Lecture 8—Image Relaxation: Restoration and Feature Extraction

ch. 6 of *Machine Vision* by Wesley E. Snyder & Hairong Qi

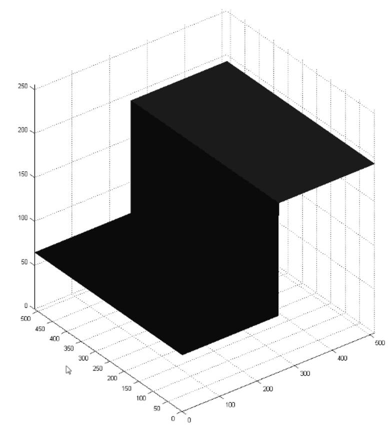
Spring 2024

16-725 (CMU RI) : BioE 2630 (Pitt)

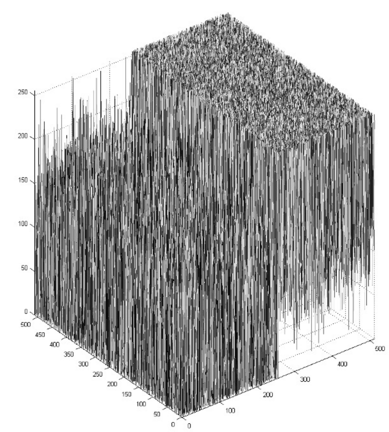
Dr. John Galeotti

All images are degraded

- Remember, all measured images are degraded
  - Noise (always)
  - Distortion = Blur (usually)
- False edges
  - From noise
- Unnoticed/Missed edges
  - From noise + blur



original image plot



noisy image plot

We need an "un-degrader"…

- To extract "clean" features for segmentation, registration, etc.
- Restoration
  - *A-posteriori* image restoration
  - Removes degradations from images
- Feature extraction
  - Iterative image feature extraction
  - Extracts features from noisy images

Image relaxation

- The basic operation performed by:
  - Restoration
  - Feature extraction (of the type in ch. 6)
- An image *relaxation* process is a multistep algorithm with the properties that:
  - The output of a step is the same form as the input (e.g., 256^2 image to 256^2 image)
    - Allows **iteration**
  - It **converges** to a bounded result
  - The operation on any pixel is dependent only on those pixels in some well defined, finite **neighborhood** of that pixel. (optional)

Restoration: An inverse problem

- Assume:
  - An ideal image, f
  - A measured image, g
  - A distortion operation, D
  - Random noise, n
- Put it all together:

$$g = D(f) + n$$

This is what we want

This is what we get

How do we extract f ?

Restoration is ill-posed

- Even without noise
- Even if the distortion is linear blur
  - Inverting linear blur = deconvolution
- But we want restoration to be well-posed…

A well-posed problem

- $g = D(f)$ is well-posed if:
  - For each f, a solution exists,
  - The solution is unique, AND
  - The solution g continuously depends on the data f
- Otherwise, it is ill-posed
  - Usually because it has a large condition number: $K >> 1$

Condition number, K

- $K \approx \Delta$ output / Δ input
- For the linear system $b = Ax$
  - $K = ||A|| \, ||A^{-1}||$
  - $K \in [1,\infty)$

K for convolved blur

- Why is restoration ill-posed for simple blur?
- Why not just linearize a blur kernel, and then take the inverse of that matrix?
  - $F = H^{-1}G$
- Because H is probably singular
- If not, H almost certainly has a large K
  - So small amounts of noise in G will make the computed F almost meaningless
- See the book for great examples

Regularization theory to the rescue!

- How to handle an ill-posed problem?
- Find a related *well-posed* problem!
  - One whose solution approximates that of our ill-posed problem
- E.g., try minimizing:

$$E = \sum_i \left(g_i - \left(f_i \otimes h \right) \right)^2$$

  - But unless we know something about the noise, this is the exact same problem!

