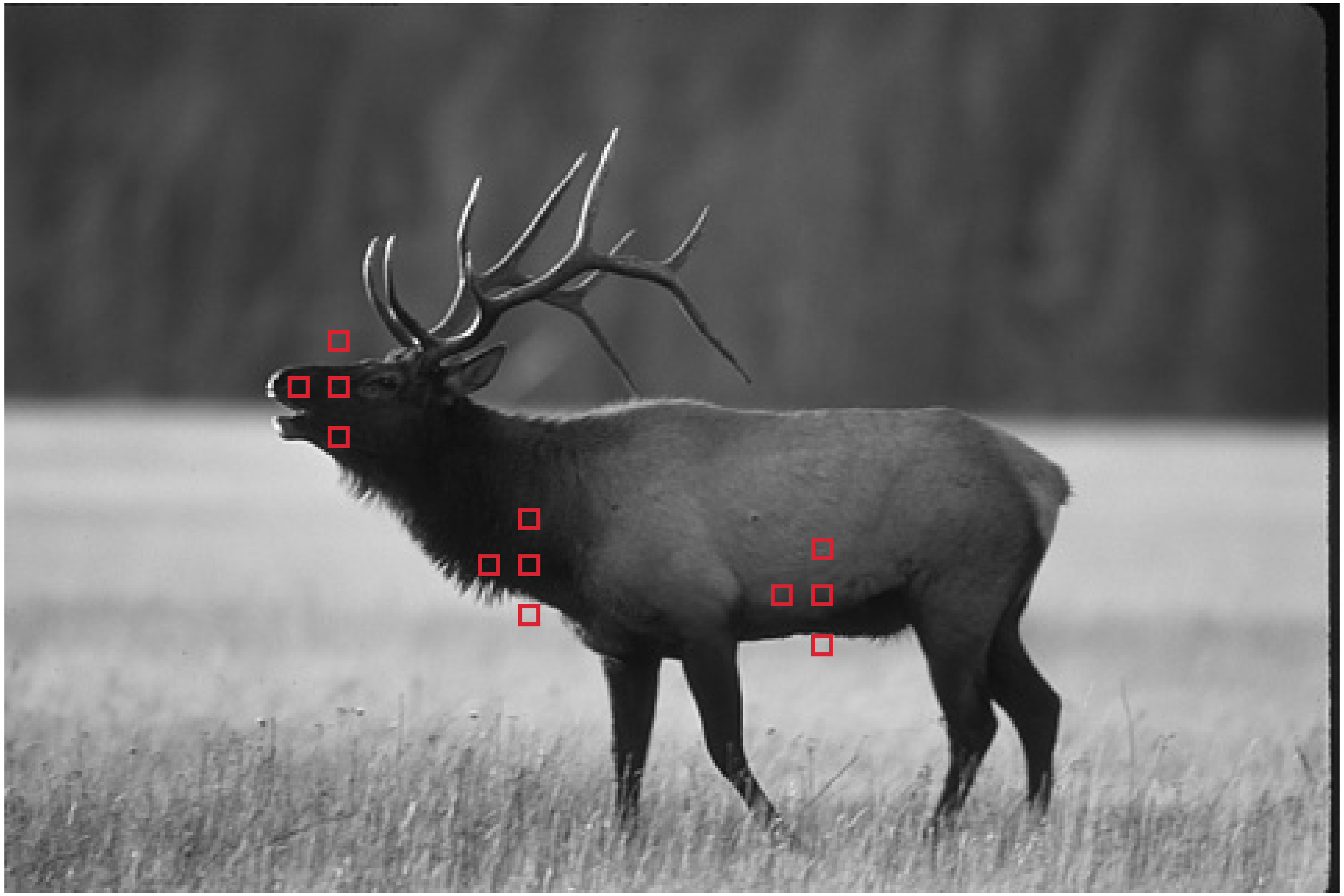
Natural Images

- What distinguishes “natural” images from “fake” ones?
  - We can take a large database of natural images and study them.



Simple Observation

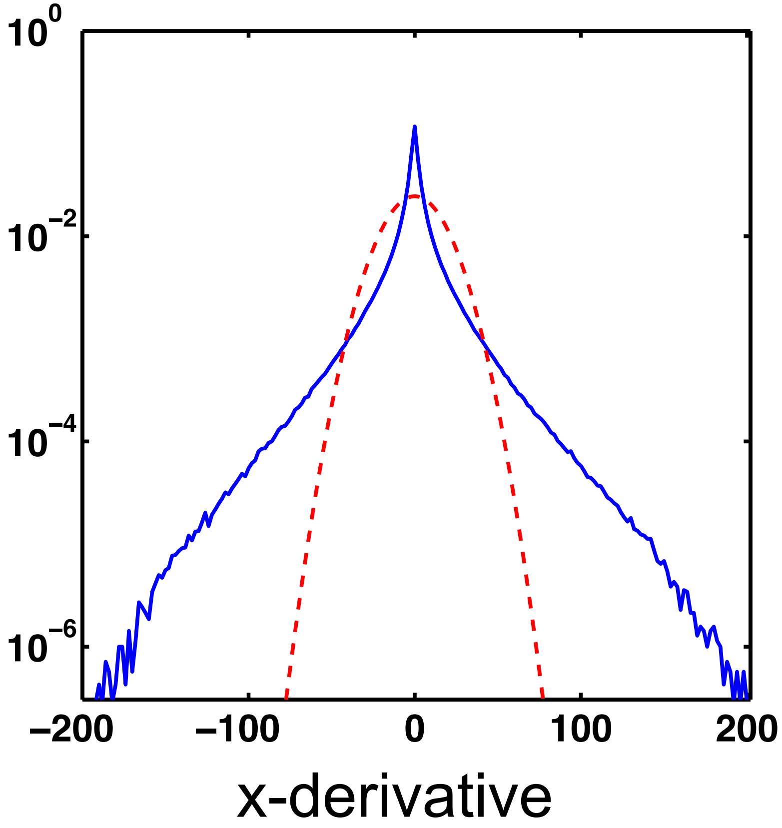
- Nearby pixels often have similar intensity:

