Optimization Methods for Total Variation Based Image Restoration

D.Goldfarb (Columbia University)
joint with W.Yin (Rice University)

Workshop on Inverse Problem at Columbia University, 05-03-2007
Image Processing

filter black box

\( f \)  \[ \rightarrow \]  \[ \rightarrow \]  \( u \)
Optimization Methods for Total Variation Based Image Restoration
Optimization Methods for Total Variation Based Image Restoration

$\mathbf{f}$  

Noise removal filter  

$\mathbf{u}$
Optimization Methods for Total Variation Based Image Restoration

Texture removal filter
One minimizes different functional to obtain $u$ for these two cases:

\[
\begin{align*}
\min_u & \quad TV(u) + \lambda \|f - u\|_{L^2}^2 \\
\min_u & \quad TV(u), \text{ s.t. } \|f - u\|_{L^2} \leq \sigma
\end{align*}
\]

\[
\begin{align*}
\min_u & \quad TV(u) + \lambda \|f - u\|_{L^1} \\
\min_u & \quad TV(u), \text{ s.t. } \|f - u\|_{L^1} \leq \sigma
\end{align*}
\]

\[
TV(u) := \int_\Omega |\nabla u(x)| \, dx
\]

- Convex problems

ROF: Rudin-Osher-Fatemi, TV/L1: Alliney, Nikonova, Chan-Esedoglu, Yin-Goldfarb-Osher
Methods

*PDE*-based Gradient descent:
- low memory usage
- slow convergence

**SOCP / interior-point method**:
- high memory usage
- better convergence

*Network flows methods*:
- low memory usage
- very fast
- *not as general*
The PDE-based gradient descent approach

- Euler-Lagrange Eqns for the *unconstrained* ROF model:

\[
0 = g(u) := \frac{1}{2\lambda} \nabla \cdot \frac{\nabla u}{|\nabla u|} + (f - u)
\]

- Solve \( \frac{\partial u}{\partial t} = g(u), \quad u(0) = f \) (Homogeneous Neumann boundary condition) to steady state

- must regularize \( \| \nabla u_{i,j} \| \approx \sqrt{|\nabla_1 u_{i,j}|^2 + |\nabla_2 u_{i,j}|^2 + \varepsilon} \)

where \( \nabla_1 u_{i,j} := u_{i+1,j} - u_{i,j} \)
\( \nabla_2 u_{i,j} := u_{i,j+1} - u_{i,j} \)
The Second-Order Programming (SOCP) approach

Discrete ROF:

$$\min \sum_{1 \leq i, j \leq n} t_{i,j}$$

s.t.

(linear) \quad u + v = f

(SOCs) \quad \|\nabla u_{i,j}\| \leq t_{i,j}

(SOC) \quad \|v\|_2^2 \leq \sigma^2.

• Handles all constrained and unconstrained ROF and TV/$L^1$ models
• Does not require regularization
• Solved by interior-point methods
• Linear algebra is accelerated by applying nested dissection

SOCPs for TV-based models: Goldfarb-Yin 05'
Optimization Methods for Total Variation Based Image Restoration

\[ C^1 : (\partial^+_{x} u)_{i,j} + (v_{i+1,j} - v_{i,j}) = f_{i+1,j} - f_{i,j} \]

\[ C^2 : (\partial^+_{y} u)_{i,j} + (v_{i,j+1} - v_{i,j}) = f_{i,j+1} - f_{i,j} \]

Cholesky with Nested Dissec.: multiplications \((943/84)n^3 + O(n^2 \log_2 n)\)

storage \((31/4)n^2 \log_2 n + O(n^2)\)

Nested Dissection: A.George 73′
Nested Dissection

\[ M = \begin{bmatrix}
M_{UU} & 0 & M_{US} \\
0 & M_{VV} & M_{VS} \\
M_{US}^{T} & M_{VS}^{T} & M_{SS}
\end{bmatrix}, \quad L = \begin{bmatrix}
L_{UU} & 0 & 0 \\
0 & L_{VV} & 0 \\
L_{SU} & L_{SV} & L_{SS}
\end{bmatrix} \]
Nested Dissection

\[
M = \begin{bmatrix}
M_{UU} & 0 & M_{US} \\
0 & M_{VV} & M_{VS} \\
M_{US}^T & M_{VS}^T & M_{SS}
\end{bmatrix}
\]

