Serial and parallel approaches for image segmentation by numerical minimization of a second-order functional

R. Zanella[a], F. Porta[a], V. Ruggiero[a,*], M. Zanetti[b]

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

Abstract

Because of its attractive features, second order segmentation has shown to be a promising tool in remote sensing. A known drawback about its implementation is computational complexity, above all for large set of data. Recently [1], an efficient version of the block-coordinate descent algorithm (BCDA) has been proposed for the minimization of a second order elliptic approximation of the Blake–Zissermann functional. Although the parallelization of linear algebra operations is expected to increase the performance of BCDA when addressing the segmentation of large-size gridded data (e.g., full-scene images or Digital Surface Models (DSMs)), numerical evidence shows that this is not sufficient to get significant reduction of computational time. Therefore a novel approach is proposed which exploits a decomposition technique of the image domain into tiles. The solution can be computed by applying BCDA on each tile in parallel way and combining the partial results corresponding to the different blocks of variables through a proper interconnection rule. We prove that this parallel method (OPARBCDA) generates a sequence of iterates which converges to a critical point of the functional on the level set devised by the starting point. Furthermore, we show that the parallel method can be efficiently implemented even in a commodity multicore CPU. Numerical results are provided to evaluate the efficiency of the parallel scheme on large images in terms of computational cost and its effectiveness with respect to the behavior on the tile junctions.

Keywords: Segmentation; Blake Zisserman functional; domain decomposition; block coordinate descent methods; parallel interconnection rule

1. Introduction

Despite the wide image processing domain provides many different techniques to extract information from images, a major limitation which is often encountered is the capability to work in real application scenarios, e.g., in remote

*Corresponding author
Email address: valeria.ruggiero@unife.it (V. Ruggiero)

sensing (RS). As an example, digital surface models (DSMs) of the Earth's surface can be obtained from LiDAR (Light Ranging and Detection) point clouds. From this huge amount of data, useful information needs to be extracted systematically and, very often, this task must be accomplished on commodity machines with limited resources. Effective object-based or simplified representations of images can be obtained via segmentation approaches, where objects are identified by segments. To process large images, a common procedure consists in splitting the input into several tiles, running a specific segmentation algorithm separately on each tile and, then, merging together the partial results. However, this strategy is empirical in nature and the global dependency of the solution on data often introduces undesired artifacts which potentially propagate from the tiles junctions to their interior. It is a matter of fact that, the theoretical justification of tiling approaches for segmentation algorithms is rarely considered in literature.

Among the mathematical methods to address the image segmentation problem, a key role is played by variational models. In this case, the solution is theoretically formulated as a minimizer of a global energy. In the seminal paper [2], Mumford ad Shah (MS) proposed a first-order functional, whose minimization determines an approximation of the image by means of a piecewise smooth function. The Blake-Zisserman (BZ) second-order model [3] has been introduced with the aim of overcoming the limitations of the Mumford-Shah approach, such as the over-segmentation of the step gradients and the lack in 2nd-order detection (gradient discontinuities) and the triple-point problem (see [4, 5] for a theoretical study). Nevertheless, the original functional formulation of the segmentation problem by MS or BZ is too strong for a numerical treatment. Among many approaches employed to numerically compute minimizers, we recall the well-known Ambrosio-Tortorelli (AT) approximation of the MS functional [6], which is prone to be numerically implemented. In their functional model, Ambrosio-Tortorelli replaced the unknown discontinuity set by an auxiliary function which smoothly approximates its indicator function. The initial choice of the discontinuity function can be made in an energetically convenient manner by exploiting theoretical known properties of the solution. Moreover, the AT approximation of the MS functional enjoys partial (quadratic) convexity property; therefore gradient-based methods have in general satisfying performance. It is worth mentioning here that a multi-grid approach to speed-up computations is exploited in [7]. The numerical treatment of the BZ functional is considered by Bellettini and Coscia [8, 9] in the one dimensional case, and by Ambrosio, Faina, March in the two dimensional case [10]. In the latter, the main result is an elliptic approximation of the functional where the technique proposed by AT (to approximate the MS functional) has been properly adapted for the 2nd-order functional. Here, two auxiliary functions are introduced as indicators of both the discontinuity and gradient discontinuity sets of the solution. More specifically, let $\Omega \in \mathbb{R}^2$ and $g \in L^\infty(\Omega)$ a given image. The goal is

to minimize the functional

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

in proper Sobolev spaces. Here $\delta, \alpha, \beta, \mu$ are positive parameters ($2\beta \geq \alpha \geq \beta$) and the terms ξ_ϵ, o_ϵ are infinitesimals. The approximation of the BZ functional takes place when $\epsilon \to 0$. Notice that, in the limit case, the minimizing functions s and z must be 1 almost everywhere on Ω in order to keep the energy finite. From the numerical point of view, by fixing ϵ as a small value, we have that functions s, z are allowed to have variations from 1 to 0 in a small neighborhood of the jump and the crease set of u, as this inhibits the costly contribution of $|\nabla u|^2$ and $|\nabla^2 u|^2$, respectively. Recently, the numerical minimization of the nonconvex functional (1) has been obtained by an especially tailored version of a block-coordinate descent algorithm (BCDA) [11], based on a compact matrix formulation of the functional [1]. Altough theoretical models such as MS and BZ are global, the non-convexity of the objective functionals forces numerical methods to provide sub-optimal solutions. The outcome of many numerical experiments has highlighted that, although the theoretical model is global, the solutions weakly depend on boundary conditions and they are energetically close to initial data. This fact motivated us in developing a tiling scheme to address the segmentation of large images where a minimizer of the functional is assembled by merging together local minimizers restricted to sub-portions of the image. Preliminary results were encouraging [12, 13].

The aim of this work is to provide mathematical arguments which justify the heuristic of this procedure. We start from the consideration that a simple idea to deal with very large images, might be to implement in a parallel way the linear algebra operations of BCDA. Nevertheless numerical evidence highlights that the gained performance is limited. Thus, inspired by recent papers on the convergence of descent scheme for semi-algebraic problems [14, 15, 16], we propose a parallel approach, based on the decomposition of the image into partially overlapping tiles, which enables us to obtain a satisfactory accuracy also on the strip regions around the tile boundaries. This parallel iterative scheme, hereafter denoted by OPARBCDA, is basically a descent method which independently computes a portion of the new iterate for any tile by BCDA. Convergence results for the sequence of iterates generated by OPARBCDA are obtained. Particularly, we prove that the whole sequence of the OPARBCDA iterates converges to a critical point of the objective function. A suitable parallel implementation of OPARDCDA, based on a run-time distribution of independent tasks on the available cores of a multicore commodity, enables to address the segmentation of large images. Further, few iterations of the scheme determines an approximate solution that shows similar accuracy in every subregions of the domain.

The paper is organized as follows: in Section 2 we describe the discretization of (1) and its features by introducing the notation. In Section 3 we recall the special version of the block-coordinate descent algorithm BCDA to address the numerical minimization of the discrete functional (1) and we give the convergence analysis of the scheme, by exploiting the Kurdyka–Lojasiewicz (KL) property of the polynomial discrete functional. In Section 4, we introduce a parallel method, named OPARBCDA and based on the decomposition of the image domain into overlapping tiles, we address its convergence properties; moreover, always in this Section, after a description of the tools to parallelize the linear algebra operations of BCDA, we discuss how to exploit the intrinsic parallel feature of OPARBCDA in a multicore CPU. Finally, in Section 5, the results of a vast numerical experimentation enable to compare and evaluate the effectiveness on the proposed approaches.

2. Discretization of the elliptic approximation of the Blake-Zisserman functional

The numerical approach for the minimization of $\mathcal{F}_\epsilon$ exploits a discrete version of the functional. The rectangular domain $\Omega \subset \mathbb{R}^2$ is discretized by a lattice of points $\Lambda = \{(it_x, jt_y);\ \ i = 1, \ldots, N,\ j = 1, \ldots, M\}$ with step sizes t_x and t_y on the x and y directions respectively. In order to take into account boundary conditions, this lattice can be viewed as a subset of an enlarged lattice $\overline{\Lambda}$. We denote the set of points on the frame outside of Ω by $\mathcal{B} \equiv \overline{\Lambda} - \Lambda$.
By using the standard representation of grey-scale images as matrices, the values g_{ij} of the given image g are defined only on the grid points (it_x, jt_y) of Λ. The approximate values of the functions s, z, u at the grid points of $\overline{\Lambda}$ are denoted by s_{ij}, z_{ij}, u_{ij}, assuming as in [10] zero boundary conditions on $\mathcal{B}$ (i. e. $s_{i,j} = z_{i,j} = u_{i,j} = 0,\ (i,j) \in \mathcal{B}$). By using a column-wise ordering for the elements of these matrices, the image matrix is denoted also by the NM vector $\mathbf{g}$ while the vectors $\mathbf{s}_{\overline{\Lambda}}, \mathbf{z}_{\overline{\Lambda}}, \mathbf{u}_{\overline{\Lambda}}$ denote the entries of the functions s, z, u at the grid points of $\overline{\Lambda}$; the discrete functional depends only on the sub–vectors, denoted in the following with $\mathbf{s}, \mathbf{z}, \mathbf{u}$, whose NM entries are the approximate values of s, z, u at the points of Λ; the others entries of $\mathbf{s}_{\overline{\Lambda}}, \mathbf{z}_{\overline{\Lambda}}, \mathbf{u}_{\overline{\Lambda}}$ are their boundary values which interleave the elements of $\mathbf{s}, \mathbf{z}, \mathbf{u}$ in $\mathbf{s}_{\overline{\Lambda}}, \mathbf{z}_{\overline{\Lambda}}, \mathbf{u}_{\overline{\Lambda}}$ and form three other sub–vectors denoted with $\mathbf{s}_{\mathcal{B}}, \mathbf{z}_{\mathcal{B}}, \mathbf{u}_{\mathcal{B}}$.
By exploiting first and second–order difference schemes to approximate differential operators, and taking account of all contributions affecting $\mathbf{s}, \mathbf{z}, \mathbf{u}$ from the boundaries, by a simple 2–D composite rectangular rule, we obtain the following

discrete form of the functional (1):

