A Parallel Linearized ADMM with Application to Multichannel TGV-Based Image Restoration

Chuan He, Changhua Hu

the High-tech Institute of Xi’an

Xuelong Li

Xi’an Institute of Optics and Precision Mechanics
Outline

- Introduction
- Proposed Method
- Multichannel TGV-Based Image Restoration
- Concluding Remarks
Introduction

Blur model

\[ f = Ku + n \]

The key to the success of image restoration:

- Regularization model incorporating image prior knowledge
- Automatic, accurate, concise, and fast solution algorithm
Convex Objective Function for Imaging Inverse Problems:

\[ \min_{x \in X} g(x) + \sum_{h=1}^{H} f_h(L_h x) \quad (1) \]

- \( g \) and \( f_h \) are convex functions whose proximity operators possess closed-forms or at least can be solved efficiently by existent methods;
- \( L_h \) is a bounded linear operator with adjoint \( L_h^* \).

The solution of (1) usually suffers from two aspects:

- Data space \( X \) in a practical application is typically of high dimension;
- Function \( g \) and the linear-operator-coupled \( f_h \) may be nondifferentiable.
Our Strategy-Parallel LADMM with “full splitting”:

\[
\min_{x \in X} g(x) + \sum_{h=1}^{H} f_h(L_h x)
\]  

- At each iteration, only the proximity operators of the convex functions and the linear operators are involved. It possesses a highly parallel structure and can be accelerated by parallel calculation techniques.
- The linear inverse operator, which usually exists in methods dealing with inverse problems, is excluded. It is not partial to a particular data boundary condition.
- It achieves a worst-case \(O(1/k)\) convergence rate by exploiting only the first-order information of the functions.
Proposed Method

Similar to ADMM, the AL functional of (1) is as follows

$$
L_A(x, a_1, \ldots, a_H; v_1, \ldots, v_H) = g(x) + \sum_{h=1}^{H} \left( f_h(a_h) + \langle v_h, L_h x - a_h \rangle + \frac{\beta_h}{2} \| L_h x - a_h \|^2 \right). \quad (2)
$$

- $v_h$ is the Lagrange multiplier and $\beta_h > 0$ is the penalty parameter.

The proposed PLADMM finding the saddle point of (2):

$$
\begin{align*}
    a_h^{k+1} &= \text{prox}_{f_h/\beta_h} \left( L_h x^{k+1} + \frac{v_h}{\beta_h} \right), \quad h = 1, \ldots, H; \\
    v_h^{k+1} &= v_h^k + \beta_h \left( L_h x^{k+1} - a_h^{k+1} \right), \quad h = 1, \ldots, H; \\
    x^{k+1} &= \text{prox}_{tg} \left( x^k - t \sum_{h=1}^{H} \beta_h L_h^* \left( L_h x^k - a_h^{k+1} + \frac{v_h^{k+1}}{\beta_h} \right) \right), \quad 0 < t \leq \left( 1 / \sum_{h=1}^{H} \beta_h \| L_h^* L_h \| \right).
\end{align*}
$$
Proposed Method

With the Moreau decomposition in convex analysis:

$$\text{prox}_{\beta f^*} v = v - \beta \text{prox}_{f/\beta} \left( \frac{v}{\beta} \right)$$

The iterative scheme of PLADMM is transformed into:

$$\begin{cases} 
\nu_h^{k+1} = \text{prox}_{\beta_h f^*_{h}} \left( \beta_h L_h \nu_h^{k+1} + \nu_h^{k} \right) & h = 1, \ldots, H; \\
\nu^{k+1} = \text{prox}_{t g} \left( \nu^{k} - t \sum_{h=1}^{H} L^*_h \left( 2 \nu_h^{k+1} - \nu_h^{k} \right) \right), & 0 < t \leq \left( \frac{1}{\sum_{h=1}^{H} \beta_h \|L^*_h L_h\|} \right).
\end{cases}$$
According to the convergence analysis of LADMM (Theorem 1 in the paper)

\[ \{x^k, a^k_1, \ldots, a^k_H, v^k_1, \ldots, v^k_H\} \] converges to a saddle point of

\[ \mathcal{L}_A (x, a_1, \ldots, a_H; v_1, \ldots, v_H); \]

\[ \{x^k\} \] converges to a solution of problem (1);

PLADMM possesses a worst-case \( O \left( \frac{1}{k} \right) \) convergence rate.
Multichannel TGV-Based Image Restoration

Objective Function:

\[
(u^*, p^*) = \arg\min_{u, p} \alpha_1 \| \nabla u - p \|_1 + \alpha_2 \| \mathcal{E}p \|_1
\]

\[
s.t. \quad \{ u \in \Omega \triangleq \{ u : 0 \leq u \leq 255 \} \cap \Psi \triangleq \{ u : \| Ku - f \|_2^2 \leq c \}.\]

