Deconvolution

EE367/CS448I: Computational Imaging and Display
stanford.edu/class/ee367
Winter 2016, Lecture 6

Gordon Wetzstein
Stanford University
Lens as Optical Low-pass Filter

- point source on focal plane maps to point
Lens as Optical Low-pass Filter

- away from focal plane: out of focus blur

focal plane

blurred point
Lens as Optical Low-pass Filter

- shift-invariant convolution

focal plane
Lens as Optical Low-pass Filter

The convolution kernel is called the point spread function (PSF).

\[ b = c \ast x \]

- \( x \): sharp image
- \( b \): measured, blurred image
- \( c \): convolution kernel (PSF)
diffraction-limited PSF of circular aperture (aka “Airy” pattern):

\[ b = c \times x \]
PSF, OTF, MTF

- point spread function (PSF) is fundamental concept in optics
- optical transfer function (OTF) is (complex) Fourier transform of PSF
- modulation transfer function (MTF) is magnitude of OTF

example:

\[ \text{MTF}=|\text{OTF}| \]
\[ \text{OTF}=F\{\text{PSF}\} \]

PSF
PSF, OTF, MTF

- example:

  \[ \text{MTF} = |\text{OTF}| \]
  \[ \text{OTF} = \mathcal{F}\{\text{PSF}\} \]
  \[ \text{PSF} \]
Deconvolution

• given measurements $b$ and convolution kernel $c$, what is $x$?
Deconvolution with Inverse Filtering

- naive solution: apply inverse kernel

\[
\tilde{x} = c^{-1} \ast b = F^{-1} \left\{ \frac{F\{b\}}{F\{c\}} \right\}
\]
Deconvolution with Inverse Filtering & Noise

- naive solution: apply inverse kernel

\[ \tilde{x} = c^{-1} \ast b = F^{-1} \left\{ \frac{F\{b\}}{F\{c\}} \right\} \]

- Gaussian noise, \( \sigma = 0.05 \)
Deconvolution with Inverse Filtering & Noise

• results: terrible!

• why? this is an ill-posed problem (division by (close to) zero in frequency domain) → noise is drastically amplified!

• need to include prior(s) on images to make up for lost data
  • for example: noise statistics (signal to noise ratio)
Deconvolution with Wiener Filtering

• apply inverse kernel and don’t divide by 0

\[ \tilde{x} = F^{-1} \left\{ \frac{|F\{c}\}|^2}{|F\{c\}|^2 + \frac{1}{SNR}} \right\} \frac{F\{b\}}{F\{c\}} \]

amplitude-dependent damping factor!

\[ SNR = \frac{\text{mean signal} = 0.5}{\text{noise std} = \sigma} \]
Deconvolution with Wiener Filtering

naive

\( \hat{x} \)

Wiener
Deconvolution with Wiener Filtering

\[ \sigma = 0.01 \]

\[ \sigma = 0.05 \]

\[ \sigma = 0.1 \]
Deconvolution with Wiener Filtering

• results: not too bad, but noisy

• this is a heuristic $\rightarrow$ dampen noise amplification
Total Variation

\[
\begin{align*}
\min_x \|Cx - b\|_2^2 + \lambda TV(x) &= \min_x \|Cx - b\|_2^2 + \lambda \|\nabla x\|_1 \\
\|x\|_1 &= \sum_i |x_i|
\end{align*}
\]

- idea: promote sparse gradients (edges)

- \(\nabla\) is finite differences operator, i.e. matrix

\[
\begin{bmatrix}
-1 & 1 \\
-1 & 1 \\
\vdots & \\
-1 &
\end{bmatrix}
\]

Rudin et al. 1992
Total Variation

express (forward finite difference) gradient as convolution!

\[ x \rightarrow \nabla_x x \rightarrow \nabla_y x \]

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & -1 & 1 \\
0 & 0 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & -1 & 0 \\
0 & 1 & 0
\end{bmatrix}
\]
Total Variation

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

better: isotropic

easier: anisotropic
Total Variation

• for simplicity, this lecture only discusses anisotropic TV:

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

• problem: $l_1$-norm is not differentiable, can’t use inverse filtering

• however: simple solution for data fitting along and simple solution for TV alone \(\rightarrow\) split problem!
Deconvolution with ADMM

- split deconvolution with TV prior:

\[
\begin{align*}
\text{minimize} & \quad \|Cx - b\|_2^2 + \lambda \|z\|_1 \\
\text{subject to} & \quad \nabla x = z
\end{align*}
\]