$$
\begin{aligned}
&F_\epsilon(\mathbf{s},\mathbf{z},\mathbf{u}) := \\
&\quad t_x t_y \left\{ \delta \sum_{i=0}^{N+1}\sum_{j=0}^{M+1} z_{i,j}^2 \left(\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{t_x^2} \right)^2 + \left(\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{t_y^2} \right)^2 \right) + \right. \\
&\quad +2\delta \sum_{i=0}^{N}\sum_{j=0}^{M} z_{i,j}^2 \left(\frac{u_{i+1,j+1} - u_{i,j+1}}{t_y t_x} - \frac{u_{i+1,j} - u_{i,j}}{t_y t_x} \right)^2 + \\
&\quad +\xi_\epsilon \sum_{i=0}^{N}\sum_{j=0}^{M} (s_{i,j}^2 + o_\epsilon) \left(\left(\frac{u_{i+1,j} - u_{i,j}}{t_x} \right)^2 + \left(\frac{u_{i,j+1} - u_{i,j}}{t_y} \right)^2 \right) + \\
&\quad +(\alpha - \beta) \left[\epsilon \sum_{i=0}^{N}\sum_{j=0}^{M} \left(\left(\frac{s_{i+1,j} - s_{i,j}}{t_x} \right)^2 + \left(\frac{s_{i,j+1} - s_{i,j}}{t_y} \right)^2 \right) + \frac{1}{4\epsilon} \sum_{i=1}^{N}\sum_{j=1}^{M} (s_{i,j} - 1)^2 \right] + \\
&\quad +\beta \left[\epsilon \sum_{i=0}^{N}\sum_{j=0}^{M} \left(\left(\frac{z_{i+1,j} - z_{i,j}}{t_x} \right)^2 + \left(\frac{z_{i,j+1} - z_{i,j}}{t_y} \right)^2 \right) + \frac{1}{4\epsilon} \sum_{i=1}^{N}\sum_{j=1}^{M} (z_{i,j} - 1)^2 \right] + \\
&\quad \left. +\mu \sum_{i=1}^{N}\sum_{j=1}^{M} (u_{i,j} - g_{i,j})^2 \right\}
\end{aligned}
\tag{2}
$$

with $2\beta \geq \alpha \geq \beta > 0$ and $\delta, \mu, \epsilon > 0$, $\xi_\epsilon, o_\epsilon, \geq 0$. In order to write in compact matrix form the functional $F_\epsilon(\mathbf{s},\mathbf{z},\mathbf{u})$, we introduce the square matrices $\mathbf{A}_K^1, \mathbf{A}_K^2$ of order $K+4$, representing first and second order discrete operators respectively:

$$
\mathbf{A}_K^1 := \begin{pmatrix}
0 & 0 & & & & & \\
 & -1 & 1 & & & & \\
 & & \ddots & \ddots & & & \\
 & & & -1 & 1 & & \\
 & & & & 0 & 0 & \\
 & & & & & 0 &
\end{pmatrix}, \quad
\mathbf{A}_K^2 := \begin{pmatrix}
0 & 0 & & & & & \\
1 & -2 & 1 & & & & \\
 & \ddots & \ddots & \ddots & & & \\
 & & 1 & -2 & 1 & & \\
 & & & 1 & -2 & 1 & \\
 & & & & 0 & 0 &
\end{pmatrix}.
$$

By denoting by $\mathbf{I}_K$ and $\mathbf{0}_K$ the identity and the null matrices of order K respectively and by $\otimes$ the Kronecker product, we can write the first and the second order discrete operators in the following way:

$$
\begin{aligned}
&\mathbf{D}_x := \frac{1}{t_x} \mathbf{W}_M \otimes \mathbf{A}_N^1; \qquad \mathbf{D}_y := \frac{1}{t_y} \mathbf{A}_M^1 \otimes \mathbf{W}_N, \\
&\mathbf{D}_{xx} := \frac{1}{t_x^2} \mathbf{W}_M \otimes \mathbf{A}_N^2; \qquad \mathbf{D}_{yy} := \frac{1}{t_y^2} \mathbf{A}_M^2 \otimes \mathbf{W}_N; \\
&\mathbf{D}_{xy} := \frac{1}{t_x t_y} (\mathbf{A}_M^1 \otimes \mathbf{J}_N)(\mathbf{J}_M \otimes \mathbf{A}_N^1) = \frac{1}{t_x t_y} (\mathbf{J}_M \otimes \mathbf{A}_N^1)(\mathbf{A}_M^1 \otimes \mathbf{J}_N)
\end{aligned}
$$

with $\mathbf{W}_M = \mathrm{diag}(\mathbf{0}_2, \mathbf{I}_M, \mathbf{0}_2)$, $\mathbf{W}_N = \mathrm{diag}(\mathbf{0}_2, \mathbf{I}_N, \mathbf{0}_2)$, $\mathbf{J}_M = \mathrm{diag}(\mathbf{0}_1, \mathbf{I}_{M+1}, \mathbf{0}_2)$ and $\mathbf{J}_N = \mathrm{diag}(\mathbf{0}_1, \mathbf{I}_{N+1}, \mathbf{0}_2)$.
In the following, for any generic $(N+4)(M+4)$ matrix operator $\mathbf{D}$, let us denote by $\mathbf{D}(:,\Lambda)$ and $\mathbf{D}(:,\mathcal{B})$ the sub–matrices of $\mathbf{D}$ given by the columns with indices corresponding to points of Λ and $\mathcal{B}$ respectively, ordered in a column–wise way. Furthermore, given a generic vector $\mathbf{v}$, let us denote by $\mathbf{R}_\mathbf{v}$ the diagonal matrix with diagonal entries equal to the elements of $\mathbf{v}$, i.e. $\mathbf{R}_\mathbf{v} = \mathrm{diag}(\mathbf{v})$. We also denote by $\mathbf{v}^2$ the vector of the squared entries of $\mathbf{v}$, i.e., $(\mathbf{v}^2)_i = (\mathbf{v}_i)^2$ and $\mathbf{1} := (1, 1, \ldots, 1)^T$.

Using this notation, the discrete functional (2) can be written as follows:

$$\begin{aligned} F_\epsilon(\mathbf{s},\mathbf{z},\mathbf{u}) := t_x t_y \Big\{ & \delta\, \mathbf{u}_\Lambda^T (\mathbf{D}_{xx}^T \mathbf{R}_{\mathbf{z}^2_{\overline{\Lambda}}} \mathbf{D}_{xx} + \mathbf{D}_{yy}^T \mathbf{R}_{\mathbf{z}^2_{\overline{\Lambda}}} \mathbf{D}_{yy} + 2\mathbf{D}_{xy}^T \mathbf{R}_{\mathbf{z}^2_{\overline{\Lambda}}} \mathbf{D}_{xy}) \mathbf{u}_{\overline{\Lambda}} + \\ & + \xi_\epsilon\, \mathbf{u}_\Lambda^T (\mathbf{D}_x^T \mathbf{R}_{\mathbf{s}^2_{\overline{\Lambda}}+o_\epsilon} \mathbf{D}_x + \mathbf{D}_y^T \mathbf{R}_{\mathbf{s}^2_{\overline{\Lambda}}+o_\epsilon} \mathbf{D}_y) \mathbf{u}_{\overline{\Lambda}} + \\ & + (\alpha-\beta)\, [\epsilon\, \mathbf{s}_\Lambda^T (\mathbf{D}_x^T \mathbf{D}_x + \mathbf{D}_y^T \mathbf{D}_y) \mathbf{s}_{\overline{\Lambda}} + \frac{1}{4\epsilon} (\mathbf{s}-\mathbf{1})^T (\mathbf{s}-\mathbf{1})] + \\ & + \beta\, [\epsilon\, \mathbf{z}_\Lambda^T (\mathbf{D}_x^T \mathbf{D}_x + \mathbf{D}_y^T \mathbf{D}_y) \mathbf{z}_{\overline{\Lambda}} + \frac{1}{4\epsilon} (\mathbf{z}-\mathbf{1})^T (\mathbf{z}-\mathbf{1})] + \\ & + \mu\, (\mathbf{u}-\mathbf{g})^T (\mathbf{u}-\mathbf{g}) \Big\}. \end{aligned} \tag{3}$$

Globally this functional is not convex, but it is quadratic with respect to each block of variables $\mathbf{s}, \mathbf{z}, \mathbf{u}$ when the others are fixed: the terms of F_ϵ containing $\mathbf{s}$ or $\mathbf{z}$ depend only on $\mathbf{u}$; on the other hand, the terms containing $\mathbf{u}$ depend on $\mathbf{s}$ and $\mathbf{z}$. Indeed, by fixing the variable $\mathbf{u}$ or the other two variables $\mathbf{s}$ and $\mathbf{z}$, we can write

$$\begin{aligned} F_\epsilon(\mathbf{s},\mathbf{z},\mathbf{u}) &= t_x t_y \left\{ \frac{1}{2} \begin{pmatrix} \mathbf{s}^T & \mathbf{z}^T \end{pmatrix} \begin{pmatrix} \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 \end{pmatrix} \begin{pmatrix} \mathbf{s} \\ \mathbf{z} \end{pmatrix} - \begin{pmatrix} \mathbf{s}^T & \mathbf{z}^T \end{pmatrix} \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} + \mathbf{c}_{sz} \right\}, \\ F_\epsilon(\mathbf{s},\mathbf{z},\mathbf{u}) &= t_x t_y \left\{ \frac{1}{2} \mathbf{u}^T \mathbf{A}_3\, \mathbf{u} - \mathbf{u}^T \mathbf{b}_3 + \mathbf{c}_u \right\}, \end{aligned} \tag{4}$$

where $\mathbf{A}_1 = \mathbf{A}_1(\mathbf{u})$, $\mathbf{A}_2 = \mathbf{A}_2(\mathbf{u})$, $\mathbf{A}_3 = \mathbf{A}_3(\mathbf{s},\mathbf{z})$ and $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ are given by