\[
L = \begin{bmatrix}
L_{UU} & 0 & 0 \\
0 & L_{VV} & 0 \\
L_{SU} & L_{SV} & L_{SS}
\end{bmatrix}
\]
Nested Dissection

\[
M = \begin{bmatrix}
M_{UU} & 0 & M_{US} \\
0 & M_{VV} & M_{VS} \\
M_{US}^T & M_{VS}^T & M_{SS}
\end{bmatrix}
\]

\[
L = \begin{bmatrix}
L_{UU} & 0 & 0 \\
0 & L_{VV} & 0 \\
L_{SU} & L_{SV} & L_{SS}
\end{bmatrix}
\]
SOCP(Matlab+Mosek) numerical results on tests using unconstrained ROF

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>best $\lambda$</th>
<th>total time*</th>
<th># of itr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>books</td>
<td>64×64</td>
<td>0.03747189</td>
<td>4.42</td>
<td>15</td>
</tr>
<tr>
<td>dotplus</td>
<td>100×100</td>
<td>0.01805139</td>
<td>13.68</td>
<td>16</td>
</tr>
<tr>
<td>scale</td>
<td>252×252</td>
<td>0.03370882</td>
<td>157.86</td>
<td>17</td>
</tr>
<tr>
<td>barbara</td>
<td>256×256</td>
<td>0.03746579</td>
<td>130.13</td>
<td>16</td>
</tr>
<tr>
<td>barbara</td>
<td>350×350</td>
<td>0.03749693</td>
<td>283.56</td>
<td>16</td>
</tr>
</tbody>
</table>

Solving TV/L1 by SOCP is a few times slower

* seconds on a SUN E450 with 350Mhz CPUs and 4GB memory
Max flow approach outline:
applicable to anisotropic TV(u) – i.e., \( l_1 \) norm

1. Decompose \( f \) into level sets \( F_l = \{ x \mid f(x) \geq l \} \)
Max flow approach outline:
applicable to anisotropic TV(u) – i.e., $l_1$ norm

1. Decompose $f$ into level sets $F_l = \{x \mid f(x) \geq l\}$

2. For each $F_l$, obtain $U_l$ by solving a max-flow prob
(solving a binary $\min_U TV(U) + \lambda \|F - U\|_{L^1}$)

3. Construct a minimizer $u$ from the minimizers $U_l$

$$TV(u) + \lambda \|f - u\|_{L^1} = \int_{levels} \left[ TV(U_l) + \lambda \|F_l - U_l\|_{L^1} \right] dl$$

Layer-cake formula: Chan-Esedoglu 05’
Optimization Methods for Total Variation Based Image Restoration

\[ f \rightarrow \text{block} \rightarrow u \]
Requires monotonicity of $f$
Is it good to break the problem up into levels?

- = finding a minimum cut of a capacitated network
(Why? Answer coming next......)

- For a 8-bit image, there are $2^8 = 256$ levels

- For a 16-bit image, there are $2^{16} = 65536$ levels

- Answer depends on
  1. how fast we can solve each
  2. how many we need to solve
A capacitated network

- A Network is a graph $G$ with nodes and edges: $G = (V, E)$
- Special nodes $s$ (source) and $t$ (sink)
- Edges carry flow
- Each edge $(i,j)$ has a maximum capacity $c_{i,j}$
A capacitated network

- A Network is a graph $G$ with nodes and edges: $G = (V, E)$
- Special nodes $s$ (source) and $t$ (sink)
- Edges carry flow
- Each edge $(i,j)$ has a maximum capacity $c_{i,j}$
- An $s$-$t$ cut $(S,T)$ is a 2-partition of $V$ such that $s$ in $S$, $t$ in $T$

*Cut value*: the total $s$-$t$ cap. across the cut $= 3 + 7 + 11 = 21$
A capacitated network

- A Network is a graph \( G \) with nodes and edges: \( G = (V, E) \)
- Special nodes \( s \) (source) and \( t \) (sink)
- Edges carry flow
- Each edge \((i,j)\) has a maximum capacity \( c_{i,j} \)
- An \( s-t \) cut \((S,T)\) is a 2-partition of \( V \) such that \( s \) in \( S \), \( t \) in \( T \)
- A min \( s-t \) cut is one that gives the minimum cut value

