Fast Distributed Total Variation

Samuel Vaiter

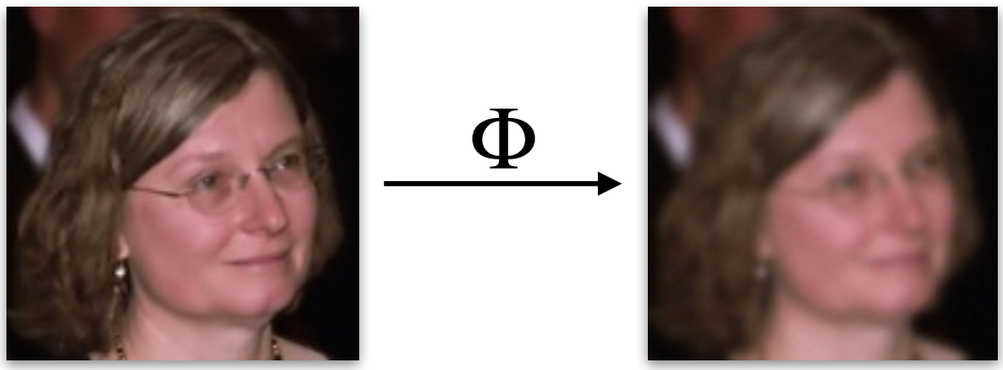
March 30, 2015

ÉCOLE POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

Total Variation

Linear inverse problems

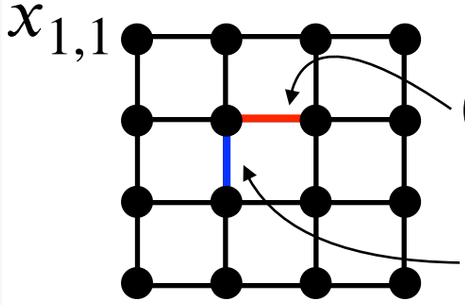
$$y = \Phi x_0 + w$$



Total Variation regularization [Rudin-Osher-Fatemi 1992]

$$x^\star \in \operatorname*{argmin}_{x \in \mathbb{R}^N} \frac{1}{2}\|y - \Phi x\|_2^2 + \lambda J(x)$$

$$J(x) = \|\nabla x\|_{1,2} = \sum_{i,j} \sqrt{(\partial_\rightarrow x)_{i,j}^2 + (\partial_\uparrow x)_{i,j}^2}$$



$$(\partial_\rightarrow x)_{i,j} = |x_{i,j} - x_{i+1,j}|$$

$$(\partial_\uparrow x)_{i,j} = |x_{i,j} - x_{i,j+1}|$$

This talk : How to distribute this minimization ?

Proximal Method

$$x^\star \in \operatorname*{argmin}_{x \in \mathbb{R}^N} \frac{1}{2} \underbrace{\|y - \Phi x\|_2^2}_{F(x)} + \lambda J(x)$$

Forward-backward

$$x^{(k+1)} = \operatorname{prox}_{\lambda J}(x^{(k)} - \gamma \nabla F(x))$$

Proximity operator

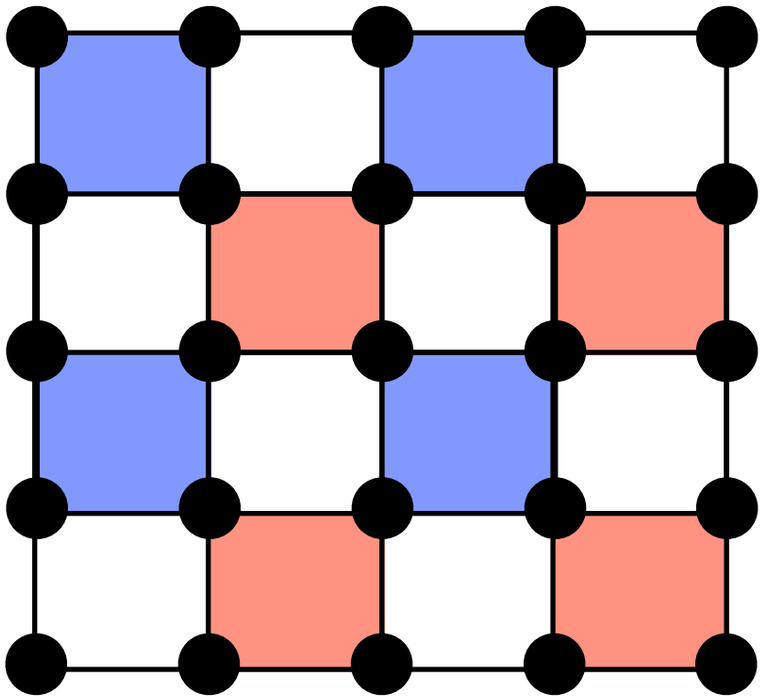
$$\operatorname{prox}_{\lambda J}(x) = \operatorname*{argmin}_{z \in \mathbb{R}^N} \frac{1}{2}\|x - z\|_2^2 + \lambda J(z)$$