$$\begin{aligned} \mathbf{A}_1 =& 2\xi_\epsilon \mathbf{R}_{|\nabla \mathbf{u}|^2} + 2\epsilon(\alpha-\beta)(\mathbf{D}_x(:,\Lambda)^T \mathbf{D}_x(:,\Lambda) + \mathbf{D}_y(:,\Lambda)^T \mathbf{D}_y(:,\Lambda)) + \\ & + \frac{\alpha-\beta}{2\epsilon} \mathbf{I}_{NM}, \\ \mathbf{b}_1 =& -2\epsilon(\alpha-\beta)(\mathbf{D}_x(:,\Lambda)^T \mathbf{D}_x(:,\mathcal{B}) + \mathbf{D}_y(:,\Lambda)^T \mathbf{D}_y(:,\mathcal{B})) \mathbf{s}_\mathcal{B} + \frac{\alpha-\beta}{2\epsilon} \mathbf{1}, \\ \mathbf{A}_2 =& 2\delta \mathbf{R}_{|\nabla^2 \mathbf{u}|^2} + 2\epsilon\beta(\mathbf{D}_x(:,\Lambda)^T \mathbf{D}_x(:,\Lambda) + \mathbf{D}_y(:,\Lambda)^T \mathbf{D}_y(:,\Lambda)) + \frac{\beta}{2\epsilon} \mathbf{I}_{NM}, \\ \mathbf{b}_2 =& -2\epsilon\beta(\mathbf{D}_x(:,\Lambda)^T \mathbf{D}_x(:,\mathcal{B}) + \mathbf{D}_y(:,\Lambda)^T \mathbf{D}_y(:,\mathcal{B})) \mathbf{z}_\mathcal{B} + \frac{\beta}{2\epsilon} \mathbf{1}, \\ \mathbf{A}_3 =& 2\delta(\mathbf{D}_{xx}(:,\Lambda)^T \mathbf{R}_{\mathbf{z}^2_{\overline{\Lambda}}} \mathbf{D}_{xx}(:,\Lambda) + \mathbf{D}_{yy}(:,\Lambda)^T \mathbf{R}_{\mathbf{z}^2_{\overline{\Lambda}}} \mathbf{D}_{yy}(:,\Lambda) + \\ & + 2\mathbf{D}_{xy}(:,\Lambda)^T \mathbf{R}_{\mathbf{z}^2_{\overline{\Lambda}}} \mathbf{D}_{xy}(:,\Lambda)) + 2\xi_\epsilon(\mathbf{D}_x(:,\Lambda)^T \mathbf{R}_{\mathbf{s}^2_{\overline{\Lambda}}+o_\epsilon} \mathbf{D}_x(:,\Lambda) + \\ & + \mathbf{D}_y(:,\Lambda)^T \mathbf{R}_{\mathbf{s}^2_{\overline{\Lambda}}+o_\epsilon} \mathbf{D}_y(:,\Lambda)) + 2\mu \mathbf{I}_{NM}, \\ \mathbf{b}_3 =& -2\delta(\mathbf{D}_{xx}(:,\Lambda)^T \mathbf{R}_{\mathbf{z}^2_{\overline{\Lambda}}} \mathbf{D}_{xx}(:,\mathcal{B}) + \mathbf{D}_{yy}(:,\Lambda)^T \mathbf{R}_{\mathbf{z}^2_{\overline{\Lambda}}} \mathbf{D}_{yy}(:,\mathcal{B}) + \\ & + 2\mathbf{D}_{xy}(:,\Lambda)^T \mathbf{R}_{\mathbf{z}^2_{\overline{\Lambda}}} \mathbf{D}_{xy}(:,\mathcal{B})) + 2\xi_\epsilon(\mathbf{D}_x(:,\Lambda)^T \mathbf{R}_{\mathbf{s}^2_{\overline{\Lambda}}+o_\epsilon} \mathbf{D}_x(:,\mathcal{B}) + \\ & + \mathbf{D}_y(:,\Lambda)^T \mathbf{R}_{\mathbf{s}^2_{\overline{\Lambda}}+o_\epsilon} \mathbf{D}_y(:,\mathcal{B})) \mathbf{u}_\mathcal{B} + 2\mu \mathbf{g}. \end{aligned} \tag{5}$$

Vectors $\mathbf{c}_{sz}$ and $\mathbf{c}_u$ are constant, thus irrelevant for the minimization.

Remark 1. *We observe that when $\alpha = \beta$ and $\xi_\epsilon = 0$, the functional does not depend anymore on the block variable* $\mathbf{s}$. *The features and the numerical treatment of this reduced version of F_ϵ are similar to the ones of the general case, with the only difference that F_ϵ is quadratic with respect to the block variable* $\mathbf{z}$ *when* $\mathbf{u}$ *is fixed, since $\mathbf{A}_1 = 0$, $\mathbf{b}_1 = 0$. Thus, since the setting $\alpha = \beta$ and $\xi_\epsilon = 0$ is a special case, in the following we address the general formulation, assuming $\alpha > \beta$ and $\xi_\epsilon > 0$.*

The functional F_ϵ has the following properties:

P1. $F_\epsilon(\mathbf{s}, \mathbf{z}, \mathbf{u})$ is continuously differentiable;

P2. the matrices $\mathbf{A}_i$, $i = 1, 2, 3$ are very sparse and structured: $\mathbf{A}_1$ and $\mathbf{A}_2$ are block tridiagonal matrices, where the diagonal blocks are tridiagonal and the off-diagonal blocks are diagonal; $\mathbf{A}_3$ is a block five matrix, with at most 13 nonzero entries for each row;

P3. in view of the terms $\frac{\alpha-\beta}{2\epsilon}\mathbf{I}$, $\frac{\beta}{2\epsilon}\mathbf{I}$ and $2\mu\mathbf{I}$, the matrices $\mathbf{A}_i$, $i = 1, 2, 3$, are symmetric and positive definite and their minimum eigenvalues $\lambda_{min}(\mathbf{A}_i)$ are below bounded by $\frac{\alpha-\beta}{2\epsilon}$, $\frac{\beta}{2\epsilon}$ and 2μ respectively;

P4. in view of the positive definiteness of matrices $\mathbf{A}_i$, $i = 1, 2, 3$, $F_\epsilon(\mathbf{s}, \mathbf{z}, \mathbf{u})$ is quadratic and strongly convex with respect to each block component $\mathbf{s}, \mathbf{z}, \mathbf{u}$, when the others are left fixed;

P5. F_ϵ is coercive in $\mathbb{R}^{3NM}$ (see [1]); thus the level sets of F_ϵ are compact;

P6. on a given level set, the matrices $\mathbf{A}_i$, $i = 1, 2, 3$, have bounded positive eigenvalues;

P7. on a given level set, the gradient of F_ϵ is Lipschitz–continuous;

P8. the functional F_ϵ is a polynomial in $\mathbf{s}, \mathbf{z}, \mathbf{u}$ and, consequently, it is a semi-algebraic function, that satisfies the Kurdyka–Lojasiewicz (KL) property on its domain (see [15] and reference therein).

In the following, for notation convenience, a generic point $(\mathbf{s}, \mathbf{z}, \mathbf{u})$ in $\mathbb{R}^{3NM}$ is represented either by $\mathbf{y}$ or $\mathbf{x}$. When $\mathbf{y}$ is used, the variables are grouped in blocks according the simple correspondence: $\mathbf{y}_1 = \mathbf{s}$, $\mathbf{y}_2 = \mathbf{z}$ and $\mathbf{y}_3 = \mathbf{u}$; on the other hand, we refer to $\mathbf{x}$ when the block decomposition is based on the spatial subdivision of Λ. The functional restricted to a block variable $\mathbf{v}_i$ is denoted by $f_{\mathbf{v}_i}$, i. e. $f_{\mathbf{v}_i}(\mathbf{v}_i) = F_\epsilon(..., \mathbf{v}_i, ...)$; the gradient of F_ϵ at $\tilde{\mathbf{v}}$ with respect to the block $\mathbf{v}_i$ is denoted by $\nabla_{\mathbf{v}_i} F_\epsilon(\tilde{\mathbf{v}})$ and $\nabla_{\mathbf{v}_i} F_\epsilon(\tilde{\mathbf{v}}) = \nabla f_{\mathbf{v}_i}(\tilde{\mathbf{v}}_i)$.

3. Numerical minimization of F_ϵ: a sequential approach

In this section we recall the original BCDA scheme presented in [1]. In view of the remark that the discrete approximation of (1) is a polynomial function, satisfying the Kurdyka-Lojasiewicz property, we obtain the convergence of the

sequence generated by BCDA to a critical point.
BCDA is a version of block coordinate descent algorithm [11], especially tailored to exploit the features of the functional $F_\epsilon(\mathbf{s},\mathbf{z},\mathbf{u})$. Starting from an initial vector $\mathbf{y}^0=(\mathbf{s}^0,\mathbf{z}^0,\mathbf{u}^0)$, the basic idea of the method is to cyclically determine for each block variable $\mathbf{y}_i$ a descent direction $\mathbf{d}_i$ by few iterations of a preconditioned conjugate gradient (PCG) method applied to the linear system $\mathbf{A}_i^k\mathbf{d}_i=\mathbf{b}_i-\mathbf{A}_i^k\mathbf{y}_i^k=-\nabla_{\mathbf{y}_i}F_\epsilon(\mathbf{y}^k)$, with $\mathbf{A}_i^k\equiv\mathbf{A}_i^k(\mathbf{y}^k)$. In view of property **P4** of the objective function, the step-lengths along the computed descent directions $\mathbf{d}_i^k$ at the iteration k can be determined without having to use an Armijo–type procedure to ensure a sufficient decrease of the objective function. Indeed when the step size α_i^k is given by the following rule

$$\alpha_i^k:=\gamma_i\frac{(\mathbf{b}_i-\mathbf{A}_i^k\mathbf{y}_i^k)^T\mathbf{d}_i^k}{\mathbf{d}_i^{k\,T}\mathbf{A}_i^k\mathbf{d}_i^k}=\gamma_i\frac{-\nabla_{\mathbf{y}_i}F_\epsilon(\mathbf{y}^k)^T\mathbf{d}_i^k}{\mathbf{d}_i^{k\,T}A_i^k\mathbf{d}_i^k},\tag{6}$$

with $0<\gamma_i\le 2(1-\rho_i)$, $0<\rho_i<1$, a sufficient decrease for the objective function restricted to the block variable $\mathbf{y}_i$ is assured:

$$f_{\mathbf{y}_i}(\mathbf{y}_i^k+\alpha_i^k\mathbf{d}_i^k)\le f_{\mathbf{y}_i}(\mathbf{y}_i^k)+\rho_i\alpha_i^k\nabla f_{\mathbf{y}_i}(\mathbf{y}_i^k)^T\mathbf{d}_i^k.\tag{7}$$

In particular, for $\gamma_i=1$, we obtain the exact one-dimensional minimizer of the strongly convex quadratic function along the direction $\mathbf{d}_i^k$ and (7) holds for $\rho_i\le\frac{1}{2}$. The special version of the block-coordinate descent algorithm for $F_\epsilon(\mathbf{s},\mathbf{z},\mathbf{u})$, named BCDA, is outlined in Algorithm 1. In view of **P5**, the level set $\mathcal{L}_{F_\epsilon^0}=\{(\mathbf{s},\mathbf{z},\mathbf{u}):F_\epsilon(\mathbf{s},\mathbf{z},\mathbf{u})\le F_\epsilon^0\equiv F_\epsilon(\mathbf{s}^0,\mathbf{z}^0,\mathbf{u}^0)\}$ is a compact subset of $\mathbb{R}^{3MN}$. Thus, for $(\mathbf{s},\mathbf{z},\mathbf{u})\in\mathcal{L}_{F_\epsilon^0}$, the eigenvalues $\lambda_j(\mathbf{A}_i^k)$ of the matrices $\mathbf{A}_i^k$, $i=1,2,3$, $k\ge0$, are bounded by positive constants (**P6**):

$$0<\lambda_m\le\lambda_j(\mathbf{A}_i^k)\le\lambda_M,\qquad k\ge0,\ j=1,...,NM\tag{8}$$

where $\lambda_m=\min\{2\mu,\frac{\alpha-\beta}{2\epsilon},\frac{\beta}{2\epsilon}\}$ and their condition numbers have above bounded by a positive constant $L\le\frac{\lambda_M}{\lambda_m}$.
In [1, subsection 2.2.1], it is proved that the directions $\mathbf{d}_i^k$, $i=1,2,3$, computed at any k–iteration of BCDA by PCG with a suitable stopping rule are gradient related search directions. Consequently, since BCDA is a special version of the Algorithm 1 in [11], Theorem 7.1 in [11] states that $\nabla F_\epsilon(\mathbf{s}^k,\mathbf{z}^k,\mathbf{u}^k)\to0$ as $k\to\infty$ and there exists at least a limit point of $\{\mathbf{s}^k,\mathbf{z}^k,\mathbf{u}^k\}$ in $\mathcal{L}_{F_\epsilon^0}$ that is a stationary point of F_ϵ.
The following proposition resumes the features of $\mathbf{d}_i^k$, $i=1,2,3$, $k\ge0$.