- **Cut value**: the total \( s-t \) cap. across the cut=15+3=18
A capacitated network

- A Network is a graph $G$ with nodes and edges: $G = (V, E)$
- Special nodes $s$ (source) and $t$ (sink)
- Edges carry flow
- Each edge $(i,j)$ has a maximum capacity $c_{i,j}$
- An $s$-$t$ cut $(S,T)$ is a 2-partition of $V$ such that $s$ in $S$, $t$ in $T$
- A min $s$-$t$ cut is one that gives the minimum cut value
- Finding a min-cut = finding a max-flow

**Cut value**: the total $s$-$t$ cap. across the cut = $15 + 3 = 18$
Max flow problem \[ G = (V, E) \]

dual var. \[ \max_{x,v} \quad v \]

s.t.

\[ u_i \quad \sum_{j:(i,j) \in E} x_{ij} - \sum_{j:(j,i) \in E} x_{ji} = \begin{cases} v & i = s \\ 0 & i \in V \setminus \{s,t\} \\ -v & i = t \end{cases} \]

\[ \delta_{ij} \quad 0 \leq x_{ij} \leq c_{ij}, \ \forall (i,j) \in E. \]
Optimization Methods for Total Variation Based Image Restoration

Max flow problem \( G = (V, E) \)

dual var. \( \max_{x,v} \ v \)

s.t.

\[
\begin{align*}
    u_i & = \sum_{j: (i,j) \in E} x_{ij} - \sum_{j: (j,i) \in E} x_{ji} \quad \begin{cases} 
        v & i = s \\
        0 & i \in V \setminus \{s,t\} \\
        -v & i = t
    \end{cases} \\
    \delta_{ij} & \geq 0, \quad \forall (i,j) \in E.
\end{align*}
\]

Min cut problem (dual of above)

\[
\begin{align*}
    \min_{u,\delta} & \quad \sum_{(i,j) \in E} c_{ij}\delta_{ij} \\
    \text{s.t.} & \quad u_i - u_j + \delta_{ij} \geq 0, \quad \forall (i,j) \in E \\
    & \quad u_s = 0, \quad u_t = 1 \quad (0 \leq u_i \leq 1, \quad \forall i \text{ implicitly}) \\
    & \quad \delta_{ij} \geq 0, \quad \forall (i,j) \in E.
\end{align*}
\]

\( \exists \) a **binary** optimal solution \( u^*, \delta^* \). Min cut is given by \( S := \{ i : u_i^* = 0 \} \), \( T := \{ i : u_i^* = 1 \} \).
In 1D, discrete $\min_u TV(u) + \lambda \|f - u\|_{L^1}$ gives
$\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|$. Let’s consider the **binary** problem.
In 1D, discrete \( \min_u TV(u) + \lambda \| f - u \|_{L^1} \) gives
\[
\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|.
\]
Let’s consider the **binary** problem.

1. Use \( \min |x| \Leftrightarrow \min y_1 + y_2, \text{ s.t. } x \leq y_1, -x \leq y_2, y_1, y_2 \geq 0 \):
In 1D, discrete $\min_u TV(u) + \lambda \|f - u\|_{L^1}$ gives
$$\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|.$$ Let’s consider the binary problem.

1. Use $\min |x| \iff \min y_1 + y_2$, s.t. $x \leq y_1, -x \leq y_2, y_1, y_2 \geq 0$:
   We have each $\min |u_{i+1} - u_i| \iff \min \delta_{i,i+1} + \delta_{i+1,i}$
   s.t. $u_{i+1} - u_i + \delta_{i+1,i} \geq 0$
   $u_i - u_{i+1} + \delta_{i,i+1} \geq 0$
   $\delta_{i+1,i}, \delta_{i,i+1} \geq 0.$
In 1D, discrete \( \min_u TV(u) + \lambda \|f - u\|_{L^1} \) gives
\[
\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|.
\]
Let’s consider the \textbf{binary} problem.

