A Parallel Approach for Image Segmentation by Numerical Minimization of a Second-Order Functional

Riccardo Zanella [1] Federica Porta [1] Massimo Zanetti [2] Valeria Ruggiero [1]

[1]Mathematics and Computer Science Dept., University of Ferrara [2]Information and Comm. Technology Dept., University of Trento

UNIVERSITÀ DEGLI STUDI DI FERRARA - EX LABORE FRUCTUS -

Introduction

In this work we are concerned with Blake–Zissermann[1] (BZ) variational method for image segmentation: this approach leads to a minimization of a non-convex objective energy functional.
Very often, the segmentation of large-size gridded data is addressed via tiling procedure: additional specific post-processing on tile boundaries may be needed in order to reduce the effect of the subdivision.
We aim to show that a simple tiling strategy enables us to treat large images even in a commodity multicore CPU, with no need of specific post-processing on tile junctions.

Blake-Zissermann continuos model

Continuous model can be stated as:

$$\begin{aligned}\mathcal{F}_\epsilon(s,z,u) = {} & \delta \int_\Omega z^2|\nabla^2 u|^2\,dx + \xi_\epsilon \int_\Omega (s^2+o_\epsilon)|\nabla u|^2 dx \\ & +(\alpha-\beta)\int_\Omega\left(\epsilon|\nabla s|^2+\frac{1}{4\epsilon}(s-1)^2\right)dx + \beta\int_\Omega\left(\epsilon|\nabla z|^2+\frac{1}{4\epsilon}(z-1)^2\right)dx \\ & +\mu\int_\Omega |u-g|^2\,dx,\end{aligned}$$

where $\Omega \subset \mathbb{R}^2$ is a rectangular domain and $g \in L^\infty(\Omega)$ is a given image. Here $\delta, \alpha, \beta, \mu$ are positive parameters $(2\beta \geq \alpha \geq \beta)$ and the terms depending on ϵ are infinitesimals.

Ambrosio-Faina-March discrete model

In [2] a discrete approximation of BZ functional is proposed; this function is not globally convex, but it is quadratic and strongly convex w.r.t. each block of variables $(\mathbf{s},\mathbf{z},\mathbf{u})$.

- when fixing $\mathbf{u}$:

$$F_\epsilon(\mathbf{s},\mathbf{z},\mathbf{u}) = t_x t_y\left\{\frac{1}{2}\begin{pmatrix}\mathbf{s}^T & \mathbf{z}^T\end{pmatrix}\begin{pmatrix}\mathbf{A}_1 & 0\\ 0 & \mathbf{A}_2\end{pmatrix}\begin{pmatrix}\mathbf{s}\\ \mathbf{z}\end{pmatrix} - \begin{pmatrix}\mathbf{s}^T & \mathbf{z}^T\end{pmatrix}\begin{pmatrix}\mathbf{b}_1\\ \mathbf{b}_2\end{pmatrix} + \mathbf{c}_{sz}\right\}$$

  where:
  - $\mathbf{A}_1, \mathbf{A}_2$ depend on $\mathbf{u}$;
  - $\mathbf{b}_1$ depends on boundary conditions on $\mathbf{s}$;
  - $\mathbf{b}_2$ depends on boundary conditions on $\mathbf{z}$;
- when fixing $\mathbf{s},\mathbf{z}$:

$$F_\epsilon(\mathbf{s},\mathbf{z},\mathbf{u}) = t_x t_y\left\{\frac{1}{2}\mathbf{u}^T\mathbf{A}_3\,\mathbf{u} - \mathbf{u}^T\mathbf{b}_3 + \mathbf{c}_u\right\}$$

  where:
  - $\mathbf{A}_3$ depends on $\mathbf{s},\mathbf{z}$,
  - $\mathbf{b}_3$ depends on $\mathbf{s},\mathbf{z}$, on boundary conditions on $\mathbf{u}$, and on measured image $\mathbf{g}$.

BCDA sequential approach

In [3], the numerical minimization of $F_\epsilon(\mathbf{s},\mathbf{z},\mathbf{u})$ is obtained by a block coordinate descent algorithm (BCDA). Starting from an initial vector $\mathbf{x}^0 = (\mathbf{s}^0,\mathbf{z}^0,\mathbf{u}^0)$, for each block variable ($\mathbf{s}$, $\mathbf{z}$ or $\mathbf{u}$) a descent direction $\mathbf{d}$ is cyclically determined by few iterations of a preconditioned conjugate gradient (PCG) method applied to the quadratic subproblem; furthermore, a Cauchy step–length along $\mathbf{d}$ is exploited.