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

\[
\begin{align*}
\text{minimize} & \quad f(x) + g(z) \\
\text{subject to} & \quad Ax + Bz = c
\end{align*}
\]

\[
\begin{align*}
f(x) & = \|Cx - b\|_2^2 \\
g(z) & = \lambda \|z\|_1 \\
A & = \nabla, \quad B = -I, \quad c = 0
\end{align*}
\]
minimize $f(x) + g(z)$

subject to $Ax + Bz = c$

• 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
minimize \quad f(x) + g(z) \quad \text{ADMM}

subject to \quad Ax + Bz = c

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

\[ L_\rho(x, y, z) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho / 2)\|Ax + Bz - c\|^2_2 \]
minimize $f(x) + g(z)$ \quad \text{ADMM}

subject to $Ax + Bz = c$

- ADMM consists of 3 steps per iteration $k$:

\[
\begin{align*}
x^{k+1} &:= \underset{x}{\text{arg min}} \ L_\rho(x, z^k, y^k) \\
z^{k+1} &:= \underset{z}{\text{arg min}} \ L_\rho(x^{k+1}, z, y^k) \\
y^{k+1} &:= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c)
\end{align*}
\]
minimize \( f(x) + g(z) \)  
subject to \( Ax + Bz = c \)

- ADMM consists of 3 steps per iteration \( k \):

\[
\begin{align*}
    x^{k+1} &:= \arg \min_x \left( f(x) + \left( \frac{\rho}{2} \right) \| Ax + Bz^k - c + u^k \| \right) \\
    z^{k+1} &:= \arg \min_z \left( g(z) + \left( \frac{\rho}{2} \right) \| Ax^{k+1} + Bz - c + u^k \| \right) \\
    u^{k+1} &:= u^k + Ax^{k+1} + Bz^{k+1} - c
\end{align*}
\]

scaled dual variable: \( u = (1/\rho)y \)
minimize \ f(x) + g(z)  \quad \text{ADMM}

subject to \ Ax + Bz = c

- ADMM consists of 3 steps per iteration k:

\[\begin{align*}
x^{k+1} & := \arg \min_x \left( f(x) + \left( \frac{\rho}{2} \right) \| Ax + Bz^k - c + u^k \|_2^2 \right) \\
z^{k+1} & := \arg \min_z \left( g(z) + \left( \frac{\rho}{2} \right) \| Ax^{k+1} + Bz - c + u^k \|_2^2 \right) \\
u^{k+1} & := u^k + Ax^{k+1} + Bz^{k+1} - c
\end{align*}\]

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

\( u \) connects them

scaled dual variable: \( u = (1 / \rho)y \)
Deconvolution with ADMM

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

subject to \( \nabla x - z = 0 \)

- ADMM consists of 3 steps per iteration \( k \):

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

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

\[
u^{k+1} := u^k + \nabla x^{k+1} - z^{k+1}
\]
minimize \[ \frac{1}{2}\|Cx - b\|_2^2 + \lambda \|z\|_1 \]

subject to \[ \nabla x - z = 0 \]

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

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

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

subject to \(\nabla x - z = 0\)

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

\[
\arg\min_x \|\tilde{C}x - \tilde{b}\|_2^2 = \arg\min_x \left( (C^TC + \rho\nabla^T\nabla)x - (C^Tb + \rho\nabla^Tv) \right)^2
\]

- inverse filtering: \(x^{k+1} = F^{-1}\left[\begin{array}{c}
F\{c\}^* \cdot F\{b\} + \rho \left( F\{\nabla_x\}^* \cdot F\{v_1\} + F\{\nabla_y\}^* \cdot F\{v_2\} \right) \\
F\{c\}^* \cdot F\{c\} + \rho \left( F\{\nabla_x\}^* \cdot F\{\nabla_x\} + F\{\nabla_y\}^* \cdot F\{\nabla_y\} \right)
\end{array}\right]
\]

\[
\Rightarrow \text{may blow up, but that's okay}
\]

constant, say \(v = z^k - u^k\)

precompute!
Deconvolution with ADMM

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2}\|Cx - b\|_2^2 + \lambda \|z\|_1 \\
\text{subject to} & \quad \nabla x - z = 0 & \text{constant, say } a = \nabla x_{k+1}^k + u_k
\end{align*}
\]

2. z-update:

\[
z^{k+1} = \arg \min_z \left( \lambda \|z\|_1 + (\rho / 2)\|\nabla x^{k+1}_k - z + u_k\|_2^2 \right)
\]