1. Use \( \min |x| \Leftrightarrow \min y_1 + y_2, \text{ s.t. } x \leq y_1, -x \leq y_2, y_1, y_2 \geq 0 \):
   We have each \( \min |u_{i+1} - u_i| \Leftrightarrow \min \delta_{i,i+1} + \delta_{i+1,i} \)
   s.t. \( u_{i+1} - u_i + \delta_{i+1,i} \geq 0 \)
   \( u_i - u_{i+1} + \delta_{i,i+1} \geq 0 \)
   \( \delta_{i+1,i}, \delta_{i,i+1} \geq 0 \).

2. Each \( \min \lambda |0 - u_i| \Leftrightarrow \min \lambda \delta_{si}, \text{ s.t. } \delta_{si} \geq u_i \) \( (u_i \geq 0 \text{ implicitly}) \)
   \( \Leftrightarrow \min \lambda \delta_{si}, \text{ s.t. } u_s - u_i + \delta_{si} \geq 0 \)
   \( u_s = 0 \).
In 1D, discrete \( \min_u TV(u) + \lambda \|f - u\|_{L^1} \) gives
\[
\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|.
\]
Let’s consider the \textbf{binary} problem.

1. Use \( \min |x| \Leftrightarrow \min y_1 + y_2, \text{ s.t. } x \leq y_1, -x \leq y_2, y_1, y_2 \geq 0 \):
We have each \( \min |u_{i+1} - u_i| \Leftrightarrow \min \delta_{i,i+1} + \delta_{i+1,i} \)
\[
\text{s.t. } u_{i+1} - u_i + \delta_{i+1,i} \geq 0
\]
\[
u_i - u_{i+1} + \delta_{i,i+1} \geq 0
\]
\[
\delta_{i+1,i}, \delta_{i,i+1} \geq 0.
\]

2. Each \( \min \lambda |0 - u_i| \Leftrightarrow \min \lambda \delta_{si}, \text{ s.t. } \delta_{si} \geq u_i \) \( (u_i \geq 0 \text{ implicitly}) \)
\[
\Leftrightarrow \min \lambda \delta_{si}, \text{ s.t. } u_s - u_i + \delta_{si} \geq 0
\]
\[
u_s = 0
\]

3. Each \( \min \lambda |1 - u_i| \Leftrightarrow \min \lambda \delta_{it}, \text{ s.t. } \delta_{it} \geq 1 - u_i \) \( (u_i \leq 1 \text{ implicitly}) \)
\[
\Leftrightarrow \min \lambda \delta_{si}, \text{ s.t. } u_i - u_t + \delta_{it} \geq 0
\]
\[
u_t = 1
\]
Combining 1,2,3 gives a min cut formulation!
Alternative explanations for the 3 types of arcs:

In 1D, binary TV/$L^1$: \[ \min_U |\text{jumps}(U)| + \lambda |U \triangle F| \]
1. contribution to \# of jumps in U
2. 3.: contribution to the length of $U \triangle F$

In 2D, binary TV/$L^1$: \[ \min_U \text{Per}(U) + \lambda \text{Area}(U \triangle F) \]
1. contribution to \text{Per}(U)
2. 3.: contribution to \text{Area}(U \triangle F)

where $U \triangle F := (F \setminus U) \cup (U \setminus F)$
In 1D, discrete min_u TV(u) + \lambda \|f - u\|_L^1 gives
\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|.
Let's consider the \textbf{binary} problem.
Example: \(u = (u_1, \ldots, u_7)\) and \(f = (0, 0, 0, 1, 1, 1, 0)\).
In 1D, discrete \( \min_u TV(u) + \lambda \| f - u \|_{L^1} \) gives
\[
\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|.
\]
Let's consider the **binary** problem.
Example: \( u = (u_1, \ldots, u_7) \) and \( f = (0, 0, 0, 1, 1, 1, 0) \).
- construct a network with terminal arcs determined by \( f \).
In 1D, discrete \( \min_u TV(u) + \lambda \| f - u \|_{L^1} \) gives \( \min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i| \).

Let’s consider the **binary** problem.

Example: \( u = (u_1, \ldots, u_7) \) and \( f = (0, 0, 0, 1, 1, 1, 0) \).

- construct a network with terminal arcs determined by \( f \).
- define an **S-T** cut by letting \( S = \{ i : u_i = 0 \} \).
In 1D, discrete $\min_u TV(u) + \lambda \|f - u\|_{L^1}$ gives $\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|$. Let’s consider the **binary** problem.