OPARBCDA parallel approach

In view of the local features of F_ϵ, a natural way to address its minimization is to split the image into p tiles T_j, $j = 1,\dots,p$, inducing a partition of the variables $\mathbf{s},\mathbf{z},\mathbf{u}$ into p blocks $\mathbf{x}_1,\mathbf{x}_2,\dots,\mathbf{x}_p$, with $\mathbf{x}_j \equiv (\mathbf{s}|_{T_j},\mathbf{z}|_{T_j},\mathbf{u}|_{T_j}) \in \mathbb{R}^{n_j}$, $\sum_{j=1}^p n_j = 3NM$. In order to avoid side effects on the tile junctions, we enlarge each $N_j \times M_j$ tile T_j with an outer frame of ν rows and columns of pixels; more precisely, the whole image is splitted into partially overlapping p tiles S_j of size $(N_j+2\nu)\times(M_j+2\nu)$, where ν is the number of overlapping pixels and $S_j \supset T_j$.

Parallel implementation

At each outer iteration, Step 1 consists of a number of independent tasks that can be concurrently solved. Manager/workers pattern ensures run-time distribution of independent tasks among POSIX threads: mutex-protected queues collect both task input and output results: a number C of computational threads (workers) is initialized and put on wait on a shared task queue, while a monitor thread (master) is responsible to extract, for each subproblem j, initial data $\mathbf{w}_j^0$ from current solution $\mathbf{x}^\ell$ and collect subproblems computed solutions. As regards Step 2.2, OpenMP compiler directive *omp parallel for* is used for evaluation of $F_\epsilon(\mathbf{y})$.

OPARBCDA method

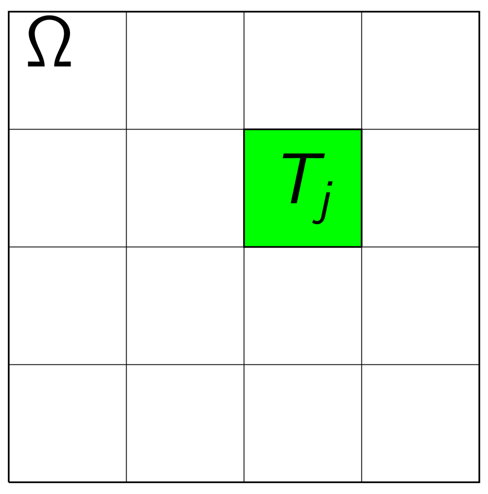


Figure 1: OPARBCDA tiling procedure.

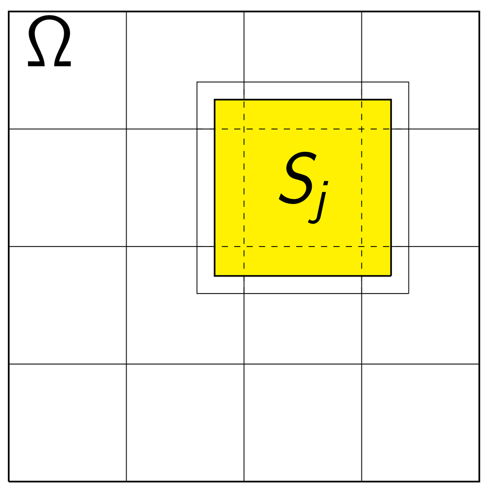


Figure 2: OPARBCDA enlarged tile extraction.

Algorithm 1 OPARBCDA

Step 0: Given $\mathbf{x}^0$, the partitions $\{T_1,\dots,T_p\}$ and $\{S_1,\dots,S_p\}$ of Λ such that $S_j \supset T_j$, $B_j = S_j - T_j$, $j = 1,\dots,p$ and $\{\theta_\ell\}$, such that $\underline{\theta} < \theta_\ell \leq \overline{\theta}$, $\ell \geq 0$ and an exit tolerance θ_{outer}, $\ell = 0$;

