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

Riccardo Zanella[1,a)], Massimo Zanetti[2,b)] and Valeria Ruggiero[1,c)]

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

[a)]Corresponding author: riccardo.zanella@unife.it
[b)]massimo.zanetti@unitn.it
[c)]valeria.ruggiero@unife.it

Abstract. Because of its attractive features, image segmentation has shown to be a promising tool in remote sensing. A known drawback about its implementation is computational complexity. Recently in [1] an efficient numerical method has been proposed for the minimization of a second-order variational approximation of the Blake-Zissermann functional. The method is an especially tailored version of the block-coordinate descent algorithm (BCDA). In order to enable the segmentation of large-size gridded data, such as Digital Surface Models, we combine a domain decomposition technique with BCDA and a parallel interconnection rule among blocks of variables. 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 tiles junctions. From the point of view of the performance, little computational effort is required to separate data in subdomains and the running time is mainly spent in concurrently solving the independent subproblems. Numerical results are provided to evaluate the effectiveness of the proposed parallel approach.

INTRODUCTION

The increasing availability of remote sensing data has stimulated the development of new techniques to analyze and understand collected data more effectively and efficiently. A crucial task in remote sensing data analysis is the image segmentation. Very often, the segmentation of large-size gridded data, such as Digital Surface Models (DSMs) [2], is addressed via tiling procedure. According to the criteria that are used to merge the segmentation results obtained on the tiles, there may be need of specific post-processing on tile boundaries in order to reduce the effect of the subdivision. Indeed, typical approaches to segmentation globally depend on data; as consequence, the segmentation restricted to sub-portions of the input data may introduce undesired artifacts. Here we are concerned with variational methods to segmentation, such as Mumford–Shah (MS) [3] and Blake–Zissermann (BZ) [4], and in particular we focus on the latter. In this variational framework, the segmentation problem is formulated as a non-convex optimization problem. Popular methods for the numerical minimization always return sub-optimal solutions (local minimizers), strongly dependent on the initialization. Therefore the global constraint on the solution is weakened and local features of the initial datum and of objective function become dominant. So far, we consider the minimization of a BZ functional approximation [5], given by

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

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. For the numerical treatment of $\mathcal{F}_\epsilon$ we propose a parallel method, named OPARBCDA, based on the decomposition of the image into partially overlapping tiles. At any iteration, this scheme computes for any tile an inexact minimizer of the objective function restricted to the tile, extracting from this vector only the entries related to the inner pixels of the tile. By a suitable parallel interconnection updating rule for the new iterate, we obtain a decreasing sequence for the objective functional and we can state that the limit points of the sequence of the iterates are stationary points. Moreover, the computed solution appears not affected by the tile junctions without any need of further processing on these regions. Consequently, by a suitable

parallel implementation, based on dynamic task distribution among the available cores, we can efficiently treat large images even in a commodity multicore CPU.

NUMERICAL MINIMIZATION OF THE APPROXIMATION OF BLAKE–ZISSERMAN FUNCTIONAL

The numerical minimization of $\mathcal{F}_\epsilon$ exploits a discrete version of the functional. The domain Ω of the image g is discretized into a lattice of points $\Lambda = \{(it_x, jt_y); i = 1, \ldots, N,\ j = 1, \ldots, M\}$. Assuming zero boundary conditions for the functions s, z, u and using first and second-order difference schemes to approximate differential operators, we obtain the discrete version $F_\epsilon(\mathbf{s}, \mathbf{z}, \mathbf{u})$ of the functional $\mathcal{F}_\epsilon(s, z, u)$, where $\mathbf{s}, \mathbf{z}, \mathbf{u}$ denotes the NM vectors of the column-wise ordered approximate values of s, z, u at the points of Λ. Globally $F_\epsilon(\mathbf{s}, \mathbf{z}, \mathbf{u})$ is not convex, but it is quadratic and strongly convex with respect to both $\mathbf{s}$ and $\mathbf{z}$ when $\mathbf{u}$ is fixed or with respect to $\mathbf{u}$ when $\mathbf{s}$ and $\mathbf{z}$ are fixed. Furthermore F_ϵ is continuously differentiable and coercive in $\mathbb{R}^{3NM}$. Further details are in [1].