Example: $u = (u_1, \ldots, u_7)$ and $f = (0, 0, 0, 1, 1, 1, 0)$.

- construct a network with terminal arcs determined by $f$.
- define an $S$-$T$ cut by letting $S = \{i : u_i = 0\}$.
  example: $u_i = (0, 0, 1, 1, 1, 0, 0)$. 

![Diagram of network with terminal arcs and cuts](image-url)
In 1D, discrete $\min_u TV(u) + \lambda \|f - u\|_{L^1}$ gives
$\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|$. 
Let’s consider the binary problem.
Example: $u = (u_1, \ldots, u_7)$ and $f = (0, 0, 0, 1, 1, 1, 0)$.
- construct a network with terminal arcs determined by $f$.
- define an $S-T$ cut by letting $S = \{i : u_i = 0\}$.
  example: $u_i = (0, 0, 1, 1, 1, 0, 0)$.
- energy = cut value.

$$\min_u \sum |u_{i+1} - u_i| + \lambda \sum |f_i - u_i| = 2 + 2\lambda.$$
In 2D, we have

\[ U_l = \{ \bullet \} \quad Per(U_l) = \text{length(○○)} \approx |\text{edges cut by ○○}| \]

\( Per(U_l) \) will get more accurately approximated if more neighbors are used
Isotropic TV v.s. Anisotropic TV

- $\nabla_1 u := u_{i+1,j} - u_{i,j}$ and $\nabla_2 u := u_{i,j+1} - u_{i,j}$

- The isotropic discretization of $TV(u)$: $\sqrt{|\nabla_1 u|^2 + |\nabla_2 u|^2}$

- An anisotropic discretization: $|\nabla_1 u| + |\nabla_2 u|

- A better anisotropic discretization:

  Given a pixel $x \in U$, let $n_4(x)$, $n_8(x)$, and $n_k(x)$ be the number of pixels which are in the four, eight, and "knight-move" connected neighborhood of $v$ outside $S$

  $\text{Per}(U) = \sum_{x \in U} [0.26n_4(x) + 0.19n_8(x) + 0.06n_k(x)]$

  Watersnake: Nguyen-Worring-van den Boomgaard 03’
In 2D, we solve

\[
\min_{U_l} \operatorname{Per}(U_l) + \lambda \operatorname{Area}(F_l \triangle U_l)
\]
Outline: Further steps

1. Decompose $f$ into $K$ level sets $F_i$
2. For each $F_i$, obtain $U_i$
   1. $U_i$ \(\leftarrow\) min-cut of a network (Graph-Cut)
   2. min-cut $\leftrightarrow$ max-flow
3. (For TV/$L^1$) Combine $K$ networks (para. max flow)
4. (For ROF) Reduce $K$ max-flows to $\log K$ parametric max-flows (e.g., $K=2^{16}=65536$, $\log K=16$)
3. Construct a minimizer $u$ from the minimizers $U_i$
Divide and conquer (Darbon & Siegelle)

Divide and Conquer: Darbon-Siegelle 06’
Max flow / min cut algorithms

- Preflow push (Goldberg-Tarjan)
  - Best complexity: $O(nm \log(n^2/m))$
- Boykov-Kolmogrov, push on path
  - Uses approximate shortest path
  - Not strongly polynomial
  - Very fast on graph with small neighborhoods
- Parametric max flow (Gallo et al.)
  - Complexity same as preflow push: $O(nm \log(n^2/m))$, if # of levels is $O(n)$
  - Arcs out of source have *increasing* capacities
  - Arcs into sink have *decreasing* capacities

Preflow: Goldberg-Tarjan, B-K: Boykov-Kolmogrov 04’
Parametric: Gallo, Grigoriadis, Tarjan 89’, Hochbaum 01’
Max-flow (Matlab/C++) numerical results