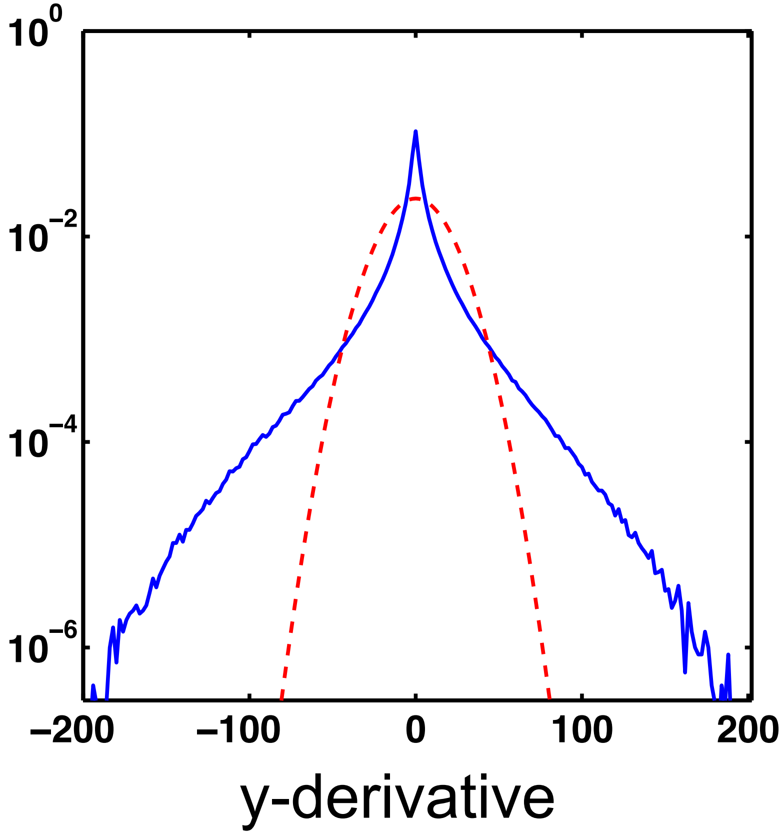


- But sometimes there are large intensity changes.

Statistics of Natural Images

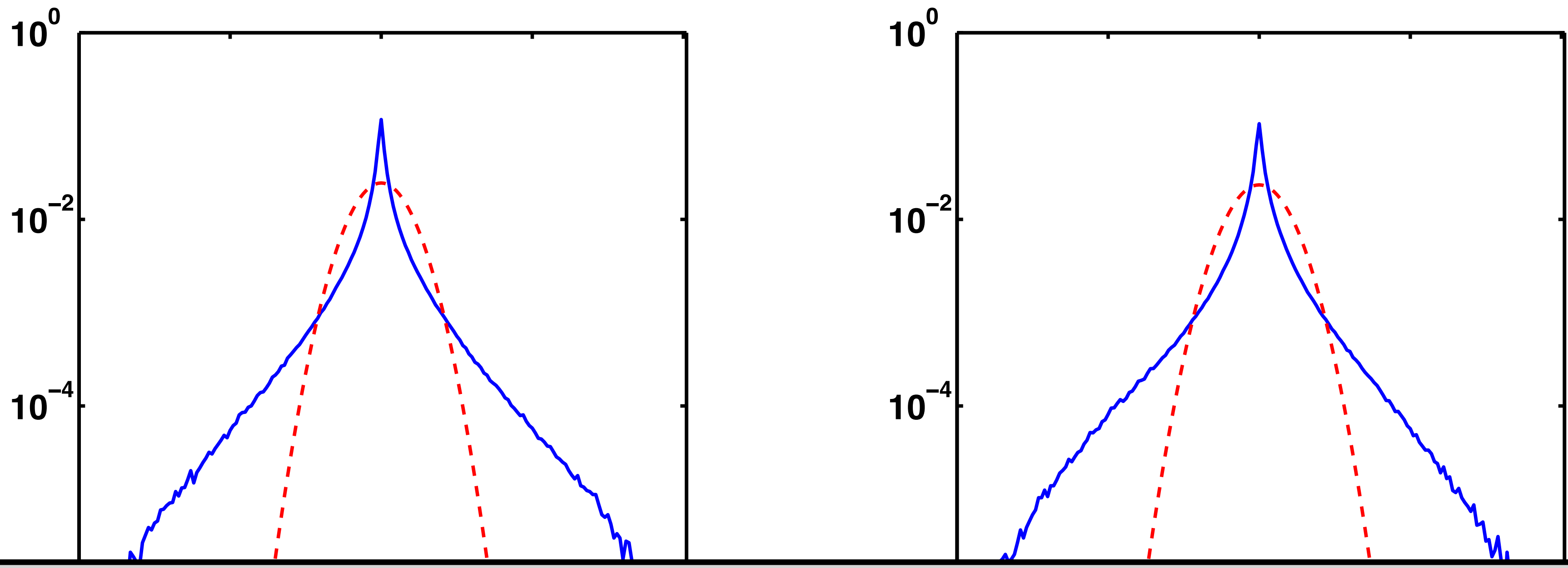
- Compute the image derivative of all images in an image database and plot a histogram:





Statistics of Natural Images

- Compute the image derivative of all images in an image database and plot a histogram:



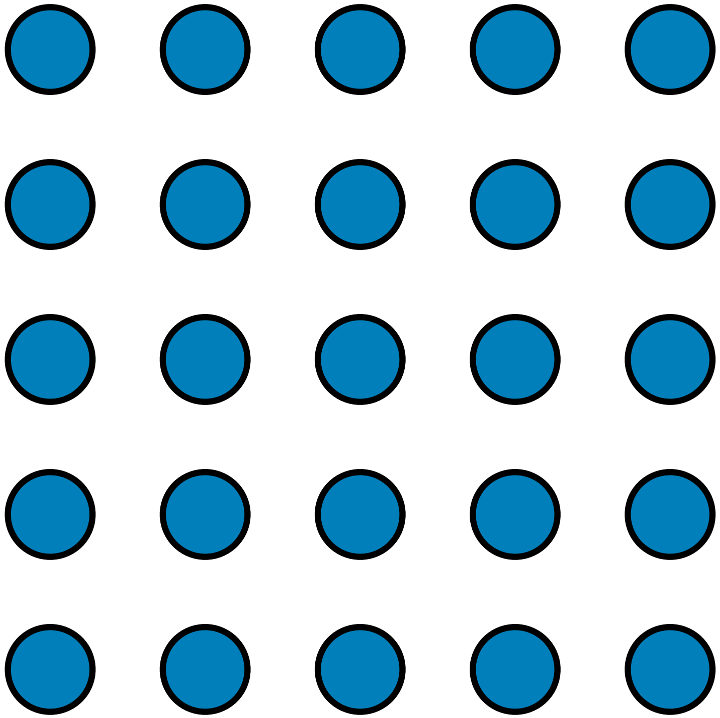
- Sharp peak at zero: Neighboring pixels most often have identical intensities.
- Heavy tails: Sometimes, there are strong intensity differences due to discontinuities in the image.