A sequential approach

In [1], the numerical minimization of $F_\epsilon(\mathbf{s}, \mathbf{z}, \mathbf{u})$ is obtained by a version of block coordinate descent algorithm [6], especially tailored to exploit the features of the functional. Starting from an initial vector $\mathbf{x}^0 = (\mathbf{s}^0, \mathbf{z}^0, \mathbf{u}^0)$, the basic idea of the method, named BCDA, is to cyclically determine for each block variable ($\mathbf{s}$, $\mathbf{z}$ or $\mathbf{u}$) a descent direction $\mathbf{d}$ by few iterations of a preconditioned conjugate gradient (PCG) method applied to the quadratic and strongly convex subproblem in this block with the other block variables fixed. Further, the step–length along $\mathbf{d}$ is easily obtained by the rule that provides the exact one-dimensional minimizer. BCDA method equipped with a suitable stopping rule for the inner PCG generates a sequence $\{\mathbf{x}^k\}$ such that its limit points in the level set $\mathcal{L}_0 = \{\mathbf{x} : F_\epsilon(\mathbf{x}) \leq F_\epsilon(\mathbf{x}^0)\}$ are stationary points of F_ϵ and $\nabla F_\epsilon(\mathbf{x}^k) \to 0$ as $k \to \infty$ [6, Th. 7.1].

A 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, $j = 1, ..., p$, inducing a partition of the variables $\mathbf{s}, \mathbf{z}, \mathbf{u}$ into p blocks $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_p$, 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 $S_j \supset T_j$. Denoting by $\mathbf{w}_j$ the vector of variables related to S_j, we observe that $\mathbf{x}_j$ is a sub-vector of $\mathbf{w}_j$, i. e. $\mathbf{x}_j = \mathbf{w}_j|_{T_j}$. Inspired by the parallel connection schemes in [6, 7], we propose a parallel block coordinate descent algorithm, named OPARBCDA and detailed in Fig. 1. Starting from an initial vector $\mathbf{x}^0$, at any ℓ-iteration, the algorithm computes p points $\mathbf{y}^j = (\mathbf{x}_1^\ell, ..., \mathbf{x}_{j-1}^\ell, \overline{\mathbf{x}}_j, \mathbf{x}_{j+1}^\ell, ..., \mathbf{x}_p^\ell)$, $j = 1, ..., p$, as inexact solution of $\min_{\mathbf{x}_j} f_j(\mathbf{x}_j) \equiv F_\epsilon(\mathbf{x}_1^\ell, ..., \mathbf{x}_{j-1}^\ell, \mathbf{x}_j, \mathbf{x}_{j+1}^\ell, ..., \mathbf{x}_p^\ell)$ (Step 1). Then, the new iterate $\mathbf{x}^{\ell+1}$ is defined by a *connection rule*, ensuring that F_ϵ does not increase (Step 2). In particular, we set $\overline{\mathbf{y}} = (\overline{\mathbf{x}}_1, \overline{\mathbf{x}}_2, ..., \overline{\mathbf{x}}_p)$; then, if $F_\epsilon(\overline{\mathbf{y}}) \leq \min_{j=1,...p} F_\epsilon(\mathbf{y}^j)$ for all $j = 1, ..., p$, we set $\mathbf{x}^{\ell+1} = \overline{\mathbf{y}}$; otherwise $\mathbf{x}^{\ell+1} = \text{argmin}\{F_\epsilon(\mathbf{y}^1), ..., F_\epsilon(\mathbf{y}^p)\}$.
The vectors $\mathbf{y}^j$, $j = 1, ..., p$, are obtained as follows: BCDA method is applied to F_ϵ restricted to the variables $\mathbf{w}_j$ related to S_j, assuming null boundary conditions; then, at each iteration, the decrease condition for f_j is verified, extracting from the current iterate $\mathbf{w}_j^{k+1}$ only the entries $\mathbf{x}_j^{k+1}$ related to the tile T_j (Step 1.2.(b)). If the condition is satisfied, the update of the iterate is performed and a new iteration is started. Otherwise, we put $\overline{\mathbf{x}}_j = \mathbf{x}_j^k$ and the inner solver is stopped. As consequence, at any outer ℓ–iteration of OPARBCDA we have that $F_\epsilon(\mathbf{x}^{\ell+1}) \leq F_\epsilon(\mathbf{y}^j) \leq F_\epsilon(\mathbf{x}^\ell)$, $j = 1, ..., p$. Furthermore, since BCDA assures that the decrease of f_j, $j = 1..., p$, is bounded below by a forcing function on $\|\nabla_{x_j} F_\epsilon(\mathbf{x}^\ell)\|^2$, then any limit point of $\{\mathbf{x}^\ell\}$ is a stationary point of F_ϵ and, from the compactness of $\mathcal{L}_0$, there exists at least a limit point and we have $\nabla F_\epsilon(\mathbf{x}^\ell) \to 0$ as $\ell \to \infty$ ([7, Th. 3.2], [6, Th. 3.5]).
Two distinct approaches are used to parallelize Steps 1 and 2 of OPARBCDA. At each outer iteration, Step 1 consists of a number of independent tasks that can be concurrently solved. However, due to iterative nature of inner BCDA solver (Step 1.2), different running times are required for the solution of separate subproblems: manager/workers pattern [8], 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