Digression: Statistics

- Remember Bayes' rule?

This is the *a posteriori conditional* pdf

This is the *conditional* pdf

This is the *a priori* pdf

Just a normalization constant

- $p(f \mid g) = p(g \mid f) * p(f) / p(g)$

This is what we want! It is our *discrimination* function.

Maximum a posteriori (MAP) image processing algorithms

- To find the f underlying a given g:
  1. Use Bayes' rule to "compute all" $p(f_q \mid g)$
     - $f_q \in$ (the set of all possible f)
  2. Pick the f_q with the maximum $p(f_q \mid g)$
     - $p(g)$ is "useless" here (it's constant across all f_q)

- This is equivalent to:
  - f = argmax(f_q) $p(g \mid f_q) * p(f_q)$

Noise term

Prior term

Probabilities of images

- Based on probabilities of pixels
- For each pixel i:
  - $p(f_i \mid g_i) \propto p(g_i \mid f_i) * p(f_i)$

- Let's simplify:
  - Assume no blur (just noise)
    - At this point, some people would say we are *denoising* the image.
  - $p(g \mid f) = \prod p(g_i \mid f_i)$
  - $p(f) = \prod p(f_i)$

Probabilities of pixel values

- $p(g_i \,|\, f_i)$
  - This could be the density of the noise…
  - Such as a Gaussian noise model
  - $= \text{constant} * e^{\text{something}}$
- $p(f_i)$
  - This could be a Gibbs distribution…
    - If you model your image as an ND Markov field
  - $= e^{\text{something}}$
- See the book for more details

Put the math together

- Remember, we want:
  - f = argmax(f_q) $p(g \mid f_q) * p(f_q)$
  - where $f_q \in$ (the set of all possible f)
- And remember:
  - $p(g \mid f) = \prod p(g_i \mid f_i) = \text{constant} * \prod e^{\text{something}}$
  - $p(f) = \prod p(f_i) = \prod e^{\text{something}}$
  - where $i \in$ (the set of all image pixels)
- But we like $\sum\text{something}$ better than $\prod e^{\text{something}}$, so take the log and solve for:
  - f = argmin(f_q) ($\sum p'(g_i \mid f_i) + \sum p'(f_i)$)

Objective functions

- We can re-write the previous slide's final equation to use *objective functions* for our noise and prior terms:
  - f = argmin(f_q) ($\sum p'(g_i \,|\, f_i) + \sum p'(f_i)$)

    $\Downarrow$
  - f = argmin(f_q) ($H_n(f, g) \quad + H_p(f)$)
- We can also combine these objective functions:
  - $H(f, g) = H_n(f, g) + H_p(f)$

Purpose of the objective functions

- Noise term $H_n(f, g)$:
  - If we assume independent, Gaussian noise for each pixel,
  - We tell the minimization that f should resemble g.
- Prior term (a.k.a. *regularization* term) $H_p(f)$:
  - Tells the minimization what properties the image should have
  - Often, this means brightness that is:
    - Constant in local areas
    - Discontinuous at boundaries