Proposition 3.1. *[1] Let assume that $\nabla_{\mathbf{y}_i}F_\epsilon(\mathbf{y}^k)=\mathbf{A}_i^k\mathbf{y}_i^k-\mathbf{b}_i\ne\mathbf{0}$ for $k\ge0$. Let consider the PCG method applied to the symmetric positive definite system $\mathbf{A}_i^k\mathbf{d}_i=\mathbf{b}_i-\mathbf{A}_i^k\mathbf{y}_i^k$. Let $\mathbf{d}_i^h$ be the vector satisfying at the h–iteration of the PCG method the (stopping) rule*

$$\|\mathbf{r}^h\|\le\eta_i^k\|\mathbf{A}_i^k\mathbf{y}_i^k-\mathbf{b}_i\|\qquad\textit{with }\eta_i^k\le\frac{c}{\sqrt{K(\mathbf{A}_i^k)}},\tag{9}$$

Algorithm 1 BCDA

Step 0: Given $\mathbf{s}^0$, $\mathbf{z}^0$, $\mathbf{u}^0$, $0 < \rho_i < 1$, $\gamma_i \in (0, 2(1-\rho_i)]$, $i = 1, 2, 3$, and an exit tolerance θ_{outer};

Step 1: $k = 0$

Step 2: Inexact minimization with respect to $\mathbf{s}$ and $\mathbf{z}$:

1. compute the search directions $\mathbf{d}_1^k$ and $\mathbf{d}_2^k$;
2. compute $\alpha_1^k = \gamma_1 \frac{(\mathbf{b}_1 - \mathbf{A}_1^k \mathbf{s}^k)^T \mathbf{d}_1^k}{\mathbf{d}_1^{k\,T} \mathbf{A}_1^k \mathbf{d}_1^k}$, $\alpha_2^k = \gamma_2 \frac{(\mathbf{b}_2 - \mathbf{A}_2^k \mathbf{z}^k)^T \mathbf{d}_2^k}{\mathbf{d}_2^{k\,T} \mathbf{A}_2^k \mathbf{d}_2^k}$
3. update $\mathbf{s}^{k+1} = \mathbf{s}^k + \alpha_1^k \mathbf{d}_1^k$; $\mathbf{z}^{k+1} = \mathbf{z}^k + \alpha_2^k \mathbf{d}_2^k$.

Step 3: Inexact minimization with respect to $\mathbf{u}$:

1. compute the search directions $\mathbf{d}_3^k$;
2. compute $\alpha_3^k = \gamma_3 \frac{(\mathbf{b}_3 - \mathbf{A}_3^k \mathbf{u}^k)^T \mathbf{d}_3^k}{\mathbf{d}_3^{k\,T} \mathbf{A}_3^k \mathbf{d}_3^k}$
3. update $\mathbf{u}^{k+1} = \mathbf{u}^k + \alpha_3^k \mathbf{d}_3^k$.

Step 4: if $(F_\epsilon(\mathbf{y}^k) - F_\epsilon(\mathbf{y}^{k+1})) \leq \theta_{outer} F_\epsilon(\mathbf{y}^{k+1})$ then stop; else $k = k + 1$ and go to Step 2.

---

where $\mathbf{r}^h = \mathbf{b}_i - \mathbf{A}_i^k \mathbf{y}_i^k - \mathbf{A}_i^k \mathbf{d}_i^h$ *is the residual of the linear system,* $K(\mathbf{A}_i^k)$ *is the spectral condition number of* $\mathbf{A}_i^k$ *and* $0 < c < 1$.
Thus the direction $\mathbf{d}_i^k := \mathbf{d}_i^h$ *is a* gradient related search direction*, i.e.*

$$\begin{aligned} \frac{\nabla_{\mathbf{y}_i} F_\epsilon(\mathbf{y}^k)^T \mathbf{d}_i^k}{\|\mathbf{d}_i^k\|} &\leq \frac{1}{2(1+\eta_i^k)} \left((\eta_i^k)^2 - \frac{1}{K(\mathbf{A}_i^k)} \right) \|\nabla_{\mathbf{y}_i} F_\epsilon(\mathbf{y}^k)\| \\ &\leq \frac{c^2 - 1}{4L} \|\nabla_{\mathbf{y}_i} F_\epsilon(\mathbf{y}^k)\| \end{aligned} \tag{10}$$

with $c^2 - 1 < 0$.

Since F_ϵ is a KL function (**P8**), we can obtain also the convergence of the sequence of iterates $\{\mathbf{y}^k\}$ generated by BCDA to some critical point of F_ϵ in $\mathcal{L}_{F_\epsilon^0}$. Indeed, the analysis of an abstract descent algorithm for a KL function and the convergence results of Theorem 2.9 in [15] enables us to obtain similar results for BCDA. In particular, in view of the continuity of F_ϵ, we have that Theorem 2.9 in [15] assures that the sequence $\{\mathbf{y}^k\}$ generated by BCDA method converges to some critical point of F_ϵ in $\mathcal{L}_{F_\epsilon^0}$, if the following conditions hold for $\{\mathbf{y}^k\}$:

- *sufficient decrease condition*

$$F_\epsilon(\mathbf{y}^{k+1}) + C_1 \|\mathbf{y}^{k+1} - \mathbf{y}^k\|^2 \leq F_\epsilon(\mathbf{y}^k), \tag{11}$$

- *relative error condition*

$$\|\nabla F_\epsilon(\mathbf{s}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1})\| \leq C_2 \|\mathbf{y}^{k+1} - \mathbf{y}^k\|, \tag{12}$$

where C_1 and C_2 are fixed positive constant. The following two propositions verify the validity of the two conditions (11) and (12) for a generic iteration of BCDA.

Proposition 3.2. *At any k–iteration of BCDA, $k \geq 0$, while $\nabla F_\epsilon(\mathbf{y}^k) \neq 0$, we have that* (11) *holds with $0 < C_1 \leq \lambda_m \min_i \frac{\rho_i}{\gamma_i}$.*

Proof. In view of the Steps 2.2, 2.3, 3.2 and 3.3 of BCDA, we have that

$$\|\mathbf{y}^{k+1} - \mathbf{y}^k\|^2 = \sum_{i=1}^{3} (\alpha_i^k)^2 \|\mathbf{d}_i^k\|^2 = \sum_{i=1}^{3} \alpha_i^k \gamma_i \frac{-(\mathbf{A}_i^k \mathbf{y}_i^k - \mathbf{b}_i)^T \mathbf{d}_i^k}{{\mathbf{d}_i^k}^T \mathbf{A}_i^k \mathbf{d}_i^k} \|\mathbf{d}_i^k\|^2, \tag{13}$$

where $\mathbf{A}_1^k = \mathbf{A}_1(\mathbf{u}^k)$, $\mathbf{A}_2^k = \mathbf{A}_2(\mathbf{u}^k)$ and $\mathbf{A}_3^k = \mathbf{A}_3^k(\mathbf{s}^{k+1}, \mathbf{z}^{k+1})$. In view of the well-known inequality related to the maximum and the minimum eigenvalues $\lambda_{max}(\mathbf{A}_i^k)$ and $\lambda_{min}(\mathbf{A}_i^k)$ of the matrix $\mathbf{A}_i^k$,

$$\lambda_{max}(\mathbf{A}_i^k) \geq \frac{(\mathbf{d}_i^k)^T \mathbf{A}_i^k \mathbf{d}_i^k}{(\mathbf{d}_i^k)^T \mathbf{d}_i^k} \geq \lambda_{min}(\mathbf{A}_i^k), \tag{14}$$

from (8) and (7), we obtain

$$\begin{aligned}
\|\mathbf{y}^{k+1} - \mathbf{y}^k\|^2 &\leq \sum_{i=1}^{3} \frac{\gamma_i}{\lambda_m} \alpha_i^k (-(\mathbf{A}_i^k \mathbf{y}_i^k - \mathbf{b}_i)^T \mathbf{d}_i^k) \\
&\leq \frac{1}{\lambda_m} \max_i \left\{ \frac{\gamma_i}{\rho_i} \right\} (F_\epsilon(\mathbf{s}^k, \mathbf{z}^k, \mathbf{u}^k) - F_\epsilon(\mathbf{s}^{k+1}, \mathbf{z}^k, \mathbf{u}^k) + \\
&+ F_\epsilon(\mathbf{s}^{k+1}, \mathbf{z}^k, \mathbf{u}^k) - F_\epsilon(\mathbf{s}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^k) \\
&+ F_\epsilon(\mathbf{s}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^k) - F_\epsilon(\mathbf{s}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1})) \\
&\leq \frac{1}{C_1} (F_\epsilon(\mathbf{y}^k) - F_\epsilon(\mathbf{y}^{k+1})),
\end{aligned}$$

where C_1 is a positive constant such that $C_1 \leq \lambda_m \min_i \frac{\rho_i}{\gamma_i}$. □

In the proof of the following proposition, it is crucial to observe that $\nabla_{\mathbf{s}} F_\epsilon(\mathbf{y}) = \nabla_{\mathbf{s}} F_\epsilon(\mathbf{s}, \mathbf{u})$, while $\nabla_{\mathbf{z}} F_\epsilon(\mathbf{y}) = \nabla_{\mathbf{z}} F_\epsilon(\mathbf{z}, \mathbf{u})$ and $\nabla_{\mathbf{u}} F_\epsilon(\mathbf{y}) = \nabla_{\mathbf{u}} F_\epsilon(\mathbf{s}, \mathbf{z}, \mathbf{u})$. Furthermore, we recall that the gradient of F_ϵ on the level set $\mathcal{L}_{F_\epsilon^0}$ is M-Lipschitz continuous (**P7**).