Given $\mathbf{x}^0$, *the partitions* $\{T_1, ..., T_p\}$ *and* $\{S_1, ..., S_p\}$ *of* Λ *such that* $S_j \supset T_j$, $j = 1, ..., p$; *given tolerances* $\theta_{outer} \geq \theta_{inner} > 0$; *set* $\ell = 0$;

repeat

Step 1: **for** $j = 1, ..., p$

if $\nabla_{x_j} F_\epsilon(\mathbf{x}^\ell) \neq 0$ **then**

compute $\mathbf{y}^j = (\mathbf{x}_1^\ell, ..., \mathbf{x}_{j-1}^\ell, \overline{\mathbf{x}}_j, \mathbf{x}_{j+1}^\ell, ..., \mathbf{x}_p^\ell)$

1. *set* $\mathbf{w}_j^0 = (\mathbf{x}^\ell)|_{S_j}$, $k = -1$
2. **repeat**
   (a) $k = k + 1$; *compute* $\mathbf{w}_j^{k+1}$ *by a step of BCDA; extract* $\mathbf{x}_j^{k+1}$; *set* $\overline{\mathbf{x}}_j = \mathbf{x}_j^{k+1}$;
   (b) **if** $f_j(\mathbf{x}_j^{k+1}) > f_j(\mathbf{x}_j^k)$ **then** $\overline{\mathbf{x}}_j = \mathbf{x}_j^k$ **exit** *next* j; **end**
3. **until** $|f_j(\mathbf{x}_j^{k+1}) - f_j(\mathbf{x}_j^k)| > \theta_{inner}|f_j(\mathbf{x}_j^{k+1})|$

else

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

end

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

1. *set* $\overline{\mathbf{y}} = (\overline{\mathbf{x}}_1, \overline{\mathbf{x}}_2, ..., \overline{\mathbf{x}}_p)$; *compute* $F_\epsilon(\overline{\mathbf{y}})$;
2. *update* $\mathbf{x}^{\ell+1} = \text{argmin}\{F_\epsilon(\overline{\mathbf{y}}), F_\epsilon(\mathbf{y}^1), ..., F_\epsilon(\mathbf{y}^p)\}$.

Step 3: $\ell = \ell + 1$;

until $|F_\epsilon(\mathbf{x}^{\ell+1}) - F_\epsilon(\mathbf{x}^\ell)| \geq \theta_{outer}|F_\epsilon(\mathbf{x}^{\ell+1})|$

FIGURE 1. OPARBCDA scheme

directive *omp parallel for* is used for evaluation of $F_\epsilon(\overline{\mathbf{y}})$. While this step is performed, worker threads are waiting on a POSIX condition variable; thus they are not requiring cpu time.

