Deconvolution with ADMM

EE367/CS448I: Computational Imaging and Display
stanford.edu/class/ee367
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
Lens as Optical Low-pass Filter

- shift-invariant convolution

focal plane
Lens as Optical Low-pass Filter

convolution kernel is called point spread function (PSF)

\[ b = c \ast x \]

sharp image

measured, blurred image

point spread function (PSF): \( c \)
Lens as Optical Low-pass Filter

diffraction-limited PSF of circular aperture (aka “Airy” pattern):

\[ b = c \ast x \]
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:

MTF = |OTF|
OTF = F{PSF}
PSF
Deconvolution

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

- naive solution: apply inverse kernel

\[ \tilde{x} = c^{-1} \ast b = F^{-1} \begin{pmatrix} F\{b\} \\ F\{c\} \end{pmatrix} \]
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

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

amplitude-dependent damping factor!

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

Naïve inverse filter

\( x \)

Wiener

\( \tilde{x} \)
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

\[
\min_x \|Cx - b\|_2^2 + \lambda TV(x) = \min_x \|Cx - b\|_2^2 + \lambda \|\nabla x\|_1
\]

\[\|x\|_1 = \sum_i |x_i|\]

- idea: promote sparse gradients (edges)

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

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

Rudin et al. 1992
Total Variation

express (forward finite difference) gradient as convolution!

\[ \nabla_y x \]

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

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

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

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

easier: anisotropic

better: isotropic
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: l1-norm doesn’t allow for an inverse filtering approach

• however: simple solution for data fitting along and simple solution for TV alone → 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):

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

where:

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

- split deconvolution with TV prior:

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

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

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

  \[
  f(x) = \|Cx - b\|_2^2 \\
  g(z) = \lambda \|z\|_1 \\
  A = \nabla, \quad B = -I, \quad c = 0
  \]
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):

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

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

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

\[
\begin{align*}
f(x) &= \|Cx-b\|_2^2 \\
g(z) &= \lambda \|z\|_1
\end{align*}
\]

\[
A = \nabla, \quad B = -I, \quad c = 0
\]
minimize \quad f(x) + g(z) \quad \text{ADMM}

subject to \quad Ax + Bz = c

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

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

\[ \uparrow \]

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 $k$:

$$
\begin{align*}
x^{k+1} & := \arg\min_x L_{\rho}(x, z^k, y^k) \\
z^{k+1} & := \arg\min_z L_{\rho}(x^{k+1}, z, y^k) \\
y^{k+1} & := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c)
\end{align*}
$$
minimize \( f(x) + g(z) \) \hfill \text{ADMM}
subject to \( Ax + Bz = c \)

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

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

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

subject to \( Ax + Bz = c \)

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

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

  (u connects them)

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

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

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

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

\[
\begin{align*}
x^{k+1} & := \arg \min_x \left( \frac{1}{2} \| Cx - b \|_2^2 + (\rho / 2) \| \nabla x - z^k + u^k \|_2^2 \right) \\
z^{k+1} & := \arg \min_z \left( \lambda \| z \|_1 + (\rho / 2) \| \nabla x^{k+1} - z + u^k \|_2^2 \right) \\
u^{k+1} & := u^k + \nabla x^{k+1} - z^{k+1}
\end{align*}
\]
Deconvolution with ADMM

\[
\begin{aligned}
\text{minimize} & \quad \frac{1}{2} \|Cx - b\|_2^2 + \lambda \|z\|_1 \\
\text{subject to} & \quad \nabla x - z = 0 \\
1. \ x\text{-update:} & \quad x^{k+1} := \arg \min_x \left( \frac{1}{2} \|Cx - b\|_2^2 + \frac{\rho}{2} \|\nabla x - z^k + u^k\|_2^2 \right) \\
\end{aligned}
\]

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

\[
\begin{aligned}
\nabla^T v &= \begin{bmatrix} \nabla_x^T \\ \nabla_y^T \end{bmatrix} \\
v &= \nabla_x^T v_1 + \nabla_y^T v_2 \\
\end{aligned}
\]
Deconvolution with ADMM

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

subject to \: \nabla x - z = 0

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

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

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

\[ \Rightarrow \text{may blow up, but that’s okay} \]
Deconvolution with ADMM

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

subject to \quad \nabla x - z = 0 \quad \text{constant, say } a = \nabla x^{k+1} + u^k

2. z-update: \quad z^{k+1} := \arg \min_z \left( \lambda \|z\|_1 + \left(\frac{\rho}{2}\right) \|\nabla x^{k+1} - z + u^k\|_2^2 \right)
Deconvolution with ADMM

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

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

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

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

\[
u^{k+1} := u^k + \nabla x^{k+1} - z^{k+1} \quad \text{trivial}
\]
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 \( k = 1 : \text{max\_iters} \)

\[
x^{k+1} := \text{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 \) \quad \text{inverse filtering}
\]

\[
z^{k+1} := S_{\lambda/\rho} (\nabla x^{k+1} + u^k) \quad \text{element-wise threshold trivial}
\]

\[
u^{k+1} := u^k + \nabla x^{k+1} - z^{k+1} \quad \rightarrow \text{easy! 😊}
\]
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, \( \lambda = 0.01, \rho = 10 \)
Deconvolution with ADMM

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

subject to \quad \nabla x - z = 0

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

\[
\lambda = 0.01, \rho = 10 \\
\lambda = 0.05, \rho = 10 \\
\lambda = 0.1, \rho = 10
\]
minimize $\frac{1}{2} \|Cx - b\|_2^2 + \lambda \|z\|_1$

subject to $\nabla x - z = 0$

Wiener filtering

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

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

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

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

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

\[
\minimize_x \frac{1}{2} \left\| Ax - b \right\|_2^2 + \underbrace{\Gamma(x)}_{\text{regularization}} \quad \Rightarrow \quad \minimize_{\{x,z\}} \ f(x) + g(z)
\]
subject to
\[
Ax = z
\]

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

• 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):**
  \[ SNR = \frac{P_{signal}}{P_{noise}} \quad SNR_{dB} = 10 \cdot \log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) \]

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

- **residual** is value of objective function:
  \[
  \begin{aligned}
  &\text{not regularized:} \quad \frac{1}{2} \|Cx - b\|_2^2 \\
  &\text{regularized:} \quad \frac{1}{2} \|Cx - b\|_2^2 + \lambda \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} x \\
  &\text{convergence:} \quad \text{residual for increasing iterations (should always decrease!)}
  \end{aligned}
  \]
References and Further Reading


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

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