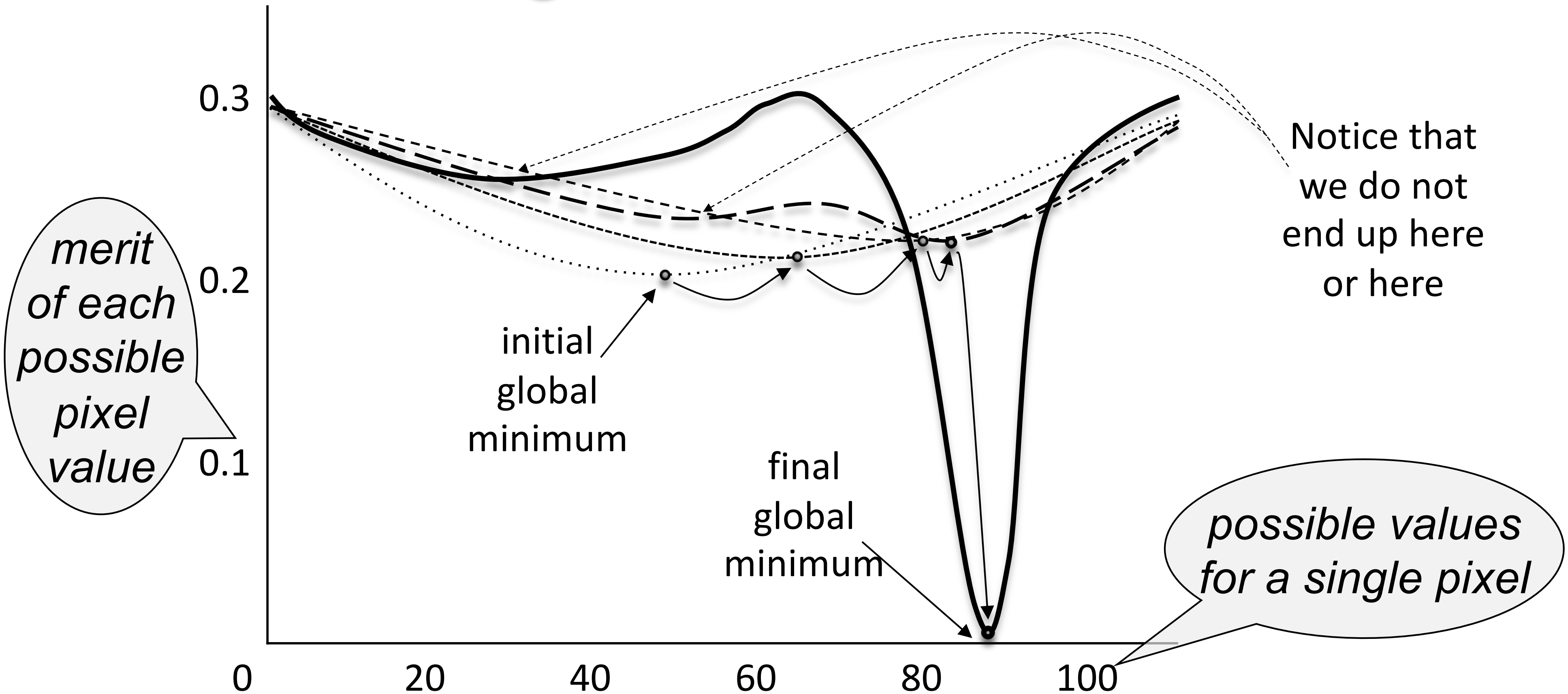
Minimization is a beast!

- Our objective function is not “nice”
  - It has many local minima
  - So gradient descent will not do well
- We need a more powerful optimizer:
- Mean field annealing (MFA)
  - Approximates simulated annealing
  - But it’s faster!
  - It’s also based on the mean field approximation of statistical mechanics

MFA

- MFA is a *continuation method*
- So it implements a *homotopy*
  - A homotopy is a continuous deformation of one hyper-surface into another
- MFA procedure:
  1. Distort our complex objective function into a convex hyper-surface (N-surface)
     - The only minima is now the global minimum
  2. Gradually distort the convex N-surface back into our objective function

MFA: Single-Pixel Visualization



Continuous deformation of a function which is initially convex to find the (near-) global minimum of a non-convex function.