Proposition 3.3. *At any k–iteration of BCDA, $k \geq 0$, while $\nabla F_\epsilon(\mathbf{y}^k) \neq 0$, we have that* (12) *holds with $C_2 \geq \frac{\sqrt{48} L \lambda_M}{(1-c^2) \min_i \gamma_i} + 4M$.*

Proof. From (13), using the rule (6) and the inequality (8), we obtain

$$\begin{aligned}\|\mathbf{y}^{k+1}-\mathbf{y}^k\|^2 &= \sum_{i=1}^{3}(\alpha_i^k)^2\|\mathbf{d}_i^k\|^2 = \sum_{i=1}^{3}\gamma_1^2\frac{(-(\mathbf{A}_i^k\mathbf{y}_i^k-\mathbf{b}_i)^T\mathbf{d}_i^k)^2}{((\mathbf{d}_i^k)^T\mathbf{A}_i^k\mathbf{d}_i^k)^2}\|\mathbf{d}_i^k\|^2 \geq \\ &\geq \sum_{i=1}^{3}\frac{\gamma_i^2}{\lambda_M^2}\frac{(-(\mathbf{A}_i^k\mathbf{y}_i^k-\mathbf{b}_i)^T\mathbf{d}_i^k)^2}{\|\mathbf{d}_i^k\|^2}.\end{aligned} \tag{15}$$

In view of (10), we obtain

$$\begin{aligned}&\|\mathbf{y}^{k+1}-\mathbf{y}^k\|^2 \geq \\ &\geq \frac{(1-c^2)^2\min_i\gamma_i^2}{16L^2\lambda_M^2}(\|\mathbf{A}_1^k\mathbf{s}^k-\mathbf{b}_1\|^2+\|\mathbf{A}_2^k\mathbf{z}^k-\mathbf{b}_2\|^2+\|\mathbf{A}_3^k\mathbf{u}^k-\mathbf{b}_3\|^2) = \\ &= \frac{(1-c^2)^2\min_i\gamma_i^2}{16L^2\lambda_M^2} \\ &\quad(\|\nabla_{\mathbf{s}}F_\epsilon(\mathbf{s}^k,\mathbf{u}^k)\|^2+\|\nabla_{\mathbf{z}}F_\epsilon(\mathbf{z}^k,\mathbf{u}^k)\|^2+\|\nabla_{\mathbf{u}}F_\epsilon(\mathbf{s}^{k+1},\mathbf{z}^{k+1},\mathbf{u}^k)\|^2) \geq \\ &\geq c_2(\|\nabla_{\mathbf{s}}F_\epsilon(\mathbf{s}^k,\mathbf{u}^k)\|+\|\nabla_{\mathbf{z}}F_\epsilon(\mathbf{z}^k,\mathbf{u}^k)\|+\|\nabla_{\mathbf{u}}F_\epsilon(\mathbf{s}^{k+1},\mathbf{z}^{k+1},\mathbf{u}^k)\|)^2,\end{aligned} \tag{16}$$

where $0 < c_2 \leq \frac{(1-c^2)^2\min_i\gamma_i^2}{48L^2\lambda_M^2}$; the last inequality follows from the Cauchy-Schwarz inequality that holds for a generic n-vector $\mathbf{v}$ and a vector $\mathbf{1}$:

$$(\mathbf{1}^T\mathbf{v})^2 \leq n\sum_{i=1}^{n}v_i^2. \tag{17}$$

On the other hand, we can write

$$\begin{aligned}&\|\nabla F_\epsilon(\mathbf{s}^{k+1},\mathbf{z}^{k+1},\mathbf{u}^{k+1})\| \leq \\ &\leq \|\nabla_{\mathbf{s}}F_\epsilon(\mathbf{s}^{k+1},\mathbf{u}^{k+1})\|+\|\nabla_{\mathbf{z}}F_\epsilon(\mathbf{z}^{k+1},\mathbf{u}^{k+1})\|+\|\nabla_{\mathbf{u}}F_\epsilon(\mathbf{s}^{k+1},\mathbf{z}^{k+1},\mathbf{u}^{k+1})\| \leq \\ &\leq \|\nabla_{\mathbf{s}}F_\epsilon(\mathbf{s}^k,\mathbf{u}^k)\|+\|\nabla_{\mathbf{s}}F_\epsilon(\mathbf{s}^{k+1},\mathbf{u}^{k+1})-\nabla_{\mathbf{s}}F_\epsilon(\mathbf{s}^k,\mathbf{u}^k)\|+ \\ &+\|\nabla_{\mathbf{z}}F_\epsilon(\mathbf{z}^k,\mathbf{u}^k)\|+\|\nabla_{\mathbf{z}}F_\epsilon(\mathbf{z}^{k+1},\mathbf{u}^{k+1})-\nabla_{\mathbf{z}}F_\epsilon(\mathbf{z}^k,\mathbf{u}^k)\|+ \\ &+\|\nabla_{\mathbf{u}}F_\epsilon(\mathbf{s}^{k+1},\mathbf{z}^{k+1},\mathbf{u}^k)\|+\|\nabla_{\mathbf{u}}F_\epsilon(\mathbf{s}^{k+1},\mathbf{z}^{k+1},\mathbf{u}^{k+1})-\nabla_{\mathbf{u}}F_\epsilon(\mathbf{s}^k,\mathbf{z}^k,\mathbf{u}^k)\|+ \\ &+\|\nabla_{\mathbf{u}}F_\epsilon(\mathbf{s}^k,\mathbf{z}^k,\mathbf{u}^k)-\nabla_{\mathbf{u}}F_\epsilon(\mathbf{s}^{k+1},\mathbf{z}^{k+1},\mathbf{u}^k)\| \leq \\ &\leq (\|\nabla_{\mathbf{s}}F_\epsilon(\mathbf{s}^k,\mathbf{u}^k)\|+\|\nabla_{\mathbf{z}}F_\epsilon(\mathbf{z}^k,\mathbf{u}^k)\|+\|\nabla_{\mathbf{u}}F_\epsilon(\mathbf{s}^{k+1},\mathbf{z}^{k+1},\mathbf{u}^k)\|) \\ &+3\|\nabla F_\epsilon(\mathbf{y}^{k+1})-\nabla F_\epsilon(\mathbf{y}^k)\|+\|\nabla F_\epsilon(\mathbf{s}^{k+1},\mathbf{z}^{k+1},\mathbf{u}^k)-\nabla F_\epsilon(\mathbf{s}^k,\mathbf{z}^k,\mathbf{u}^k)\|.\end{aligned} \tag{18}$$

Thus, using (16) and the M-Lipschitz continuity of the gradient of F_ϵ on the level set $\mathcal{L}_{F_\epsilon^0}$, we obtain

$$\|\nabla F_\epsilon(\mathbf{s}^{k+1},\mathbf{z}^{k+1},\mathbf{u}^{k+1})\| \leq C_2\|\mathbf{y}^{k+1}-\mathbf{y}^k\|, \tag{19}$$

where C_2 is a positive constant such that $C_2 \geq \frac{1}{\sqrt{c_2}}+4M \geq \frac{\sqrt{48}L\lambda_M}{(1-c^2)\min_i\gamma_i}+4M$. □

Because of Theorem 2.9 in [15], the two previous propositions enable to obtain the following convergence result.

Proposition 3.4. *Given a starting point* $\mathbf{y}^0$*, the sequence* $\{\mathbf{y}^k\}$ *generated by BCDA converges to some critical point of* F_ϵ *in* $\mathcal{L}_{F_\epsilon^0}$*. Moreover the sequence* $\{\mathbf{y}^k\}$ *has finite length, i. e.* $\sum_{k=0}^{\infty} \|\mathbf{y}^{k+1} - \mathbf{y}^k\| < \infty$.

Since BCDA has the same properties of an abstract descent method satisfying (11), (12), and the discrete BZ function is a polynomial (that is a real analytic function), results about the rate of convergence of BCDA are obtained in Theorem 4 in [17]. Indeed in this case the desingularizing function related to the Kurdyka-Lojasiewicz property at a critical point of F_ϵ is of the form $\phi(t) = \frac{C}{\sigma} t^\sigma$, with $C > 0$ and $\sigma \in (0, 1]$; the convergence is obtained in a finite number of step for $\sigma = 1$, while for $\sigma \in (\frac{1}{2}, 1)$ we have exponential convergence and for $\sigma \in (0, \frac{1}{2})$ polynomial convergence.

We conclude the section, by stating in the following proposition a feature of BCDA method, useful for the next section.