PLADMM Scheme:

\[
\begin{align*}
\mathbf{v}_{1,i,j,l}^{k+1} &= P_{\mathbb{B}_{\alpha_1}} \left( \beta_1 \left( \nabla u^k_{i,j,l} - p^k_{i,j,l} \right) + \mathbf{v}_{1,i,j,l}^k \right) \\
\mathbf{v}_{2,i,j,l}^{k+1} &= P_{\mathbb{B}_{\alpha_2}} \left( \beta_2 \left( \mathcal{E}p^k_{i,j,l} + \mathbf{v}_{2,i,j,l}^k \right) + \mathbf{v}_{2,i,j,l}^k \right) \\
\mathbf{v}_{3}^{k+1} &= \beta_3 S_{\sqrt{c}} \left( \frac{\mathbf{v}_{3}^k}{\beta_3} + Ku^k - f \right) \\
\mathbf{u}^{k+1} &= P_{\Omega} \left( u^k - t \left( \nabla^T \mathbf{v}_{1}^{k+1} + K^T \tilde{\mathbf{v}}_{3}^{k+1} \right) \right) \\
p_{1}^{k+1} &= p_{1}^k - t \left( \nabla^T \mathbf{v}_{1}^{k+1} + \nabla^T \tilde{\mathbf{v}}_{2,1}^{k+1} - \tilde{\mathbf{v}}_{1,1}^{k+1} \right) \\
p_{2}^{k+1} &= p_{2}^k - t \left( \nabla^T \mathbf{v}_{2}^{k+1} + \nabla^T \tilde{\mathbf{v}}_{2,2}^{k+1} - \tilde{\mathbf{v}}_{1,2}^{k+1} \right) \\
\tilde{\mathbf{v}}_{h}^{k+1} &= 2\mathbf{v}_{h}^{k+1} - \mathbf{v}_{h}^{k}
\end{align*}
\]
Multichannel TGV-Based Image Restoration

The experiment was performed in MATLAB on a PC with an Intel Core i5 CPU (3.20GHz) and 8GB of RAM.

Images: Lena (256 × 256), Peppers (512 × 512), and Monarch (768 × 512)
Multichannel TGV-Based Image Restoration

<table>
<thead>
<tr>
<th>Problem</th>
<th>Image</th>
<th>Blur Kernels</th>
<th>$\sigma$</th>
<th>PSNR</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lena</td>
<td>Set 1</td>
<td>3</td>
<td>20.05</td>
<td>0.5239</td>
</tr>
<tr>
<td>2</td>
<td>Peppers</td>
<td>Set 2</td>
<td>8</td>
<td>17.55</td>
<td>0.5140</td>
</tr>
<tr>
<td>3</td>
<td>Monarch</td>
<td>Set 3</td>
<td>10</td>
<td>17.95</td>
<td>0.4608</td>
</tr>
</tbody>
</table>

The three blurs are generated: (1). Generate 9 kernels: \{A(13), A(15), A(17), G(11, 9), G(21, 11), G(31, 13), M(21, 45), M(41, 90), M(61, 135)\}; (2). Assign the above 9 kernels to \{K11, K12, K13; K21, K22, K23; K31, K32, K33\}; (3). then with the above kernels, we generate the final three sets of blurs for comparison by multiplying relative weights \{1, 0, 0; 0, 1, 0; 0, 0, 1\} (Set 1), \{0.6, 0.2, 0.2; 0.15, 0.7, 0.15; 0.1, 0.1, 0.8\} (Set 2), and \{0.7, 0.15, 0.15; 0.1, 0.8, 0.1; 0.2, 0.2, 0.6\} (Set 3) to the corresponding blur kernels.
## Multichannel TGV-Based Image Restoration

### Comparison in PSNR, SSIM, and CPU time

<table>
<thead>
<tr>
<th>Problem</th>
<th>Method</th>
<th>PSNR</th>
<th>SSIM</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PLADMM-TGV</td>
<td>26.21</td>
<td>0.7680</td>
<td>32.24</td>
</tr>
<tr>
<td></td>
<td>APEADMM-TGV</td>
<td>26.21</td>
<td>0.7649</td>
<td>35.62</td>
</tr>
<tr>
<td></td>
<td>FTVD-v4</td>
<td>26.04</td>
<td>0.7583</td>
<td>10.37</td>
</tr>
<tr>
<td>2</td>
<td>PLADMM-TGV</td>
<td>25.57</td>
<td>0.7632</td>
<td>155.46</td>
</tr>
<tr>
<td></td>
<td>APEADMM-TGV</td>
<td>25.56</td>
<td>0.7623</td>
<td>199.93</td>
</tr>
<tr>
<td></td>
<td>FTVD-v4</td>
<td>25.25</td>
<td>0.7507</td>
<td>54.69</td>
</tr>
<tr>
<td>3</td>
<td>PLADMM-TGV</td>
<td>23.85</td>
<td>0.8083</td>
<td>234.45</td>
</tr>
<tr>
<td></td>
<td>APEADMM-TGV</td>
<td>23.83</td>
<td>0.8063</td>
<td>302.68</td>
</tr>
<tr>
<td></td>
<td>FTVD-v4</td>
<td>23.61</td>
<td>0.7965</td>
<td>84.43</td>
</tr>
</tbody>
</table>

APEADMM: He2014(IEEE-TIP)
PSNR=17.55, SSIM=0.5140

Degraded

PSNR=25.56, SSIM=0.7623

APEADMM-TGV

PSNR=25.57, SSIM=0.7632

PLADMM-TGV

PSNR=25.25, SSIM=0.7507

FTVD-v4
Some Related Works


The related MATLAB codes can be found on my Researchgate.
Conclusion

- Full Splitting
  - More Parallel

- Excluding Matrix Inverse Operation
  - More concise and more fast

- Extension
  - Other regularizations and other image inverse problems
THANK YOU!

Dr. Chuan He
hechuan8512@163.com