Generalized objective functions for MFA

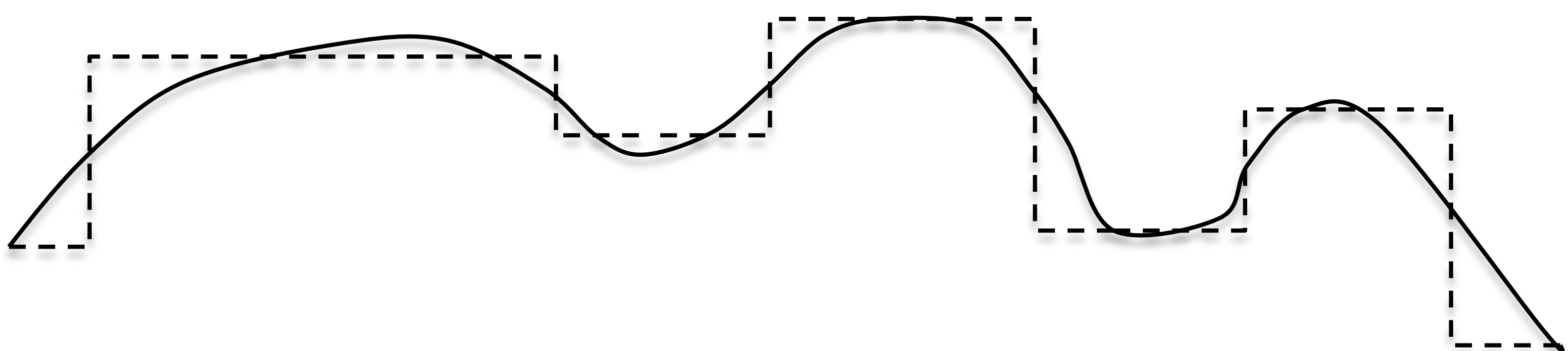
- Noise term: $\sum_i \left(\left(D(f) \right)_i - g_i \right)^2$
  - $(D(f))_i$ denotes some distortion (e.g., blur) of image f in the vicinity of pixel I
- Prior term: $-\frac{1}{\tau} \sum_i e^{-\frac{\left(R(f) \right)_i^2}{\tau^2}}$
  - τ represents a priori knowledge about the roughness of the image, which is altered in the course of MFA
  - $(R(f))_i$ denotes some function of image f at pixel i
  - The prior will seek the f which causes $R(f)$ to be zero (or as close to zero as possible)

$R(f)$: choices, choices

- Piecewise-constant images

$$R^2(f) = \left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2$$

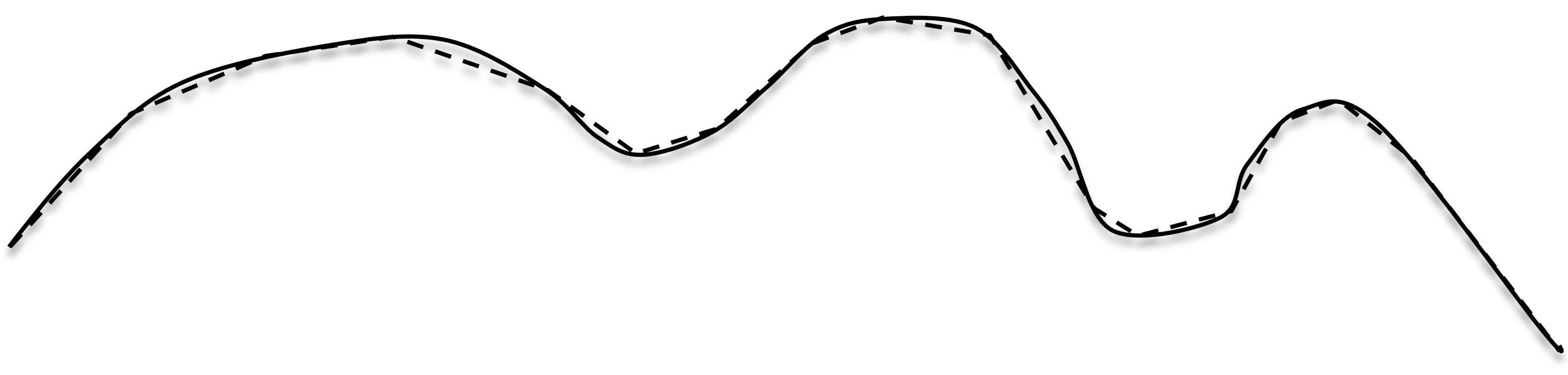
- =0 if the image is constant
- $\approx$0 if the image is piecewise-constant (why?)
  - The noise term will force a piecewise-constant image