Lemma 3.1. *Let assume that* K *iterations of BCDA are executed, with* $K \leq \overline{K}$*. Thus we have*

$$\frac{C_1}{\overline{K}} \|\mathbf{y}^K - \mathbf{y}^0\|^2 \leq \frac{C_1}{K} \|\mathbf{y}^K - \mathbf{y}^0\|^2 \leq F_\epsilon(\mathbf{y}^0) - F_\epsilon(\mathbf{y}^K) \tag{20}$$

Proof. We observe that

$$\|\mathbf{y}^K - \mathbf{y}^0\|^2 = \| \sum_{i=0}^{K-1} (\mathbf{y}^{i+1} - \mathbf{y}^i)\|^2 \leq (\sum_{i=0}^{K-1} \|\mathbf{y}^{i+1} - \mathbf{y}^i\|)^2$$
$$\leq K \sum_{i=0}^{K-1} \|\mathbf{y}^{i+1} - \mathbf{y}^i\|^2$$

where the last inequality follows from (17). From (11), it is immediate to obtain (20). □

4. Numerical minimization of F_ϵ: a parallel approach

In this section, we address the minimization problem of the discrete functional F_ϵ by subdividing the lattice Λ into p tiles $T_1, ..., T_p$, with $T_i \cap T_j = \emptyset$, $i \neq j$. This subdivision leads a partition of the variable $\mathbf{x} \equiv (\mathbf{s}, \mathbf{z}, \mathbf{u})$ into p blocks $\mathbf{x}_{T_1}, \mathbf{x}_{T_2}, ..., \mathbf{x}_{T_p}$, with $\mathbf{x}_{T_j} \in \mathbb{R}^{n_j}$, $j = 1, ..., p$, $\sum_{j=1}^p n_j = NM$. Here each block of variables $\mathbf{x}_{T_j}$ includes the approximations of the functions $s_{i,j}, z_{i,j}, u_{i,j}$ related to the points of Λ belonging to $n_j \equiv N_j \times M_j$ tile T_j, denoted by $\mathbf{s}_{T_j}, \mathbf{z}_{T_j}, \mathbf{u}_{T_j}$, with $\sum_{j=1}^p N_j = N, \sum_{j=1}^p M_j = M$.
In addition to this partition of Λ, we consider a further partition of Λ into partially overlapping p tiles S_j of size $(N_j + \nu) \times (M_j + \nu)$, where S_j is the tile T_j

with an outer frame of ν rows and columns of pixels, that is ν is the number of rows/columns of overlapping pixels and $S_j \supset T_j$. Denoting by $\mathbf{x}_{S_j}$ the variables related to S_j, we observe that the vector of variables $\mathbf{x}$ can be considered as the union of $\mathbf{x}_{S_j}$ and $\mathbf{x}_{\Lambda - S_j}$ for any $j = 1, ..., p$, and that $\mathbf{x}_{S_j}$ is also the union of $\mathbf{x}_{T_j} = (\mathbf{s}_{T_j}, \mathbf{z}_{T_j}, \mathbf{u}_{T_j})$ related to the tile T_j, and $\mathbf{x}_{B_j} = (\mathbf{s}_{B_j}, \mathbf{z}_{B_j}, \mathbf{u}_{B_j})$ related to the frame $B_j \equiv S_j - T_j$. Figure 1 contains a sketch of both the tiling procedures. In the following we describe the OPARBCDA approach, namely a parallel ver-

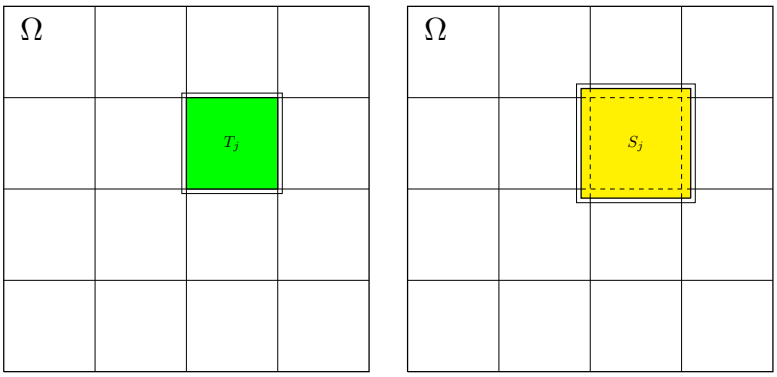

Figure 1: Tiling procedure.

sion of the BCDA scheme, which exploits a decomposition of the image domain in overlapping tiles and enables to address the segmentation of large images by solving a sequence of independent smaller problems, without any necessity of a post-processing procedure. After the introduction of the OPARBCDA method and the analysis of its theoretical and converging properties (Subsection 4.1), details of our practical parallel implementation are given in Subsection 4.2.

4.1. A parallel approach for a tiling procedure: the OPARBCDA method

Inspired by parallel connection schemes in [18] and [11], we propose a parallel version of the BCDA method, named OPARBCDA and detailed in Algorithm 2. We consider the same tiling partitions (with and without overlapping) already mentioned in the previous section.

In this scheme, starting from an initial point $\mathbf{x}^0$, at any ℓ iteration, we compute for each tile S_j, $j = 1, ..., p$, an inexact minimum point of the objective function restricted to the variables related to S_j, with boundary conditions given by the values of the previous iterate $\mathbf{x}^\ell$ on the frame of S_j. Then, from the inexact computed solution, only the entries corresponding to the tile T_j are extracted, neglecting the values on B_j. Then, the new iterate of OPARBCDA is updated by a *connection rule*, ensuring that the value of F_ϵ does not increase. In particular, exploiting the local features of the functional, the value of F_ϵ at the point $\widetilde{\mathbf{m}} = (\overline{\mathbf{x}}_1, \overline{\mathbf{x}}_2, ..., \overline{\mathbf{x}}_p)$ is computed and, if $F_\epsilon(\widetilde{\mathbf{m}}) \leq F_\epsilon(\mathbf{m}^j)$ for all $j = 1, ..., p$, we set $\mathbf{x}^{\ell+1} = \widetilde{\mathbf{m}}$. Otherwise, the new iterate is given by the rule $\mathbf{x}^{\ell+1} = \text{argmin}\{F_\epsilon(\mathbf{m}^1), ..., F_\epsilon(\mathbf{m}^p)\}$. This connection rule and a suitable

Algorithm 2 OPARBCDA

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

Step 1: $\ell = 0$; fix the parameters γ_i, ρ_i, $i = 1, 2, 3$ and $\overline{K}$ for BCDA;

Step 2: for $j = 1, ..., p$

if $\nabla_{\mathbf{x}_{T_j}} F_\epsilon(\mathbf{x}^\ell) \neq 0$ then
compute $\mathbf{m}^j = (\mathbf{x}_1^\ell, ..., \mathbf{x}_{j-1}^\ell, \overline{\mathbf{x}}_j, \mathbf{x}_{j+1}^\ell, ..., \mathbf{x}_p^\ell)$ as follows:

1. set $\mathbf{x}_{S_j}^0 = (\mathbf{x}^\ell)|_{S_j}$, $k = -1$;
2. repeat
   (a) $k = k + 1$; compute $\mathbf{x}_{S_j}^{k+1}$ by a step of BCDA; extract $\mathbf{x}_{T_j}^{k+1}$; set $\overline{\mathbf{x}}_j = \mathbf{x}_{T_j}^{k+1}$;
   (b) if $f_{\mathbf{x}_{S_j}}(\mathbf{x}_{T_j}^k; \mathbf{x}_{B_j}^0) - f_{\mathbf{x}_{S_j}}(\mathbf{x}_{T_j}^{k+1}; \mathbf{x}_{B_j}^0) < C_1 \|\mathbf{x}_{T_j}^k - \mathbf{x}_{T_j}^{k+1}\|^2$ then redefine $\overline{\mathbf{x}}_j = \mathbf{x}_{T_j}^k$ exit next j; end
   until $\|\nabla f_{\mathbf{x}_{S_j}}(\mathbf{x}_{S_j}^{k+1})\| \leq \theta_\ell \|\mathbf{x}_{S_j}^{k+1} - \mathbf{x}_{S_j}^\ell\|$

else
$\mathbf{m}^j = \mathbf{x}^\ell$;
end

Step 3: define the new iterate $\mathbf{x}^{\ell+1}$:

1. compute $F_\epsilon(\widetilde{\mathbf{m}})$ where $\widetilde{\mathbf{m}} = (\overline{\mathbf{x}}_1, \overline{\mathbf{x}}_2, ..., \overline{\mathbf{x}}_p)$
2. update $\mathbf{x}^{\ell+1} = \text{argmin}\{F_\epsilon(\widetilde{\mathbf{m}}), F_\epsilon(\mathbf{m}^1), ..., F_\epsilon(\mathbf{m}^p)\}$.

Step 4: 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 2.

---

implementation of the BCDA method to obtain an inexact minimum for each inner subproblem enable to apply Theorem 3.5 in [18], which guarantees the stationarity of any limit point of the sequence $\{\mathbf{x}^\ell\}$ and, since $\mathcal{L}_{F_\epsilon^0}$ is compact, also ensures that a limit point of $\{\mathbf{x}^\ell\}$ exists and the gradients sequence enjoys the following property: $\nabla F_\epsilon(\mathbf{x}^\ell) \to 0$ as $\ell \to \infty$. Indeed it is simple to verify that at any ℓ–iteration of the algorithm OPARBCDA the following conditions hold for $\ell \geq 0$:

$$F_\epsilon(\mathbf{x}^{\ell+1}) \leq F_\epsilon(\mathbf{m}^j) \leq F_\epsilon(\mathbf{x}^\ell) \quad j = 1, ..., p, \tag{21}$$

$$C_3 \|\nabla_{\mathbf{x}_{T_j}} F_\epsilon(\mathbf{x}^\ell)\|^2 \leq F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\mathbf{x}^{\ell+1}) \quad j = 1, ..., p, \tag{22}$$

where C_3 is a positive constant. In particular, condition (22) is preserved when

at the first step of BCDA a sufficient decrease of the objective $f_{\mathbf{x}_{S_j}}$ is assured with respect to the variables $\mathbf{x}_{T_j}$. In order to obtain this decrease, at the first step of BCDA, the sub–vectors $(\mathbf{d}_i^0)_{|T_j}$ of the computed descent directions $\mathbf{d}_i^0$, $i = 1, 2, 3$, restricted to entries of T_j, have to be gradient related with respect to $\mathbf{g}_1 \equiv \nabla_{\mathbf{s}_{T_j}} f_{\mathbf{x}_j}(\mathbf{s}_{T_j}^0, \mathbf{z}_{T_j}^0, \mathbf{u}_{T_j}^0; \mathbf{x}_{B_j}^0)$, $\mathbf{g}_2 \equiv \nabla_{\mathbf{z}_{T_j}} f_{\mathbf{x}_j}(\mathbf{s}_{T_j}^0, \mathbf{z}_{T_j}^0, \mathbf{u}_{T_j}^0; \mathbf{x}_{B_j}^0)$, $\mathbf{g}_3 \equiv \nabla_{\mathbf{u}_{T_j}} f_{\mathbf{x}_j}(\mathbf{s}_{T_j}^1, \mathbf{z}_{T_j}^1, \mathbf{u}_{T_j}^0; \mathbf{x}_{B_j}^0)$ respectively; using an argument similar to the one in the proof of Proposition 3.1, this condition is verified if the inner PCG scheme is stopped when the norm of the residual at the h–step, restricted to the entries related to T_j, satisfies the criterion:

$$\|(\mathbf{r}_{\mathbf{y}_i}^h)_{|T_j}\| \quad \leq \quad \frac{c}{\sqrt{K(A_i^0(T_j, T_j))}} \|\mathbf{g}_i\|, \qquad i = 1, 2, 3,$$

where, in concordance with the standard notation of the Section 3, $\mathbf{y}_1 = \mathbf{s}$, $\mathbf{y}_2 = \mathbf{z}$ and $\mathbf{y}_3 = \mathbf{u}$. From the practical point of view, since $\|(\mathbf{r}_{\mathbf{y}_i}^h)_{|T_j}\| \leq \|\mathbf{r}_{\mathbf{y}_i}^h\|$ and $K(A_i^0(T_j, T_j)) \leq K(A_i^0)$, in the standard stopping criterion of PCG (9) it must to replace the norm of the gradient of the functional with respect to the block variable $\mathbf{y}_i$ at $\mathbf{x}_{S_j}^0$ with the one of corresponding sub–vector $\|\mathbf{g}_i\|$related to T_j. Furthermore, at the first step of BCDA, a sufficient decrease of the objective function restricted to variables $(\mathbf{y}_i)_{T_j}$ is assured if the step-lengths α_i^0 satisfy (see (6)):

$$\alpha_i^0 \leq 2(1 - \rho_i) \frac{-g_i^T (\mathbf{d}_i^0)_{|T_j}}{(\mathbf{d}_i^0)_{|T_j}^T A_i^0(T_j, T_j)(\mathbf{d}_i^0)_{|T_j}}$$