Step 1: for $j = 1,\dots,p$

1.1 if $\nabla_{x_{T_j}} F_\epsilon(\mathbf{x}^\ell) \neq 0$ then

- compute $\mathbf{y}^j = (\mathbf{x}_1^\ell,\dots,\mathbf{x}_{j-1}^\ell,\overline{\mathbf{x}}_j,\mathbf{x}_{j+1}^\ell,\dots,\mathbf{x}_p^\ell)$;
  - (a) set $\mathbf{x}_{S_j}^0 = (\mathbf{x}^\ell)|_{S_j}$, $k = -1$,
  - (b) repeat:
    $k = k+1$; compute $\mathbf{x}_{S_j}^{k+1}$ by a step of BCDA; extract $\mathbf{x}_{T_j}^{k+1}$; set $\overline{\mathbf{x}}_j = \mathbf{x}_{T_j}^{k+1}$;
    if $f_{\mathbf{x}_{S_j}}(\mathbf{x}_{T_j}^k;\mathbf{x}_{B_j}^0) - f_{\mathbf{x}_{S_j}}(\mathbf{x}_{T_j}^{k+1};\mathbf{x}_{B_j}^0) < \lambda_{min}\|\mathbf{x}_{T_j}^k - \mathbf{x}_{T_j}^{k+1}\|^2$ then
    $\overline{\mathbf{x}}_j = \mathbf{x}_{T_j}^k$ exit next j; end
    until $\|\nabla_{\mathbf{x}_{S_j}} f_{\mathbf{x}_{S_j}}(\mathbf{x}_{S_j}^{k+1})\| \leq \theta_\ell\|\mathbf{x}_{S_j}^{k+1} - \mathbf{x}_{S_j}^\ell\|$

else

- $\mathbf{y}^j = \mathbf{x}^\ell$;

end

Step 2: define the new iterate $\mathbf{x}^{\ell+1}$:

2.1 compute $F_\epsilon(\overline{\mathbf{y}})$ where $\overline{\mathbf{y}} = (\overline{\mathbf{x}}_1,\overline{\mathbf{x}}_2,\dots,\overline{\mathbf{x}}_p)$

2.2 update $\mathbf{x}^{\ell+1} = \text{argmin}\{F_\epsilon(\overline{\mathbf{y}}),F_\epsilon(\mathbf{y}^1),\dots,F_\epsilon(\mathbf{y}^p)\}$.

Step 3: if $(F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\mathbf{x}^{\ell+1}) \leq \theta_{outer}F_\epsilon(\mathbf{x}^{\ell+1})$ then stop; else $\ell = \ell+1$ and go to Step 1.

Numerical evaluation

We considered a 2020×2020 image and compared the solution $\mathbf{x}^s = (\mathbf{u}^s,\mathbf{s}^s,\mathbf{z}^s)$ obtained by BCDA on the whole dataset and the one $\mathbf{x}^t = (\mathbf{u}^t,\mathbf{s}^t,\mathbf{z}^t)$ computed by OPARBCDA, splitting the image into t=8×8 and t=16×16 tiles. BCDA is stopped when the relative difference of F_ϵ at two successive iterates is less than 1e-03, while OPARDCDA exits when the current value of F_ϵ is less or equal than the minimum achieved by BCDA. We performed runs with up to 15 workers plus one monitor, while for Step 2 parallelization we set the number of OpenMP threads equal to C+1: this approach would ensure a total number of active threads equal to C+1 at each parallelized step of the algorithm.

	F_ϵ	rel.err	ext. it.	time [s]
ground truth solution	8.693e+07			
BCDA	8.736e+07	5.006e-03		102.4
8×8 tile grid				
OPARBCDA $\nu = 0$	8.732e+07	4.511e-03	7	31.9 (C=15)
OPARBCDA $\nu = 4$	8.709e+07	1.929e-03	2	25.8 (C=15)
OPARBCDA $\nu = 8$	8.708e+07	1.760e-03	2	27.9 (C=15)
16×16 tile grid				
OPARBCDA $\nu = 0$	8.735e+07	4.858e-03	74	52.5 (C=15)
OPARBCDA $\nu = 4$	8.710e+07	1.957e-03	3	15.1 (C=15)
OPARBCDA $\nu = 8$	8.710e+07	1.955e-03	3	17.4 (C=15)

Table 1: Reference BCDA and OPARBCDA comparison.