$R(f)$: Piecewise-planer images

$$R^2(f) = \left(\frac{\partial^2 f}{\partial x^2}\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}\right)^2 + \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2$$

- =0 if the image is a plane
- $\approx$0 if the image is piecewise-planar
  - The noise term will force a piecewise-planar image



Graduated nonconvexity (GNC)

- Similar to MFA
  - Uses a descent method
  - Reduces a control parameter
  - Can be derived using MFA as its basis
  - “Weak membrane” GNC is analogous to piecewise-constant MFA
- But different:
  - Its objective function treats the presence of edges explicitly
    - Pixels labeled as edges don’t count in our noise term
    - So we must explicitly minimize the # of edge pixels

Variable conductance diffusion (VCD)

- Idea:
  - Blur an image everywhere,
  - except at features of interest
    - such as edges

VCD simulates the diffusion eq.

$$\frac{\partial f_i}{\partial t} = \nabla \cdot \left(c_i \cdot \nabla_i f \right)$$

temporal derivative

spatial derivative

- Where:
  - t = time
  - $\nabla_i f$ = spatial gradient of f at pixel i
  - c_i = conductivity (to blurring)

Isotropic diffusion

- If c_i is constant across all pixels:
  - *Isotropic* diffusion
    - Not really VCD
  - Isotropic diffusion is equivalent to convolution with a Gaussian
  - The Gaussian's variance is defined in terms of t and c_i