→ no closed formula for TV

- Row/Column splitting [Condat 2013]
- Augmented method [Chambolle-Pock 2011]

A Splitting Scheme

$$\mathrm{prox}_{\lambda J}(x) = \underset{z \in \mathbb{R}^N}{\mathrm{argmin}} \; \frac{1}{2}\|x - z\|_2^2 + \lambda J(z)$$



$$J(x) = J_{\text{even}}(x_{\text{even}}) + J_{\text{odd}}(x_{\text{odd}})$$

Primal-dual problem

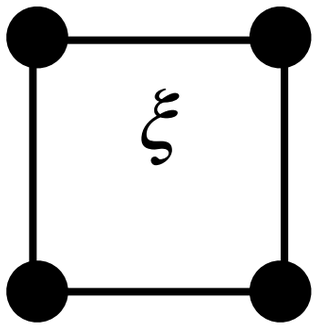
$$\max_{x_{\text{even}}, x_{\text{odd}}} \; -J_{\text{even}}{}^*(x_{\text{even}}) - J_{\text{odd}}{}^*(x_{\text{odd}}) - \frac{1}{2\lambda}\|x_{\text{even}} + x_{\text{odd}}\|^2 + \langle x_{\text{even}} + x_{\text{odd}},\, y\rangle$$

↓ alternate minimization

fully decomposable on each square

Minimization Over a Square

Dual problem $\in \mathbb{R}^4$



$$\xi^\star \in \underset{\|\xi\|_2 \leq 1}{\operatorname{argmin}} \; \| \operatorname{div} \xi + \bar{y} \|_2^2$$

$$\langle \nabla x,\, \xi \rangle = -\langle x,\, \operatorname{div} \xi \rangle$$

$$x^\star = \operatorname{div} \xi^\star$$

→ Explicit descent

→ **Newton minimization**

Empirical observation: **one** Newton step is enough

+ reprojection on the unit ball

Algorithm

While dual gap big enough

- Over-relaxation (FISTA-like acceleration)
- Minimize over every even square
- Minimize over every odd square
- Compute dual gap

easy distributed calculus

reduction step

→ Implementation in CUDA (2D and 3D)

(not open-source yet)

Preliminary Benchmark

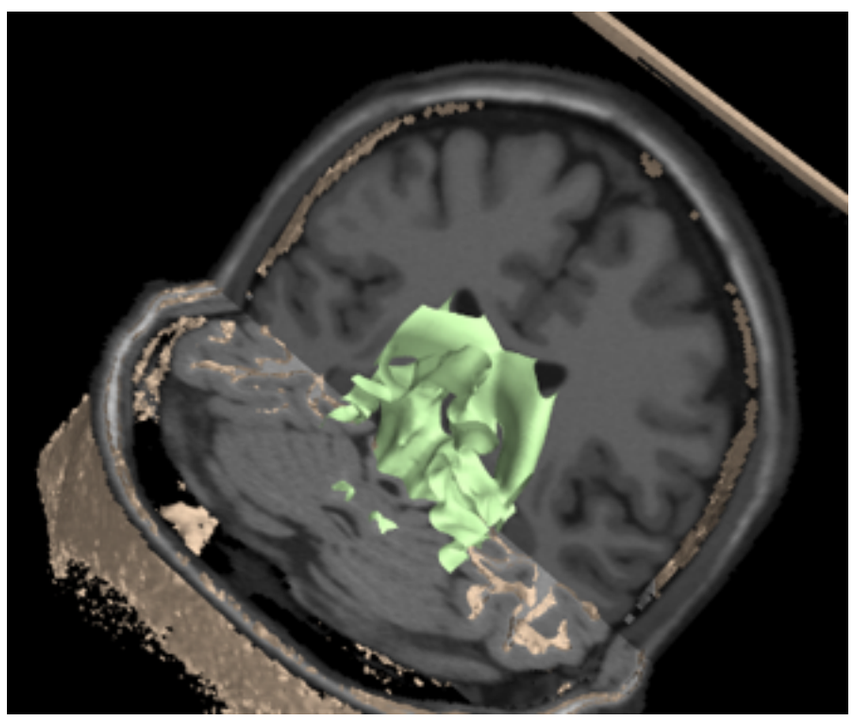
@Xeon E5-2670 / Tesla K20m (linux 3.12, CUDA 5.5)

2D

dimension of the problem

parameter (ms)	256^2	512^2	1024^2	2048^2	4096^2	8192^2
1.0	0.4	0.5	0.9	2.3	7.9	79
5.0	1.8	2.6	4.3	13	63	379
10.0	3.9	5.7	11	39	170	892
20.0	9.3	12	24	92	406	1961

3D



181 x 217 x 181
35 ms

~100-1000x faster
than CPU CP

Thanks for your attention

Any question ?