- \(l_1\)-norm is not differentiable! yet, closed-form solution via \textit{element-wise} soft thresholding:

\[
z^{k+1} := S_{\lambda/\rho}(a)
\]

\[
S_{\kappa}(a) = \begin{cases} 
    a - \kappa & a > \kappa \\
    0 & |a| \leq \kappa \\
    a + \kappa & a < -\kappa
\end{cases}
\]

\[
\kappa = \lambda / \rho
\]
Deconvolution with ADMM

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|Cx - b\|_2^2 + \lambda \|z\|_1 \\
\text{subject to} & \quad \nabla x - z = 0
\end{align*}
\]

for \( k=1:\text{max\_iters} \)

\[
\begin{align*}
x^{k+1} & := \arg \min_x \left( \frac{1}{2} \|\tilde{C}x - \tilde{b}\|_2^2 \right) \quad \text{solve with inverse filtering} \\
z^{k+1} & := S_{\lambda/\rho}(\nabla x^{k+1} + u^k) \quad \text{element-wise thresholding} \\
u^{k+1} & := u^k + \nabla x^{k+1} - z^{k+1} \quad \text{trivial}
\end{align*}
\]
Deconvolution with ADMM

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| Cx - b \|_2^2 + \lambda \| z \|_1 \\
\text{subject to} & \quad \nabla x - z = 0
\end{align*}
\]

Wiener filtering

ADMM with anisotropic TV, \( \lambda = 0.01, \rho = 10 \)
Deconvolution with ADMM

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

subject to \( \nabla x - z = 0 \)

- too much TV: “patchy”, too little TV: noisy

\( \lambda = 0.05, \rho = 10 \) \hspace{1cm} \( \lambda = 0.01, \rho = 10 \) \hspace{1cm} \( \lambda = 0.1, \rho = 10 \)
Deconvolution with ADMM

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

subject to \quad \nabla x - z = 0

Wiener filtering

ADMM with anisotropic TV, \quad \lambda = 0.1, \quad \rho = 10
Deconvolution with ADMM

\[
\text{minimize } \frac{1}{2} \| Cx - b \|_2^2 + \lambda \| z \|_1 \\
\text{subject to } \nabla x - z = 0
\]

- too much TV: okay because image actually has sparse gradients!

\( \lambda = 0.05, \rho = 10 \) \hspace{1cm} \( \lambda = 0.01, \rho = 10 \) \hspace{1cm} \( \lambda = 0.1, \rho = 10 \)
Outlook ADMM

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

\[
\begin{align*}
\text{minimize } & \quad \frac{1}{2} \| Ax - b \|_2^2 + \Gamma(x) \\
\text{subject to } & \quad Ax = z
\end{align*}
\]

\[
\begin{align*}
\text{minimize } & \quad f(x) + g(z) \\
\text{subject to } & \quad Ax = z
\end{align*}
\]

• example priors: noise statistics, sparse gradient, smoothness, …
• weighted sum of different priors also possible
• anisotropic TV is one of the easiest priors
• implement matrix-free operations for $Ax$ and $A'x$ if efficient (e.g. multiplications and divisions in frequency space)

• split difficult problems (e.g., inverse problems with non-differentiable priors) into easier subproblems - ADMM
Homework 3

• implement:
  • filtering
  • inverse filtering and Wiener filtering
  • deconvolution with ADMM + (anisotropic) TV prior
Notes for Homework 3

- notes for ADMM implementation:
  - initialize U, Z, X with 0
  - implement with matrix-free form: all FT multiplications / divisions

- in 2D, finite differences matrix becomes (anisotropic form), use matrix free-operations as well!

- see note notes in HW

- check ADMM example scripts: http://web.stanford.edu/~boyd/papers/admm/
Notes for Homework 3

- signal-to-noise ratio (SNR): 
  \[ SNR = \frac{P_{\text{signal}}}{P_{\text{noise}}} \quad SNR_{\text{dB}} = 10 \cdot \log_{10} \left( \frac{P_{\text{signal}}}{P_{\text{noise}}} \right) \]

- peak signal-to-noise ratio (PSNR): 
  \[ MSE = \frac{1}{mn} \sum_m \sum_n (x_{\text{target}} - x_{\text{est}})^2 \]
  \[ PSNR = 10 \cdot \log_{10} \left( \frac{\text{max}(x_{\text{target}})^2}{MSE} \right) = 10 \cdot \log_{10} \left( \frac{1}{MSE} \right) \]

- residual is value of objective function:
  
  not regularized: \( \frac{1}{2} \|Cx - b\|_2^2 \)
  regularized: \( \frac{1}{2} \|Cx - b\|_2^2 + \lambda \left\| \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} x \right\|_1 \)

- convergence: residual for increasing iterations (should always decrease!)
• Boyd, Parikh, Chu, Peleato, Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers", Foundations and Trends in Machine Learning, 2011


• Rudin, Osher, Fatemi, “Nonlinear total variation based noise removal algorithms”, Physica D: Nonlinear Phenomena 60, 1

• http://www.imagemagick.org/Usage/fourier/