Numerical Results

In order to evaluate the effectiveness of OPARBCDA, we consider a Trento DSM with size 2020×2020, spatial resolution of 1 mt and range of data $\mathbf{g}$ between 183.770 and 971.916 (`www.territorio.provincia.tn.it/portal/server.pt/community/lidar/847/lidar/23954`). We compare 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. 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. Since available core number is 16, 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. In Table 1 we show the percentage of entries of difference maps $|\mathbf{u}^s - \mathbf{u}^t|$, $|\mathbf{s}^s - \mathbf{s}^t|$, $|\mathbf{z}^s - \mathbf{z}^t|$ that are less than $\sigma_1 = 0.03$ and $\sigma_2 = 0.01$, for two partitions and two different size in pixels of overlapping frame ν along with outer number ℓ of required iterations. In Table 2 we show the comparison between BCDA and OPARBCDA with respect to a ground truth $\mathbf{x}^* = (\mathbf{u}^*, \mathbf{s}^*, \mathbf{z}^*)$, obtained by performing 1000 iterations of BCDA on the whole image with a tolerance 10e-10 for inner PCG; here, $F_\epsilon(\mathbf{x}^*)$ =8.09e+07. For each algorithm, we report mean and standard deviation of difference maps between ground truth and achieved solutions, as long as the percentage of entries of $|\mathbf{u}^s - \mathbf{u}^t|$, $|\mathbf{s}^s - \mathbf{s}^t|$, $|\mathbf{z}^s - \mathbf{z}^t|$ that are less than $\sigma_1 = 0.03$ and $\sigma_2 = 0.01$. We recall that $\mathbf{s}$ and $\mathbf{z}$ assume values in $[0, 1]$. For visualization purposes, only the central 1010×1010 portion of the image is considered. Parallel approach tile cuts are located at rows/columns [507 760 1013 1265]; Fig. 2 (a) shows the entries of $|\mathbf{z}^s - \mathbf{z}^*| > 0.01$, while Fig. 2 (b) shows the entries of $|\mathbf{z}^t - \mathbf{z}^*| > 0.01$ with t $= 8 \times 8$, $\nu = 10$.
Computational times in seconds (s.), averaged over 10 runs, are plotted in Fig. 2 (c). BCDA approach on whole image took 143.5 s., while, for $C = 15$ and $\nu = 4$, parallel OPARBCDA took 21.7 s. with t=8×8 and 18.2 s. with t=16×16.

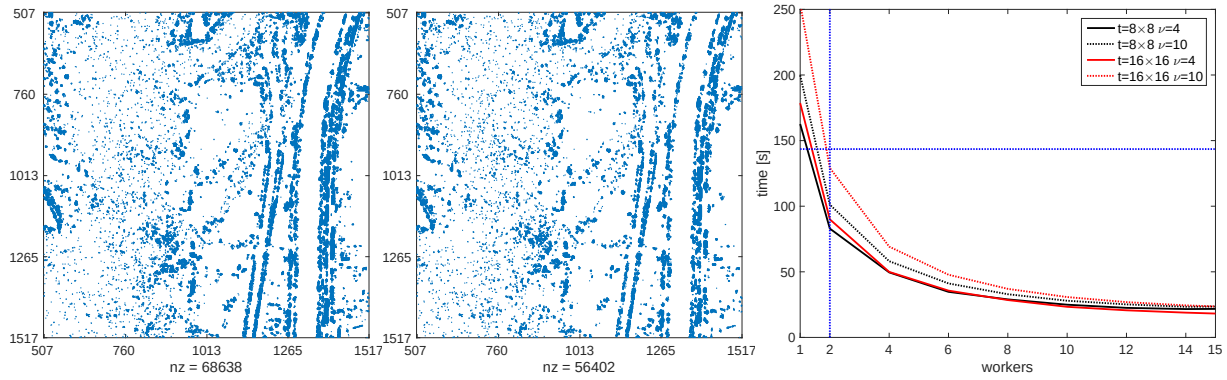
TABLE 1. Error analysis for OPARBCDA with respect to BCDA for a similar minimum approximation.

	ℓ	$\lvert\mathbf{u}^s - \mathbf{u}^t\rvert < \sigma_1$	$\lvert\mathbf{s}^s - \mathbf{s}^t\rvert < \sigma_1$	$\lvert\mathbf{z}^s - \mathbf{z}^t\rvert < \sigma_1$	$\lvert\mathbf{u}^s - \mathbf{u}^t\rvert < \sigma_2$	$\lvert\mathbf{s}^s - \mathbf{s}^t\rvert < \sigma_2$	$\lvert\mathbf{z}^s - \mathbf{z}^t\rvert < \sigma_2$
t=8×8 ν=4	2	97.97%	99.27%	98.05%	93.89%	97.71%	96.01%
t=8×8 ν=10	2	98.17%	99.30%	98.14%	94.62%	97.84%	96.24%
t=16×16 ν=4	3	97.81%	99.25%	97.95%	93.20%	97.62%	95.73%
t=16×16 ν=10	3	97.90%	99.20%	97.97%	94.06%	97.58%	95.91%

TABLE 2. Error analysis for both BCDA and OPARBCDA with respect to a ground truth $\mathbf{x}^*$. rel.err denotes the relative error of $F_\epsilon(\mathbf{x}^s)$ and $F_\epsilon(\mathbf{x}^t)$ with respect to $F_\epsilon(\mathbf{x}^*)$.