<table>
<thead>
<tr>
<th>Model</th>
<th>Name</th>
<th>Size</th>
<th>best $\lambda$</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TV/L$^1$</td>
<td>Barbara (8-bit)</td>
<td>512×512</td>
<td>0.5</td>
<td>0.96</td>
</tr>
<tr>
<td>TV/L$^1$</td>
<td>Barbara (8-bit)</td>
<td>1024×1024</td>
<td>0.5</td>
<td>3.98</td>
</tr>
<tr>
<td>ROF</td>
<td>Barbara (8-bit)</td>
<td>512×512</td>
<td>0.0375</td>
<td>1.83</td>
</tr>
<tr>
<td>ROF</td>
<td>Barbara (8-bit)</td>
<td>1024×1024</td>
<td>0.0375</td>
<td>7.50</td>
</tr>
<tr>
<td>ROF</td>
<td>Barbara (16-bit)</td>
<td>1024×1024</td>
<td>0.0375</td>
<td>13.5</td>
</tr>
</tbody>
</table>

Laptop - CPU: Pentium Duo 2.0GHz, Memory: 1.5 GB
Input $f$

Output $u$

$\lambda > \lambda_1$

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1$</td>
<td>$\lambda_2$</td>
<td>$\lambda_3$</td>
<td>$\lambda_4$</td>
<td>$\lambda_5$</td>
</tr>
<tr>
<td>19.40</td>
<td>13.40</td>
<td>7.96</td>
<td>4.57</td>
<td>2.35</td>
</tr>
</tbody>
</table>
Input $f$

Output $u$

$\lambda_1 > \lambda > \lambda_2$

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1$</td>
<td>$\lambda_2$</td>
<td>$\lambda_3$</td>
<td>$\lambda_4$</td>
<td>$\lambda_5$</td>
</tr>
<tr>
<td>19.40</td>
<td>13.40</td>
<td>7.96</td>
<td>4.57</td>
<td>2.35</td>
</tr>
</tbody>
</table>
Optimization Methods for Total Variation Based Image Restoration

Input $f$

Output $u$

$\lambda_2 > \lambda > \lambda_3$

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1$</td>
<td>$\lambda_2$</td>
<td>$\lambda_3$</td>
<td>$\lambda_4$</td>
<td>$\lambda_5$</td>
</tr>
<tr>
<td>19.40</td>
<td>13.40</td>
<td>7.96</td>
<td>4.57</td>
<td>2.35</td>
</tr>
</tbody>
</table>
Optimization Methods for Total Variation Based Image Restoration

Input $f$

Output $u$

$\lambda_3 > \lambda > \lambda_4$

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1$</td>
<td>$\lambda_2$</td>
<td>$\lambda_3$</td>
<td>$\lambda_4$</td>
<td>$\lambda_5$</td>
</tr>
<tr>
<td>19.40</td>
<td>13.40</td>
<td>7.96</td>
<td>4.57</td>
<td>2.35</td>
</tr>
</tbody>
</table>
Optimization Methods for Total Variation Based Image Restoration

Input $f$

Output $u$

$\lambda_4 > \lambda > \lambda_5$

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1$</td>
<td>$\lambda_2$</td>
<td>$\lambda_3$</td>
<td>$\lambda_4$</td>
<td>$\lambda_5$</td>
</tr>
<tr>
<td>19.40</td>
<td>13.40</td>
<td>7.96</td>
<td>4.57</td>
<td>2.35</td>
</tr>
</tbody>
</table>
Optimization Methods for Total Variation Based Image Restoration

Input $f$

Output $u$

$\lambda_5 > \lambda > 0$

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1$</td>
<td>$\lambda_2$</td>
<td>$\lambda_3$</td>
<td>$\lambda_4$</td>
<td>$\lambda_5$</td>
</tr>
<tr>
<td>19.40</td>
<td>13.40</td>
<td>7.96</td>
<td>4.57</td>
<td>2.35</td>
</tr>
</tbody>
</table>
Optimization Methods for Total Variation Based Image Restoration

\[
f(\text{TV}/L^1) \Rightarrow u + v
\]

Face illumination correction: Chen-Yin-Zhou-Domaniciu-Huang 06’
<table>
<thead>
<tr>
<th></th>
<th>Barbara 512x512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>TV/L1 1-threaded</td>
</tr>
<tr>
<td>Grayscale</td>
<td>8-bit</td>
</tr>
<tr>
<td>Lambda</td>
<td>1.0</td>
</tr>
<tr>
<td># neighbors</td>
<td>16</td>
</tr>
<tr>
<td>Initial graph construction time</td>
<td>0.48</td>
</tr>
<tr>
<td>Parametric max-flow time</td>
<td>1.37</td>
</tr>
<tr>
<td>Divide-n-conquer max-flow time</td>
<td>2.49</td>
</tr>
</tbody>
</table>