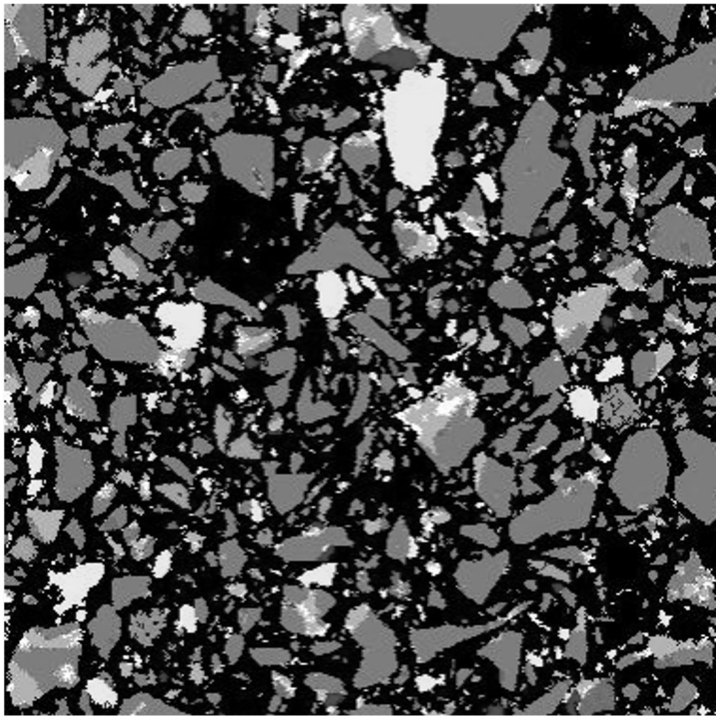
VCD

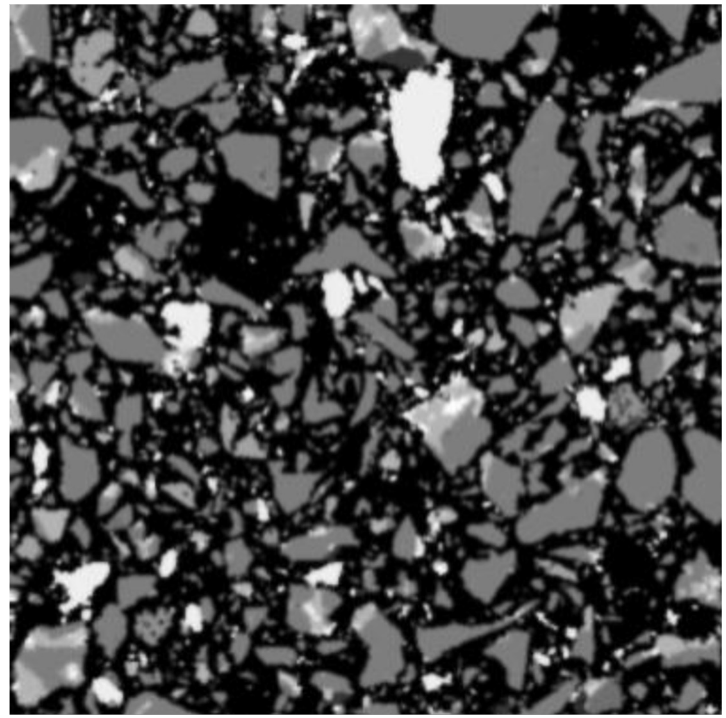
- c_i is a function of spatial coordinates, parameterized by i
  - Typically a property of the local image intensities
  - Can be thought of as a factor by which space is locally compressed
- To smooth except at edges:
  - Let c_i be small if i is an edge pixel
    - Little smoothing occurs because “space is stretched” or “little heat flows”
  - Let c_i be large at all other pixels
    - More smoothing occurs in the vicinity of pixel i because “space is compressed” or “heat flows easily”

VCD

- A.K.A. Anisotropic diffusion
- With repetition, produces a nearly piecewise uniform result
  - Like MFA and GNC formulations
  - Equivalent to MFA w/o a noise term
- Edge-oriented VCD:
  - VCD + diffuse tangential to edges when near edges
- Biased Anisotropic diffusion (BAD)
  - Equivalent to MAP image restoration

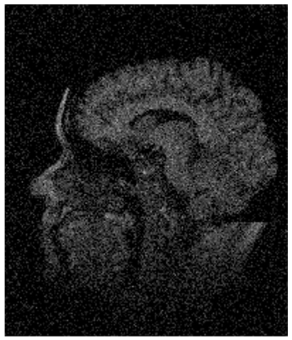
VCD Sample Images

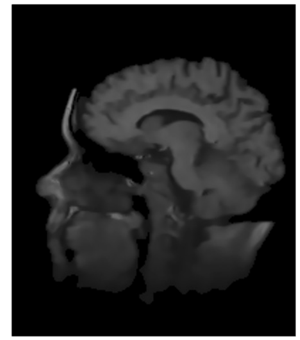




- From the Scientific Applications and Visualization Group at NIST
- http://math.nist.gov/mcsd/savg/software/filters/

Various VCD Approaches:
Tradeoffs and example images





- Mirebeau J., Fehrenbach J., Risser L., Tobji S., "Anisotropic Diffusion in ITK", the *Insight Journal*
- Images copied per Creative Commons license
- http://www.insight-journal.org/browse/publication/953
  - Then click on the "Download Paper" link in the top-right

Edge Preserving Smoothing

- Other techniques constantly being developed (but none is perfect)
- E.g., "A Brief Survey of Recent Edge-Preserving Smoothing Algorithms on Digital Images"
  - https://arxiv.org/abs/1503.07297
- SimpleITK filters:
  - BilateralImageFilter
  - Various types of AnisotropicDiffusionImageFilter
  - Various types of CurvatureFlowImageFilter

Congratulations!

- You have made it through most of the "introductory" material.
- Now we're ready for the "fun stuff."
- "Fun stuff" (why we do image analysis):
  - Segmentation
  - Registration
  - Shape Analysis
  - Etc.