BCDA	mean	std. deviation	$< \sigma_1$	$< \sigma_2$				
$\mathbf{u}^s - \mathbf{u}^*$	4.66e-05	3.03e-02	94.40%	88.05%	$F_\epsilon(\mathbf{x}^s)$=8.16e+07 rel.err.=8.18e-03			
$\mathbf{s}^s - \mathbf{s}^*$	-3.85e-04	1.57e-02	97.79%	94.77%				
$\mathbf{z}^s - \mathbf{z}^*$	-8.18e-03	9.37e-02	96.22%	93.21%				
OPARBCDA	t=8×8 ν=4 $F_\epsilon(\mathbf{x}^t)$=8.14e+07 rel.err.=5.82e-03				t=8×8 ν=10 $F_\epsilon(\mathbf{x}^t)$=8.14e+07 rel.err.=5.77e-03			
	mean	std. deviation	$< \sigma_1$	$< \sigma_2$	mean	std. deviation	$< \sigma_1$	$< \sigma_2$
$\mathbf{u}^t - \mathbf{u}^*$	3.25e-06	2.68e-02	95.48%	89.87%	-3.13e-06	2.66e-02	95.62%	90.46%
$\mathbf{s}^t - \mathbf{s}^*$	-2.74e-04	1.37e-02	98.31%	95.92%	-2.82e-04	1.36e-02	98.34%	95.99%
$\mathbf{z}^t - \mathbf{z}^*$	-5.80e-03	8.19e-02	97.04%	94.51%	-5.79e-03	8.14e-02	97.11%	94.68%
OPARBCDA	t=16×16 ν=4 $F_\epsilon(\mathbf{x}^t)$=8.14e+07 rel.err.=5.94e-03				t=16×16 ν=10 $F_\epsilon(\mathbf{x}^t)$=8.13e+07 rel.err.=5.24e-03			
	mean	std. deviation	$< \sigma_1$	$< \sigma_2$	mean	std. deviation	$< \sigma_1$	$< \sigma_2$
$\mathbf{u}^t - \mathbf{u}^*$	3.14e-05	2.75e-02	95.16%	88.82%	-7.13e-06	2.59e-02	95.82%	90.81%
$\mathbf{s}^t - \mathbf{s}^*$	-2.76e-04	1.39e-02	98.25%	95.74%	-2.52e-04	1.33e-02	98.43%	96.20%
$\mathbf{z}^t - \mathbf{z}^*$	-6.13e-03	8.31e-02	96.78%	93.97%	-5.36e-03	7.90e-02	97.23%	94.84%

Increasing the overlap size to $\nu = 10$ led to a modest variation on computational time (23.5 and 23.6 s. respectively), and it is sufficient for achieving an accuracy similar to serial solution. The described results appear promising in order to provide a fast and accurate tool for the segmentation of huge images.



(a) (b) (c)

FIGURE 2. (a): Entries of $|\mathbf{z}^s - \mathbf{z}^*| > 0.01$. (b): Entries of $|\mathbf{z}^t - \mathbf{z}^*| > 0.01$ with t =8×8, $\nu = 10$. (c): Computational time for parallel OPARBCDA with respect to the number C of workers. Horizontal line at 143.5 s. is serial BCDA time.

ACKNOWLEDGMENTS

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

REFERENCES

[1] M. Zanetti, V. Ruggiero, and M. J. Miranda, Commun. Nonlinear Sci. Numer. Simul. **36**, 528–548 (2016).
[2] G. Forlani, C. Nardinocchi, M. Scaioni, and P. Zingaretti, Pattern Anal. Appl. **8**, 357–374 (2006).
[3] D. Mumford and J. Shah, Comm. Pure Appl. Math. **42**, 577–685 (1989).
[4] A. Blake and A. Zisserman, (MIT Press, Cambridge, MA, 1987), pp. xii+225.
[5] L. Ambrosio, L. Faina, and R. March, SIAM J. Math. Anal. **32**, 1171–1197 (2001).
[6] L. Grippo and M. Sciandrone, Optim. Methods Softw. **10**, 587–637 (1999).
[7] O. L. Mangasarian, SIAM J. Optim. **33**, 1916–1925 (1995).
[8] J. Ortega-Arjona and G. Roberts, "Architectural patterns for parallel programming," (Kloster Irsee, Germany, 1988).