Modeling the prior $p(\mathbf{T})$

- The prior models our a-priori assumption about images
  - here we want to model the statistics of natural images
  - more specifically the local neighborhood statistics of each pixel:
    - nearby pixels have often similar intensity
    - but in the presence of boundaries the intensity difference can be large
  - let’s formulate this as a graphical model...

Modeling Compatibilities

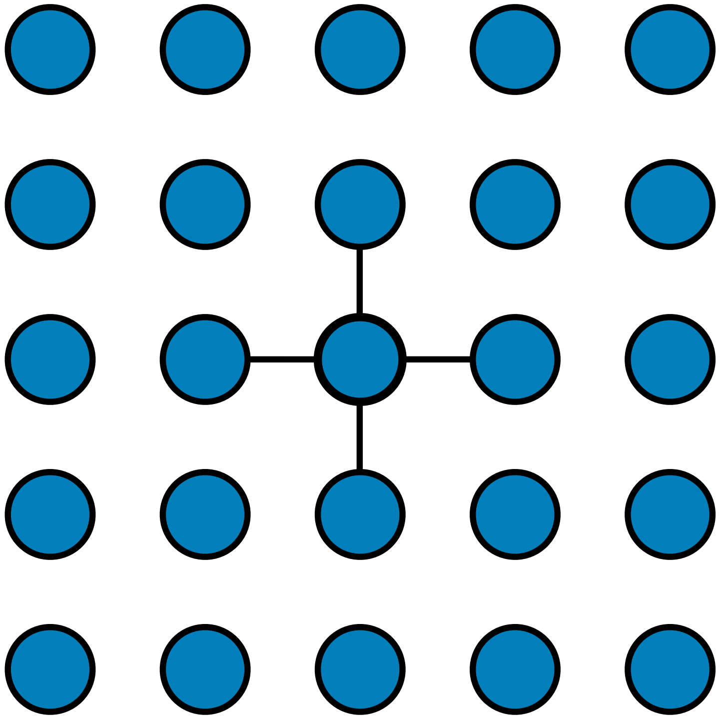
- Pixel grid:



Let's assume that we want to model how compatible or consistent a pixel is with its 4 nearest neighbors.

Modeling Compatibilities

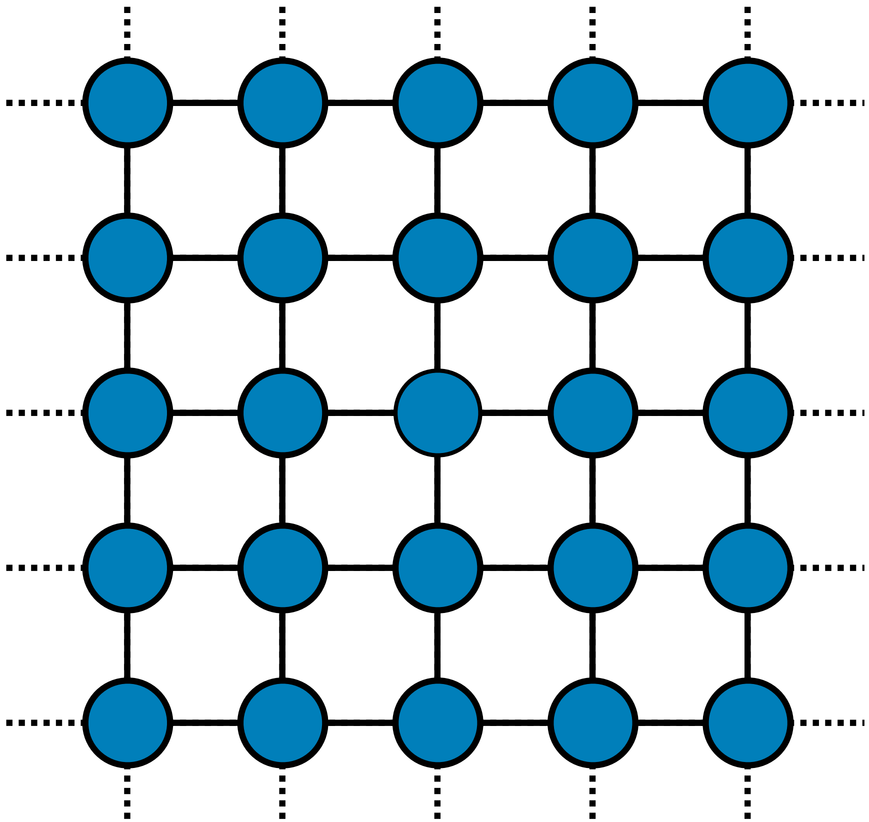
- Pixel grid (as nodes of a graph):



Denote this by drawing a line (edge) between two pixels (nodes).

Modeling Compatibilities

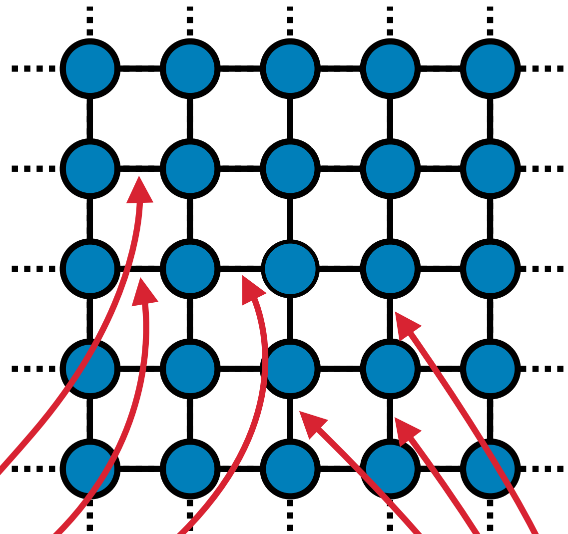
- Pixel grid (as nodes of a graph):



We do this for all pixels.