Thus, since $(\mathbf{d}_i^0)^T A_i^0 \mathbf{d}_i^0 \geq (\mathbf{d}_i^0)_{|T_j}^T A_i^0(T_j, T_j)(\mathbf{d}_i^0)_{|T_j}$, we set $\alpha_i^0 = \gamma_i \frac{-g_i^T (\mathbf{d}_i^0)_{|T_j}}{\mathbf{d}_i^{0T} A_i^0 \mathbf{d}_i^0}$. Using the same argument as for the proofs of Propositions 3.3 and 3.2, we have

$$\begin{aligned}
&\|\nabla_{\mathbf{x}_{T_j}} F_\epsilon(\mathbf{x}^\ell)\| = \|\nabla_{\mathbf{x}_{T_j}} f_{\mathbf{x}_{S_j}}(\mathbf{y}^0)\| \leq \|\mathbf{g}_1\| + \|\mathbf{g}_2\| + \|\nabla_{\mathbf{u}_{T_j}} f_{S_j}(\mathbf{x}_{T_j}^0; \mathbf{x}_{B_j}^0)\| \leq \\
&\leq \|\mathbf{g}_1\| + \|\mathbf{g}_2\| + \|\mathbf{g}_3\| + \|\nabla_{\mathbf{u}_{T_j}} f_{S_j}(\mathbf{x}_{T_j}^0; \mathbf{x}_{B_j}^0) - \mathbf{g}_3\| \leq \\
&\leq \|\mathbf{g}_1\| + \|\mathbf{g}_2\| + \|\mathbf{g}_3\| + M\|\mathbf{x}_{T_j}^1 - \mathbf{x}_{T_j}^0\| \leq (M + \frac{1}{\sqrt{c_2}})\|\mathbf{x}_{T_j}^1 - \mathbf{x}_{T_j}^0\| \leq \\
&\leq \frac{(M + \frac{1}{\sqrt{c_2}})}{\sqrt{C_1}} \sqrt{f_{\mathbf{x}_{S_j}}(\mathbf{x}_{T_j}^0; \mathbf{x}_{B_j}^0) - f_{\mathbf{x}_{S_j}}(\mathbf{x}_{T_j}^1; \mathbf{x}_{B_j}^0)}. \qquad (23)
\end{aligned}$$

In the next iterations of BCDA, the control at the Step 2.2 (b) assures at each step a decrease of the objective function at most equal to $C_1\|\mathbf{x}_{T_j}^k - \mathbf{x}_{T_j}^{k+1}\|^2$. If the following condition

$$f_{S_j}(\mathbf{x}_{T_j}^k; \mathbf{x}_{B_j}^0) - f_{S_j}(\mathbf{x}_{T_j}^{k+1}; \mathbf{x}_{B_j}^0) \geq C_1\|\mathbf{x}_{T_j}^k - \mathbf{x}_{T_j}^{k+1}\|^2 \qquad (24)$$

is satisfied, the update of the iterate is performed and a new iteration is started. Otherwise, we put $\overline{\mathbf{x}}_j = \mathbf{x}_{T_j}^k$ and the inner solver is stopped. Indeed, condition (24) assumes the role of an inner stopping criterion. As consequence, at any outer ℓ–iteration of OPARBCDA we have that

$$F_\epsilon(\mathbf{x}^{\ell+1}) \leq F_\epsilon(\mathbf{m}^j) \leq F_\epsilon(\mathbf{x}^\ell) \qquad j = 1, ..., p$$

and, from (23), (24) and the connection rule, the inequality (22) remains satisfied with $C_3 \leq \frac{C_1}{(M+\frac{1}{\sqrt{c_2}})^2}$:

$$C_3\|\nabla_{\mathbf{x}_{T_j}} F_\epsilon(\mathbf{x}^\ell)\|^2 \leq (F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\mathbf{m}^j)) \leq (F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\mathbf{x}^{\ell+1})).$$

Thus, in view of Theorem 3.5 in [18], we can obtain the following convergence result.

Proposition 4.1. *A limit point of the sequence $\{\mathbf{x}^\ell\}$ generated by OPARBCDA is a stationary point of F_ϵ. Moreover, since $\mathcal{L}_{F_\epsilon^0}$ is compact, there exists at least a limit point of $\{\mathbf{x}^\ell\}$ and we have $\nabla F_\epsilon(\mathbf{x}^\ell) \to 0$ as $\ell \to \infty$.*

Before to prove that the whole sequence $\{\mathbf{x}^\ell\}$ generated by OPARBCDA converges to some critical point of F_ϵ in $\mathcal{L}_{F_\epsilon^0}$, we recall a property of functions having the KL property, proved in [14, 16].

Lemma 4.1. *([14, Lemma 1],[16, Lemma 6]) Let $\Omega \in \mathbb{R}^n$ be a compact set and let $f : \mathbb{R}^n \to (-\infty, \infty]$ be a proper and lower semicontinuous function. Assume that f is constant on Ω and satisfies the KL property at each point of Ω. Then, there exists $\sigma > 0$, $\eta > 0$ and a continuous and concave function $\phi : [0, \eta) \to [0, \infty)$, which is continuously differentiable on $(0, \eta)$ and satisfies $\phi(0) = 0$, $\phi' > 0$ on $(0, \eta)$, such that*

$$\phi'(f(\mathbf{x}) - f(\overline{\mathbf{x}}))dist(0, \partial f(\mathbf{x})) \geq 1 \tag{25}$$

for every $\overline{\mathbf{x}} \in \Omega$ and every $\mathbf{x}$ such that $dist(\mathbf{x}, \Omega) < \sigma$ and $f(\overline{\mathbf{x}}) < f(\mathbf{x}) < f(\overline{\mathbf{x}}) + \eta$.

By following similar arguments to those of Theorem 1 in [14], we can prove the convergence of the whole sequence $\{\mathbf{x}^\ell\}$ generated by OPARBCDA.

Proposition 4.2. *Given a starting point $\mathbf{x}^0$, the sequence $\{\mathbf{x}^\ell\}$ generated by OPARBCDA has finite length, i. e. $\sum_{\ell=0}^{\infty} \|\mathbf{x}^{\ell+1} - \mathbf{x}^\ell\| < \infty$ and, thus, it converges to some critical point of F_ϵ in $\mathcal{L}_{F_\epsilon^0}$.*

Proof. Let K_j the number of iterations performed by BCDA on the tile S_j. For any j, the following relations hold true:

$$\begin{aligned} f_{\mathbf{x}_{S_j}}(\mathbf{x}^0_{T_j}; \mathbf{x}^0_{B_j}) - f_{\mathbf{x}_{S_j}}(\mathbf{x}^{K_j}_{T_j}; \mathbf{x}^0_{B_j}) &= \sum_{k=0}^{K_j-1} f_{\mathbf{x}_{S_j}}(\mathbf{x}^k_{T_j}; \mathbf{x}^0_{B_j}) - f_{\mathbf{x}_{S_j}}(\mathbf{x}^{k+1}_{T_j}; \mathbf{x}^0_{B_j}) \\ &\geq \sum_{k=0}^{K_j-1} C_1\|\mathbf{x}^k_{T_j} - \mathbf{x}^{k+1}_{T_j}\|^2 \geq \frac{C_1}{K_j}\left(\sum_{k=0}^{K_j-1} \|\mathbf{x}^k_{T_j} - \mathbf{x}^{k+1}_{T_j}\|\right)^2 \\ &\geq \frac{C_1}{\overline{K}}\left(\sum_{k=0}^{K_j-1} \|\mathbf{x}^k_{T_j} - \mathbf{x}^{k+1}_{T_j}\|\right)^2 \end{aligned} \tag{26}$$

where the first inequality is a consequence of (24) while the second one follows from (17); $\overline{K}$ is the prefixed maximum number of iterations for the inner BCDA method.
Moreover we can state that, whatever the result of the connection rule may be, the values of the objective function F_ϵ obey to the following condition:

$$\begin{aligned} F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\mathbf{x}^{\ell+1}) &\geq \frac{1}{p}\sum_{j=1}^{p} F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\mathbf{m}^j) = \\ &= \frac{1}{p}\sum_{j=1}^{p} f_{\mathbf{x}_{S_j}}(\mathbf{x}^0_{T_j};\mathbf{x}^0_{B_j}) - f_{\mathbf{x}_{S_j}}(\mathbf{x}^{K_j}_{T_j};\mathbf{x}^0_{B_j}) \\ &\geq \frac{C_1}{\overline{K}p}\sum_{j=1}^{p}\left(\sum_{k=0}^{K_j-1}\|\mathbf{x}^k_{T_j} - \mathbf{x}^{k+1}_{T_j}\|\right)^2 \geq \frac{C_1}{\overline{K}p^2}\left(\sum_{j=1}^{p}\sum_{k=0}^{K_j-1}\|\mathbf{x}^k_{T_j} - \mathbf{x}^{k+1}_{T_j}\|\right)^2 \end{aligned} \tag{27}$$

where the first inequality follows from (21) and the second one from (26).
On the other hand, the gradient of the function F_ϵ has the following property:

$$\begin{aligned} \|\nabla F_\epsilon(\mathbf{x}^\ell)\| &\leq \sum_{j=1}^{p}\|\nabla_{\mathbf{x}_{T_j}} F_\epsilon(\mathbf{x}^\ell)\| \leq C_4\sum_{j=1}^{p}\|\mathbf{x}^1_{T_j} - \mathbf{x}^0_{T_j}\| \\ &\leq C_4\sum_{j=1}^{p}\sum_{k=0}^{K_j-1}\|\mathbf{x}^{k+1}_{T_j} - \mathbf{x}^k_{T_j}\| \end{aligned} \tag{28}$$