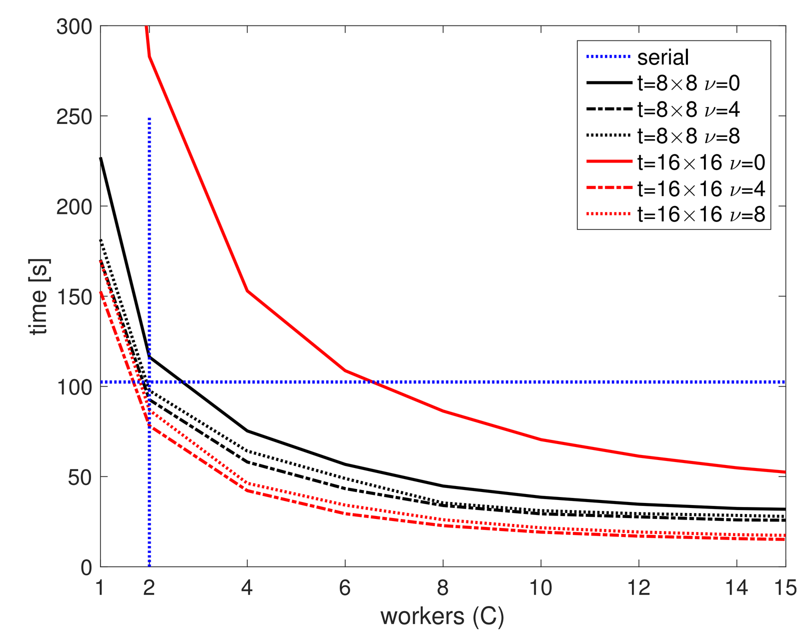


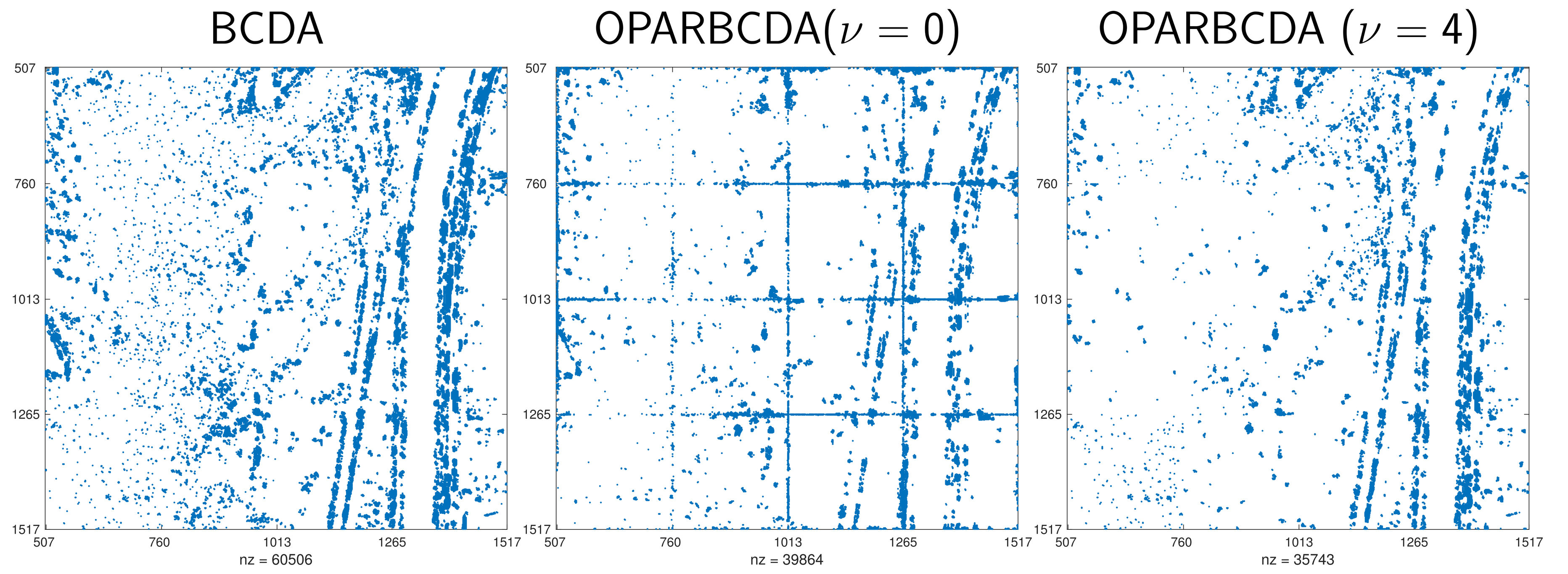
Figure 3: Computational time for parallel OPARBCDA w.r.t the number C of workers.

Test image available at: www.territorio.provincia.tn.it/portal/server.pt/community/lidar/847/lidar/23954
Test platform consists of a commodity PC equipped with a dual-head Intel(R) Xeon CPU E5-2630 at 2.4 GHz with 256 GB of RAM, running CentOS Linux release 7.2 and Intel compiler 16.0.

Accuracy on tile junctions

Entries of central portions of $|\mathbf{z}^t - \mathbf{z}^*| > 0.01$ with t =8×8, $\nu = 0$ and $\nu = 4$.

BCDA OPARBCDA($\nu = 0$) OPARBCDA ($\nu = 4$)



References and Acknowledgements

[1] A. Blake and A. Zisserman. MIT Press, Cambridge, MA, 1987.

[2] L. Ambrosio, L. Faina, and R. March. *SIAM J. Math. Anal.*, 32:1171–1197, 2001.

[3] M. Zanetti, V. Ruggiero, and M. Jr. Miranda. *Commun. Nonlinear Sci. Numer. Simul.*, 36:528–548, 2016.

This research was supported by: INDAM-GNCS2016, FIRB2012 grant RBFR12M3AC.