Markov Random Fields

- This is an undirected graphical model, or more specifically a Markov random field.
  - Each edge (in this particular graph) corresponds to a term in the (image) prior that models how compatible two neighboring pixels are in terms of their intensities:



compatibility of horizontal neighbors

compatibility of vertical neighbors

$$p(\mathbf{T}) = \prod_{i,j} f_H(T_{i,j}, T_{i+1,j}) \cdot f_V(T_{i,j}, T_{i,j+1})$$

product over all the pixels

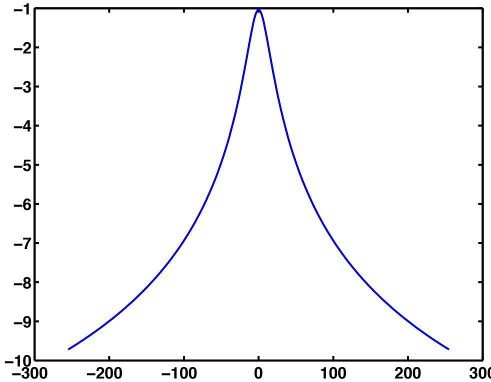
Modeling the Potentials

- What remains is to model the potentials
(or compatibility functions), e.g.:

$$f_H(T_{i,j}, T_{i+1,j})$$

- Gaussian distributions are inappropriate:
  - They do not match the statistics of natural images well.
  - They would lead to blurred discontinuities.
- We need discontinuity-preserving potentials:
  - One possibility: Student-t distribution.

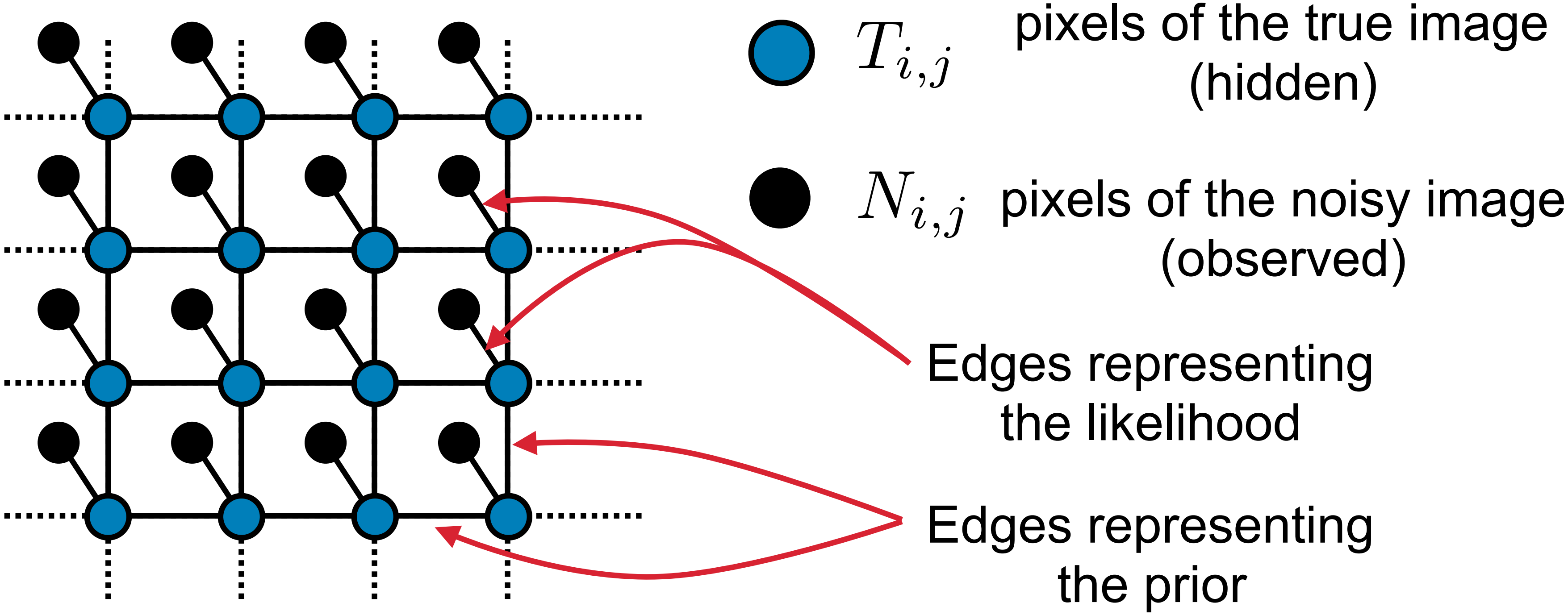
$$f_H(T_{i,j}, T_{i+1,j}) = \left(1 + \frac{1}{2\sigma^2}(T_{i,j} - T_{i+1,j})^2\right)^{-\alpha}$$



log-density

MRF Model of the (complete) Posterior

- We can now put the likelihood and the prior together in a single MRF model:



$$p(\mathbf{T}|\mathbf{N}) \propto p(\mathbf{N}|\mathbf{T})p(\mathbf{T})$$

$$= \left(\prod_{i,j} p(N_{i,j}|T_{i,j}) \right) \left(\prod_{i,j} f_H(T_{i,j}, T_{i+1,j}) \cdot f_V(T_{i,j}, T_{i,j+1}) \right)$$

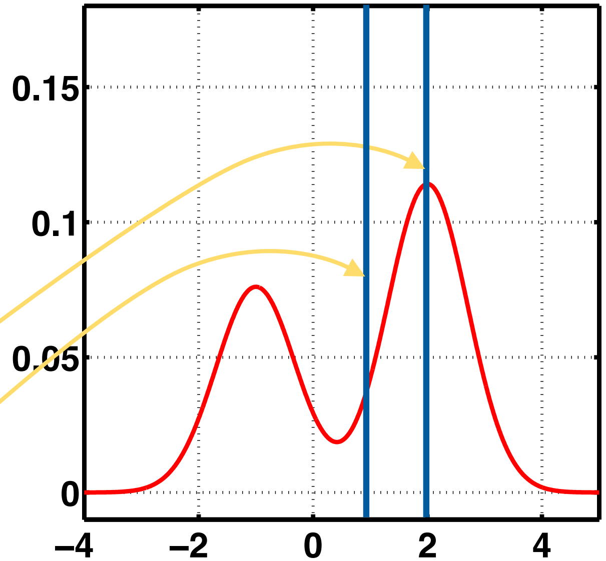
Denoising as Probabilistic Inference

- Probabilistic inference generally means one of three things
  - computing the maximum of the posterior distribution - here $p(\mathbf{T}|\mathbf{N})$ that is computing the state that is the most probable given our observations (maximum a-posteriori (MAP) estimation)
  - computing expectations over the posterior distribution, such as the mean of the posterior
  - computing marginal distributions

- Visualization of the difference between those cases:
  - assume we have the following posterior distribution: in particular - the posterior may be multi-modal

Maximum (MAP estimate)

Mean



Probabilistic Inference

- Methods that can be used for MAP estimation
  - continuous optimization methods
  - graph-based methods (graph cuts)
  - belief propagation: in particular max-product algorithm
    - however: we have a graph with cycles (=loopy) !
    - no convergence / correctness guarantees !
    - in practice “loopy belief propagation” obtains good results

- Method that can be used for expectations and marginal distributions
  - belief propagation: in particular sum-product algorithm
    - same notes as above - we have a cyclic graph - loopy belief propagation !