CPU: AMD Opteron 1Ghz, Memory: 3GB Op. System: Linux
<table>
<thead>
<tr>
<th>Image</th>
<th>Noisy Barbara 512x512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>ROF Divide-n-Con.</td>
</tr>
<tr>
<td>Grayscale</td>
<td>8-bit</td>
</tr>
<tr>
<td>Lambda</td>
<td>0.025</td>
</tr>
<tr>
<td># neighbors</td>
<td>16</td>
</tr>
<tr>
<td>Initial graph construction time</td>
<td>0.84</td>
</tr>
<tr>
<td>Parametric max-flow time</td>
<td>3.59</td>
</tr>
<tr>
<td>Divide-n-conquer max-flow time</td>
<td>6.39</td>
</tr>
</tbody>
</table>

CPU: AMD Opteron 1Ghz, Memory: 3GB Op. System: Linux
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Image</strong></td>
<td>CT 512x512</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>TV/L1 1-threaded</td>
</tr>
<tr>
<td><strong>Grayscale</strong></td>
<td>8-bit</td>
</tr>
<tr>
<td><strong>Lambda</strong></td>
<td>0.5</td>
</tr>
<tr>
<td><strong># neighbors</strong></td>
<td>16</td>
</tr>
<tr>
<td><strong>Initial graph construction time</strong></td>
<td>0.78</td>
</tr>
<tr>
<td><strong>Parametric max-flow time</strong></td>
<td>2.47</td>
</tr>
<tr>
<td><strong>Divide-n-conquer max-flow time</strong></td>
<td>4.96</td>
</tr>
</tbody>
</table>

CPU: AMD Opteron 1Ghz, Memory: 3GB Op. System: Linux
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Image</strong></td>
<td>CT 512x512</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>TV/L1 1-threaded</td>
</tr>
<tr>
<td><strong>Grayscale</strong></td>
<td>8-bit</td>
</tr>
<tr>
<td><strong>Lambda</strong></td>
<td>1.0</td>
</tr>
<tr>
<td><strong># neighbors</strong></td>
<td>16</td>
</tr>
<tr>
<td><strong>Initial graph construction time</strong></td>
<td>0.48</td>
</tr>
<tr>
<td><strong>Parametric max-flow time</strong></td>
<td>1.12</td>
</tr>
<tr>
<td><strong>Divide-n-conquer max-flow time</strong></td>
<td>2.33</td>
</tr>
</tbody>
</table>

**CPU:** AMD Opteron 1Ghz, **Memory:** 3GB, **Op. System:** Linux
3D Brain MRI Image, Original, Size: 181x217x181 (1mmx1mmx1mm)
Used as input for TV/L1

T1 3D image:  

z=50

T2 3D image:  

z=50

z=100

z=100
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Image</strong></td>
<td>Brain MRI T1</td>
</tr>
<tr>
<td></td>
<td>181x217x181</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>TV/L1 1-threaded</td>
</tr>
<tr>
<td><strong>Grayscale</strong></td>
<td>8-bit</td>
</tr>
<tr>
<td><strong>Lambda</strong></td>
<td>1.0</td>
</tr>
<tr>
<td><strong># neighbors</strong></td>
<td>6</td>
</tr>
<tr>
<td><strong>Initial graph</strong></td>
<td>construction time</td>
</tr>
<tr>
<td></td>
<td>6.55</td>
</tr>
<tr>
<td><strong>Parametric max-flow</strong></td>
<td>time</td>
</tr>
<tr>
<td></td>
<td>23.18</td>
</tr>
<tr>
<td><strong>Divide-n-conquer</strong></td>
<td>max-flow time</td>
</tr>
<tr>
<td></td>
<td>67.56</td>
</tr>
</tbody>
</table>

**CPU:** AMD Opteron 1Ghz, **Memory:** 3GB, **Op. System:** Linux
3D Brain MRI Image, 5% noise, Size: 181x217x181 (1mmx1mmx1mm) 
Used as input for ROF

T1 3D image: 

\[ z = 50 \] 

\[ z = 100 \] 