Denoising as Inference - Continuous Optimization

- The most straightforward idea for maximizing the posterior is to apply well-known continuous optimization techniques.

- Especially gradient techniques have found widespread use, e.g.:
  - Simple gradient ascent, also called hill-climbing.
  - Conjugate gradient methods.
  - And many more.

- Since the posterior may be multi-modal, this will give us a local optimum and not necessarily the global optimum.

Gradient Ascent

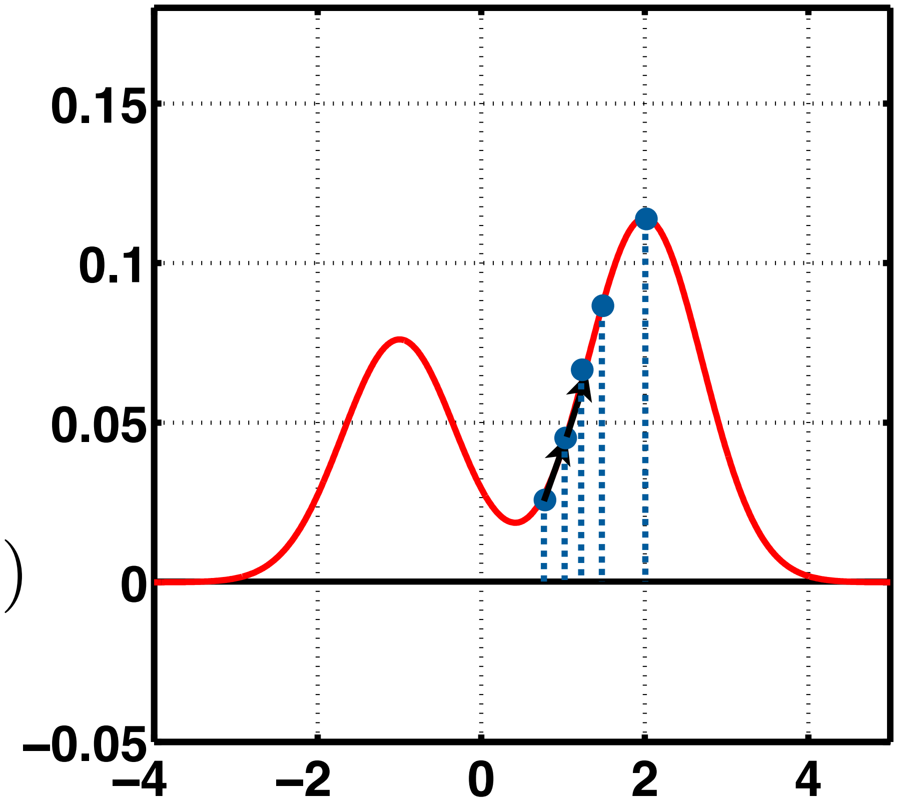
- Iteratively maximize a function $f(x)$:
  - Initialize somewhere: $x^{(0)}$
  - Compute the derivative: $\frac{\mathrm{d}}{\mathrm{d}x} f(x) = f'(x)$
  - Take a step in the direction of the derivative:

$$x^{(1)} \leftarrow x^{(0)} + \boxed{\eta} \cdot f'(x^{(0)})$$

step size

  - Repeat...

$$x^{(n+1)} \leftarrow x^{(n)} + \eta \cdot f'(x^{(n)})$$



Gradient Ascent

- We can do the same in multiple dimensions:

$$\mathbf{x}^{(n+1)} \leftarrow \mathbf{x}^{(n)} + \eta \cdot \boxed{\nabla} f(\mathbf{x}^{(n)})$$

gradient

- Issues:
  - How to initialize?
    - bad initialization with result in "wrong" local optimum
  - How to choose the step size η?
    - the wrong one can lead to instabilities or slow convergence.

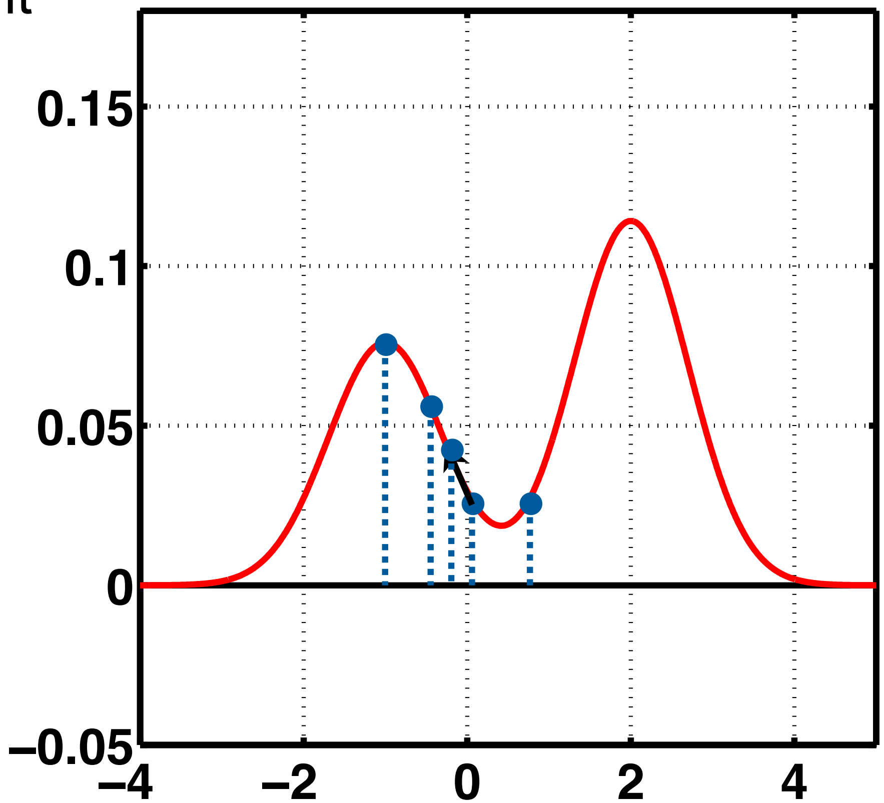


Image Denoising with Continuous Optimization

- We want to maximize the posterior:

$$p(\mathbf{T}|\mathbf{N}) \propto p(\mathbf{N}|\mathbf{T})p(\mathbf{T})$$

- equivalently we can maximize the log-posterior:
  - numerically much more stable and often more convenient

$$\log p(\mathbf{T}|\mathbf{N}) = \log p(\mathbf{N}|\mathbf{T}) + \log p(\mathbf{T}) + \text{const.}$$

- for gradient ascent we need the partial derivatives w.r.t. to a particular pixel $T_{k,l}$

$$\frac{\partial}{\partial T_{k,l}}\{\log p(\mathbf{T}|\mathbf{N})\} = \frac{\partial}{\partial T_{k,l}}\{\log p(\mathbf{N}|\mathbf{T})\} + \frac{\partial}{\partial T_{k,l}}\{\log p(\mathbf{T})\}$$

- in the following we look at the two derivatives separately
  - first - the image prior and
  - second - the likelihood.

Image Denoising with Continuous Optimization

- Let us first look at the log-prior:

$$\begin{aligned}\log p(\mathbf{T}) &= \log \left[\frac{1}{Z} \prod_{i,j} f_H(T_{i,j}, T_{i+1,j}) \cdot f_V(T_{i,j}, T_{i,j+1}) \right] \\ &= \sum_{i,j} \log f_H(T_{i,j}, T_{i+1,j}) + \log f_V(T_{i,j}, T_{i,j+1}) + \text{const}\end{aligned}$$

Gradient of the Log-Prior

- Calculate the partial derivative w.r.t. a particular pixel $T_{k,l}$:

$$\frac{\partial}{\partial T_{k,l}} \log p(\mathbf{T}) = \frac{\partial}{\partial T_{k,l}} \sum_{i,j} \log f_H(T_{i,j}, T_{i+1,j}) + \log f_V(T_{i,j}, T_{i,j+1}) + \text{const}$$

$$= \sum_{i,j} \frac{\partial}{\partial T_{k,l}} \log f_H(T_{i,j}, T_{i+1,j}) + \frac{\partial}{\partial T_{k,l}} \log f_V(T_{i,j}, T_{i,j+1})$$

- Only the 4 terms from the 4 neighbors remain:

$$\frac{\partial}{\partial T_{k,l}} \log p(\mathbf{T}) = \frac{\partial}{\partial T_{k,l}} \log f_H(T_{k,l}, T_{k+1,l}) + \frac{\partial}{\partial T_{k,l}} \log f_H(T_{k-1,l}, T_{k,l}) +$$
$$+ \frac{\partial}{\partial T_{k,l}} \log f_V(T_{k,l}, T_{k,l+1}) + \frac{\partial}{\partial T_{k,l}} \log f_V(T_{k,l-1}, T_{k,l})$$

Gradient of the Log-Prior

- Almost there... simply apply the chain rule:

$$\frac{\partial}{\partial T_{k,l}} \log f_H(T_{k,l}, T_{k+1,l}) = \frac{\frac{\partial}{\partial T_{k,l}} f_H(T_{k,l}, T_{k+1,l})}{f_H(T_{k,l}, T_{k+1,l})}$$

- last thing: calculate derivative of the compatibility function (or potential function)

$$\frac{\partial}{\partial T_{k,l}} f_H(T_{k,l}, T_{k+1,l}) = \frac{\partial}{\partial T_{k,l}} \left(1 + \frac{1}{2\sigma^2}(T_{k,l} - T_{k+1,j})^2\right)^{-\alpha}$$

Gradient of the Log-Likelihood

- Let us now look at the log-likelihood

$$\log p(\mathbf{N}|\mathbf{T}) = \log \left[\prod_{i,j} p(N_{i,j}|T_{i,j}) \right]$$
$$= \sum_{i,j} \log p(N_{i,j}|T_{i,j})$$
$$= \sum_{i,j} \log \mathcal{N}(N_{i,j} - T_{i,j}|0, \sigma^2)$$

- the partial derivative of the log-likelihood is thus simply:

$$\frac{\partial}{\partial T_{k,l}} \log p(\mathbf{N}|\mathbf{T}) = \frac{\partial}{\partial T_{k,l}} \log \left\{ \mathcal{N}(N_{k,l} - T_{k,l}|0, \sigma^2) \right\}$$

Probabilistic Inference

- Methods that can be used for MAP estimation
  - continuous optimization methods
  - graph-based methods (graph cuts)
  - belief propagation: in particular max-product algorithm
    - however: we have a graph with cycles (=loopy) !
    - no convergence / correctness guarantee !
    - in practice "loopy belief propagation" obtains good results

- Method that can be used for expectations and marginal distributions
  - belief propagation: in particular sum-product algorithm
    - same notes as above - we have a cyclic graph - loopy belief propagation !

Loopy Belief Propagation

- Empirical Observation: [Murphy,Weiss,Jordan@uai'99]
  - even for graphs with cycles: simply apply belief propagation (BP)...
  - observation: BP often gives good results even for graphs with cycles (if it converges)
  - issues
    - may not converge !
    - cycling error - old information is mistaken as new
    - convergence error - unlike in a tree, neighbors need not be independent. Loopy BP treats them as if they were
  - not really well understood under which conditions BP works well for cyclic graphs...

Loopy Belief Propagation

- Loopy BP for Image Denoising: [Lan,Roth,Huttenlocher,Black@eccv'06]
  - different update schemes: synchronous, random, ...
  - synchronous message updates: all messages are updated simultaneously
  - checkerboard-like update: alternate updates between neighbors
  - best results (image denoising) with random updates: at each step, messages are updated with fixed probability
- Some Results from the above paper:

original image

noisy image

result from image denoising with loopy BP with different potentials (left: Student t-distribution)

Denoising Results

original image

noisy image,
σ=20

PSNR 22.49dB
SSIM 0.528

denoised using
gradient ascent

PSNR 27.60dB
SSIM 0.810

Denoising Results

- Very sharp discontinuities. No blurring across boundaries.
- Noise is removed quite well nonetheless.

Denoising Results

- Because the noisy image is based on synthetic noise, we can measure the performance:
  - PSNR: Peak signal-to-noise ratio $PSNR = 20 \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right)$
  - SSIM [Wang et al., 04]: Perceptual similarity
    - Gives an estimate of how humans would assess the quality of the image.

original image

noisy image,
σ=20

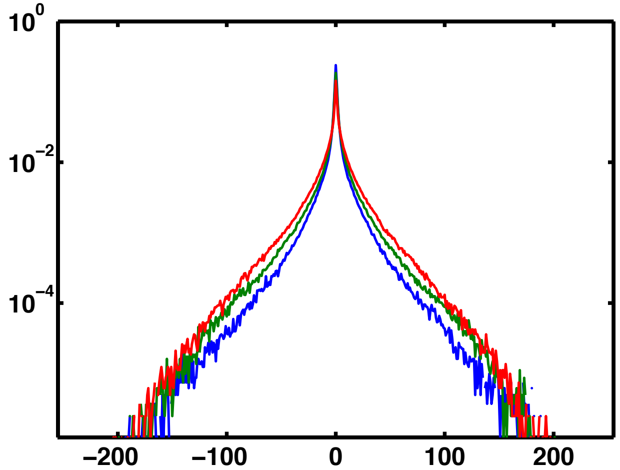
PSNR 22.49dB
SSIM 0.528

denoised using
gradient ascent

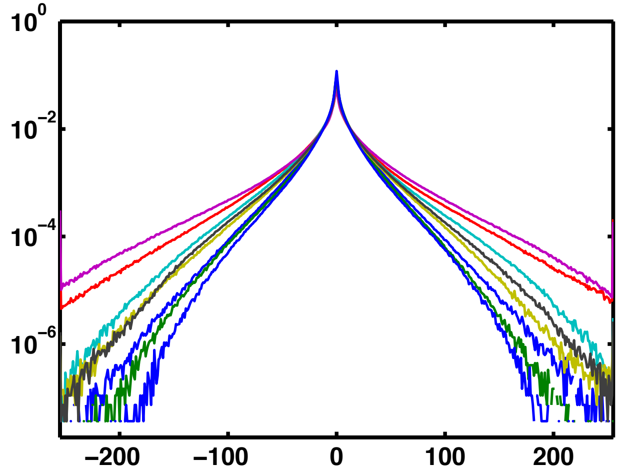
PSNR 27.60dB
SSIM 0.810

Is this the end of the story?

- No, natural images have many complex properties that we have not modeled.
  - For example, they have complex structural properties that are not modeled with a simple MRF based on a 4-connected grid.
  - Natural images have scale-invariant statistics, our model does not.
  - Responses to random linear filters are heavy-tailed.
  - Etc.



Derivative histogram on 4 spatial scales



Histograms of random (zero-mean) linear filters

Image Processing & Stereo

- Today we shift gears and look at another problem domain: Image processing

- 4 applications of interest
  - Image denoising.
  - Image inpainting.
  - Super-resolution.
  - Stereo

- Acknowledgement
  - Majority of Slides (adapted) from Stefan Roth @ TU Darmstadt

Image Inpainting

- In image inpainting the goal in to fill in a “missing” part of an image:
  - Restoration of old photographs, e.g. scratch removal...

old photograph

user-supplied mask

[Bertalmio et al., 2000]

Image Inpainting

- There are many different ways to do inpainting:
  - PDEs: [Bertalmio et al, 2000], ...
  - Exemplar-based: [Criminisi et al., 2003], ...
  - And many more.

- But, we can also apply what we already know:
  - We model the problem in a Bayesian fashion, where we regard the inpainted image as the true image. We are given the corrupted image with missing pixels.

$$p(\text{true image}|\text{corrupted image}) = p(\mathbf{T}|\mathbf{C})$$

  - Then we apply probabilistic inference...

Image Inpainting

- We apply Bayes' rule:

$$p(\mathbf{T}|\mathbf{C}) = \frac{p(\mathbf{C}|\mathbf{T}) \cdot p(\mathbf{T})}{p(\mathbf{C})}$$

  - I know this may be boring, but this general approach really is this versatile...

- Modeling the prior:
  - Important observation: This is the very same prior that we use for denoising!
    - We can re-use the prior model from denoising here.
  - Once we have a good probabilistic model of images,
    we can use it in many applications !

Inpainting Likelihood

- Again, we assume independence of the pixels:

$$p(\mathbf{C}|\mathbf{T}) = \prod_{i,j} p(C_{ij}|T_{ij})$$

- Desiderata:
  - We want to keep all known pixels fixed.
  - For all unknown pixels all intensities should be equally likely.

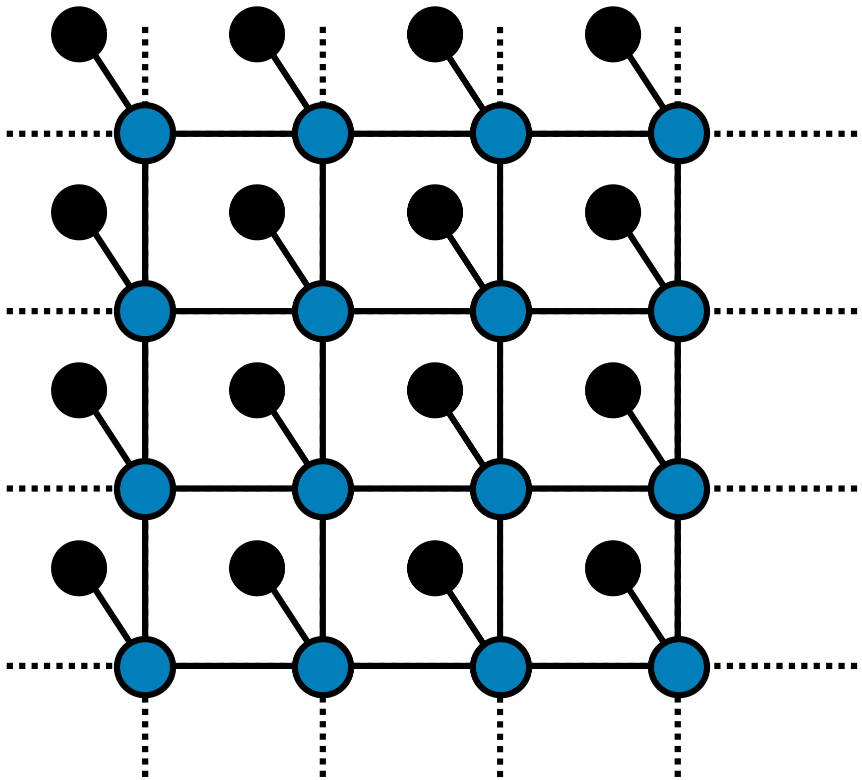
- Simple likelihood model:

$$p(C_{ij}|T_{ij}) = \begin{cases} \text{const}, & C_{ij} \text{ is corrupted} \\ \delta(T_{ij} - C_{ij}), & C_{ij} \text{ is not corrupted} \end{cases}$$