T2 3D image: 

\[ z = 50 \] 

\[ z = 100 \]
<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image (noisy)</td>
<td>Brain MRI T1 181x217x181</td>
</tr>
<tr>
<td>Model</td>
<td>ROF Divide-n-Con.</td>
</tr>
<tr>
<td>Grayscale</td>
<td>8-bit</td>
</tr>
<tr>
<td>Lambda</td>
<td>0.3</td>
</tr>
<tr>
<td># neighbors</td>
<td>6</td>
</tr>
<tr>
<td>Initial graph construction time</td>
<td>7.09</td>
</tr>
<tr>
<td>Parametric max-flow time</td>
<td></td>
</tr>
<tr>
<td>Divide-n-conquer max-flow time</td>
<td>41.18</td>
</tr>
</tbody>
</table>

**Notes:**
- **CPU:** AMD Opteron 1Ghz, **Memory:** 3GB, **Op. System:** Linux.
34.12 Divide-n-conquer max-flow time
7.11 Parametric max-flow time
6 # neighbors
0.3 Lambda
8-bit Grayscale
ROF
Divide-n-Con.

<table>
<thead>
<tr>
<th>Image (noisy)</th>
<th>Brain MRI T2 181x217x181</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>ROF Divide-n-Con.</td>
</tr>
<tr>
<td>Grayscale</td>
<td>8-bit</td>
</tr>
<tr>
<td>Lambda</td>
<td>0.3</td>
</tr>
<tr>
<td># neighbors</td>
<td>6</td>
</tr>
<tr>
<td>Initial graph construction time</td>
<td>7.11</td>
</tr>
<tr>
<td>Parametric max-flow time</td>
<td></td>
</tr>
<tr>
<td>Divide-n-conquer max-flow time</td>
<td>34.12</td>
</tr>
</tbody>
</table>

CPU: AMD Opteron 1Ghz, Memory: 3GB Op. System: Linux
Iterative regularization

$$\min_u \{ J(u) + H(u, f) \}$$

$J$ and $H$ are convex functionals of $u$, $J(u) \geq 0, J(0) = 0, H(u, f) \geq 0$

Iterate

$$u_k = \arg \min_{u \in BV(\Omega)} \{ J(u) + H(u, f) - \langle u, p_{k-1} \rangle \}$$

where subgradient $p_0 = 0 \in \partial J(u_0), p_{k-1} \in \partial J(u_{k-1})$
Bregman Distance

\[ D(u, v) = J(u) - J(v) - \langle p, u - v \rangle \]

for \( p \in \partial J(u) \), is the Bregman distance assoc. with \( J(\cdot) \)

- \( D(u, v) \geq 0 \) and \( D(u, v) = 0 \) iff \( u = v \) (for \( J(\cdot) \) str. convex)
- \( D(u, v) \neq D(v, u) \) in general
- \( \Delta \) inequality does not hold
Convergence Analysis

- \( \{H(u_k, f)\} \) is monotonically non-increasing
  \[
  H(u_k, f') \leq H(u_k, f) + D(u_k, u_{k-1}) \leq H(u_{k-1}, f')
  \]

- MAIN THEOREM
  If (i) \( g \) is the true noise free image
    (ii) \( H(g, g) = 0 \)
    (iii) \( H(g, f') \leq \delta^2 \) (\( \delta^2 \) : noise level)
  then the distance between \( u_k \) and \( g \) decreases, i.e.,
  \[
  D(g, u_k) \leq D(g, u_k) + D(u_k, u_{k-1}) \leq D(g, u_{k-1})
  \]
as long as \( H(u_k, f') > \delta^2 \)
One-step ROF

(a) original  
(b) noisy f, SNR=14.8  
(c) noise+128, δ=10.0

(d) u: ROF  
(e) f−u+128, ||f−u||_{L^2}=10.2  
(f) (f−u)−noise+128

ROF with $\lambda = 0.085$, $||f−u||_{L^2} \approx \delta$, signal contained in $v = f − u$.  
Iterative ROF regularization

ROF with iterated refinement with $\delta = 10$ and $\lambda = 0.013$
Best resolution obtained at $k = 4$. Noise returns in $u_5, u_6, \ldots$
Iterated ROF regularization
Deblurring + Denoising

Gaussian blur/noise, $\delta=10$ and $\lambda=0.1$