MRF Model of the Posterior

- The posterior for inpainting has the same graphical model structure as the one for denoising.

- Nonetheless, the potentials representing the likelihood are different.



Inpainting Results

“Corrupted” image (artificially corrupted for benchmarking)

Inpainted image obtained using gradient ascent

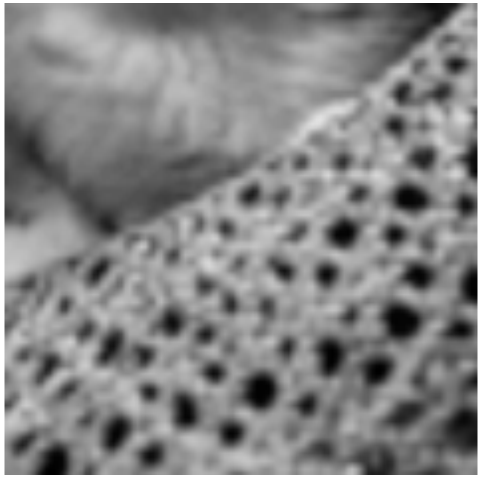
Other Inpainting Results

From [Bertalmio et al., 2000]

Image Super-Resolution

- In super-resolution, the goal is to increase the resolution of an image, while making the high-resolution image seem sharp and natural.

64x64 original image



128x128 super-resolved image (bicubic interpolation)

From [Tappen et al., 2003]

- Many applications:
  - Resizing of old, low-res digital imagery.
  - Improving resolution of images from cheap cameras.
  - Up-conversion of standard definition TV and movie footage to high-definition (720p or 1080p).

Image Super-Resolution

- The story parallels what we have seen before.
- There are many special purpose super-resolution techniques:
  - The simplest ones are bilinear and bicubic interpolation.
  - Example-based methods.
  - Etc.
- But once again, we can formulate the problem in a Bayesian way and super-resolve images using probabilistic inference:

$$p(\text{high res. image}|\text{low res. image}) = p(\mathbf{H}|\mathbf{L})$$

Image Super-Resolution

- Yet again apply Bayes rule:

$$p(\mathbf{H}|\mathbf{L}) = \frac{p(\mathbf{L}|\mathbf{H}) \cdot p(\mathbf{H})}{p(\mathbf{L})}$$

  - this should be getting really boring by now...

- Modeling the prior of high-resolution images:
  - again we can use the prior model that we already have!

- Modeling the likelihood:
  - The likelihood has to model how the low-resolution pixels correspond to the high-resolution pixels.
  - For simplicity, assume a fixed zooming factor of 2x.

Super-Resolution Likelihood

- If we have a zoom factor of 2x, there is a 2x2 patch of high-resolution pixels that corresponds to a single low-resolution pixel.
- Again assume conditional independence of the low-resolution pixels given the high-resolution image.
- We can thus write the likelihood as:

$$p(\mathbf{L}|\mathbf{H}) = \prod_{i,j} p(L_{i,j}|H_{2i,2j}, H_{2i+1,2j}, H_{2i,2j+1}, H_{2i+1,2j+1})$$

- We may for example assume that a low-resolution pixel is the average of the 4 high-resolution pixels plus a small amount of Gaussian noise:

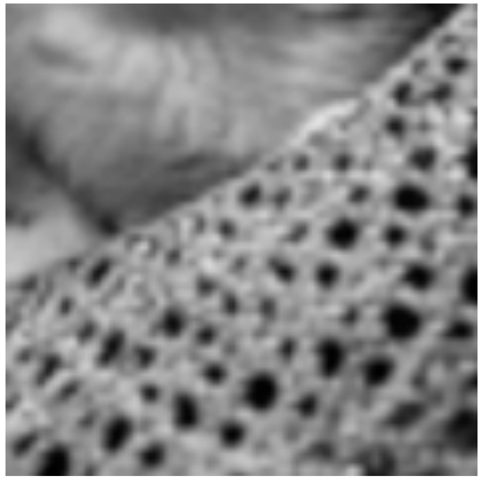
$$p(L_{i,j}|H_{2i,2j}, H_{2i+1,2j}, H_{2i,2j+1}, H_{2i+1,2j+1}) = \mathcal{N}\left(L_{i,j} - \frac{1}{4}(H_{2i,2j} + H_{2i+1,2j} + H_{2i,2j+1} + H_{2i+1,2j+1}); 0, \sigma^2\right)$$

Super-resolution Results

- Gradient ascent does not work well here, because the likelihood is more complex than for denoising or inpainting.
- But belief propagation can be made to work (with some tricks...):

original

MRF + BP



bicubic interpolation

From [Tappen et al., 2003]

Super-resolution Results

- Gradient ascent does not work very well here, because the likelihood is more complex than in denoising or inpainting.
- But belief propagation can be made to work (with some tricks...):

original

MRF + BP

bicubic interpolation

From [Tappen et al., 2003]

Super-resolution Results

- Gradient ascent does not work very well here, because the likelihood is more complex than in denoising or inpainting.
- But belief propagation can be made to work (with some tricks...):

original

MRF + BP

bicubic interpolation

From [Tappen et al., 2003]

Summary

- Many image processing problems can be formulated as problems of probabilistic inference.
  - This is only one of many different ways of approaching these problems!

- Advantages:
  - Unified approach to many different problems, in which important components (prior) may be re-used.
  - Is is relatively easy to understand what the various parts do.
  - Good application performance, despite generality.

- Disadvantages:
  - Computationally often expensive.
  - Special purpose techniques often have somewhat better application performance.