where to obtain the first inequality we employ (23) and $C_4 \leq M + \dfrac{1}{\sqrt{c_2}}$.
Let $\omega(\mathbf{x}^0)$ be the set of limit points of the sequence $\{\mathbf{x}^\ell\}$ generated by OPAR-BCDA. By the boundedness of the sequence $\{\mathbf{x}^\ell\}$,we obtain that $\lim_{\ell\to+\infty} dist(\mathbf{x}^\ell, \omega(\mathbf{x}^0)) = 0$, $\omega(\mathbf{x}^0)$ is a nonempty, compact and connected set and the objective function F_ϵ is finite and constant on $\omega(\mathbf{x}^0)$ (Lemma 5 in [16]). Since $\{\mathbf{x}^\ell\}$ is bounded, there exists a subsequence $\{\mathbf{x}^{\ell_q}\}$ such that $\mathbf{x}^{\ell_q} \to \overline{\mathbf{x}}$ as $q \to \infty$. For continuity of F_ϵ, $F_\epsilon(\mathbf{x}^\ell) \to F_\epsilon(\overline{\mathbf{x}})$ as $\ell \to \infty$. Since $\{F_\epsilon(\mathbf{x}^\ell)\}$ is a nonincreasing sequence, we have $F_\epsilon(\overline{\mathbf{x}}) < F_\epsilon(\mathbf{x}^\ell)$, for $\ell \geq 0$. Thus, for any $\eta > 0$, there exists a nonnegative integer ℓ_0 such that $F_\epsilon(\mathbf{x}^\ell) < F_\epsilon(\overline{\mathbf{x}}) + \eta$ for all $\ell > \ell_0$. On the other hand, since $\lim_{\ell\to\infty} \mathrm{dist}(\mathbf{x}^\ell, \omega(\mathbf{x}^0)) = 0$, for any $\sigma > 0$, there exists a positive integer ℓ_1 such that $\mathrm{dist}(\mathbf{x}^\ell, \omega(\mathbf{x}^0)) < \sigma$ for all $\ell > \ell_1$. Thus, from Lemma 4.1, for any $\ell > \overline{\ell} = \max\{\ell_0, \ell_1\}$, since F_ϵ is finite and constant on $\omega(\mathbf{x}^0)$, there exists a continuous and concave function $\phi : [0, \eta] \to [0, \infty)$, which is continuously differentiable on $(0, \eta)$ and satisfies $\phi(0) = 0$, $\phi' > 0$ on $(0, \eta)$, such that

$$\phi'(F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\overline{\mathbf{x}}))\|\nabla F_\epsilon(\mathbf{x}^\ell)\| \geq 1\,. \tag{29}$$

Concavity of ϕ implies

$$\phi(F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\overline{\mathbf{x}})) - \phi(F_\epsilon(\mathbf{x}^{\ell+1}) - F_\epsilon(\overline{\mathbf{x}})) \geq \phi'(F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\overline{\mathbf{x}}))(F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\mathbf{x}^{\ell+1}))\,.$$

By using (29), we obtain

$$\phi(F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\overline{\mathbf{x}})) - \phi(F_\epsilon(\mathbf{x}^{\ell+1}) - F_\epsilon(\overline{\mathbf{x}})) \geq \frac{1}{\|\nabla F_\epsilon(\mathbf{x}^\ell)\|}(F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\mathbf{x}^{\ell+1}))\,.$$

Furthermore, in view of (27) and (28), the previous inequality can be written as

$$\begin{aligned}
&\phi(F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\overline{\mathbf{x}})) - \phi(F_\epsilon(\mathbf{x}^{\ell+1}) - F_\epsilon(\overline{\mathbf{x}})) \geq \\
&\geq \frac{\frac{C_1}{\overline{K}p^2}\left(\sum_{j=1}^p \sum_{k=0}^{K_j-1} \|\mathbf{x}_{T_j}^k - \mathbf{x}_{T_j}^{k+1}\|\right)^2}{C_4 \sum_{j=1}^p \sum_{k=0}^{K_j-1} \|\mathbf{x}_{T_j}^{k+1} - \mathbf{x}_{T_j}^k\|} = \\
&= \frac{C_1}{C_4 \overline{K} p^2} \sum_{j=1}^p \sum_{k=0}^{K_j-1} \|\mathbf{x}_{T_j}^{k+1} - \mathbf{x}_{T_j}^k\|\,.
\end{aligned} \tag{30}$$

Finally, since if $\mathbf{x}^{\ell+1} = \widetilde{\mathbf{m}}$ it holds that

$$\|\mathbf{x}^{\ell+1} - \mathbf{x}^\ell\| \leq \sum_{j=1}^p \|\mathbf{x}_{T_j}^{K_j} - \mathbf{x}_{T_j}^0\| \leq \sum_{j=1}^p \sum_{k=0}^{K_j-1} \|\mathbf{x}_{T_j}^{k+1} - \mathbf{x}_{T_j}^k\|$$

and if $\mathbf{x}^{\ell+1} = \mathbf{m}^{\overline{j}}$ for a suitable $\overline{j}$, it holds that

$$\|\mathbf{x}^{\ell+1} - \mathbf{x}^\ell\| = \|\mathbf{x}_{T_{\overline{j}}}^{K_{\overline{j}}} - \mathbf{x}_{T_{\overline{j}}}^0\| = \sum_{j=1}^p \|\mathbf{x}_{T_j}^{K_j} - \mathbf{x}_{T_j}^0\| \leq \sum_{j=1}^p \sum_{k=0}^{K_j-1} \|\mathbf{x}_{T_j}^{k+1} - \mathbf{x}_{T_j}^k\|\,,$$

we can conclude that

$$\|\mathbf{x}^{\ell+1} - \mathbf{x}^\ell\| \leq \frac{C_4 \overline{K} p^2}{C_1}\big(\phi(F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\overline{\mathbf{x}})) - \phi(F_\epsilon(\mathbf{x}^{\ell+1}) - F_\epsilon(\overline{\mathbf{x}}))\big)\,.$$

Summing from $i = \overline{\ell} + 1$ to ℓ gives

$$\sum_{i=\overline{\ell}+1}^{\ell} \|\mathbf{x}^{i+1} - \mathbf{x}^i\| \leq \frac{C_4 \overline{K} p^2}{C_1}\big(\phi(F_\epsilon(\mathbf{x}^{\overline{\ell}+1}) - F_\epsilon(\overline{\mathbf{x}})) - \phi(F_\epsilon(\mathbf{x}^{\ell+1}) - F_\epsilon(\overline{\mathbf{x}}))\big)$$

and since the term on the right hand side is bounded, the series on the left converges. This shows that $\{\mathbf{x}^\ell\}$ is a Cauchy sequence, which converges to some $\overline{\mathbf{x}} \in \omega(\mathbf{x}^0)$. □

Following [17], we can obtain the following results about the convergence rate of OPARBCDA.

Proposition 4.3. *Let $\Phi : (0, \eta) \to [0, \infty)$ be any primitive of $-(\phi')^2$.*

(i) If $\lim_{\ell\to\infty} \Phi(r)$ is finite, the algorithm converges in a finite number of steps.

(ii) If $\lim_{\ell\to\infty} \Phi(r) = \infty$, there exists $\overline{\ell}$ such that

- $F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\overline{\mathbf{x}}) = \mathcal{O}(\Phi^{-1}((\ell+1-\overline{\ell})\frac{C_1}{\overline{K}p^2C_4^2})$
- $\|\mathbf{x}^\ell - \overline{\mathbf{x}}\| = \mathcal{O}(\phi \circ \Phi^{-1}((\ell+1-\overline{\ell})\frac{C_1}{\overline{K}p^2C_4^2}))$

Proof. Let $\overline{\ell}$ be large enough to have that for all $\ell \geq \overline{\ell}$ the KL inequality (29) holds. We denote $r^\ell = F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\overline{\mathbf{x}})$, where $\overline{\mathbf{x}} \in \omega(\mathbf{x}^0)$. Thus, we have

$$\begin{aligned}
&\phi'(r^\ell)^2(F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\mathbf{x}^{\ell+1})) \geq \\
&\geq \phi'(r^\ell)^2 \frac{C_1}{\overline{K}p^2} \left(\sum_{j=1}^{p} \sum_{k=0}^{K_j-1} \|\mathbf{x}_{T_j}^k - \mathbf{x}_{T_j}^{k+1}\| \right)^2 \geq \\
&\geq \phi'(r^\ell)^2 \frac{C_1}{\overline{K}p^2C_4^2} \|\nabla F_\epsilon(\mathbf{x}^\ell)\|^2 \geq \frac{C_1}{\overline{K}p^2C_4^2}
\end{aligned} \tag{31}$$

where the first inequality follows from (27), the second from (28) and the last from the KL inequality (29).
Let Φ be a primitive of $-(\phi')^2$. Then, since ϕ is a nonincreasing , we have

$$\begin{aligned}
&\Phi(r^{\ell+1}) - \Phi(r^\ell) = \int_{r^{\ell+1}}^{r^\ell} \phi'(t)^2 dt \\
&\geq \phi'(r^\ell)^2(F_\epsilon(\mathbf{x}^\ell) - F_\epsilon(\mathbf{x}^{\ell+1}))
\end{aligned} \tag{32}$$

Combining (31) and (32), we obtain

$$\Phi(r^{\ell+1}) - \Phi(r^{\overline{\ell}}) = \sum_{i=\overline{\ell}}^{\ell} \Phi(r^{i+1}) - \Phi(r^i) \geq (\ell+1-\overline{\ell})\frac{C_1}{\overline{K}p^2C_4^2}. \tag{33}$$

Thus the proof follows from the same argument of [17, Theorem 5, equation (9)]. □

The theoretical results about the convergence and the rate of convergence of OPARBCDA does not depend on the size ν of the frame of each tile T_j. Nevertheless, practical experience shows that a selection of $\nu > 0$ decreases the number of external iterations, above all for a subdivision into tiles of small size (see the numerical results in section 5.2). When $\nu > 0$ the iterations are very few, less than ten, also for huge images. Indeed, in the connection rule at the Step 3, the selection $\nu > 0$ facilitates the choices $\mathbf{x}^{\ell+1} = \widetilde{\mathbf{m}}$ rather than $\mathbf{x}^{\ell+1} = \mathbf{m}_j$, for a suitable j, fully exploiting the parallel processing of each tile. When $\nu = 0$, at some initial iteration and/or for small size tiles, it can arises that $F_\epsilon(\widetilde{\mathbf{m}}) > \min_j\{F_\epsilon(\mathbf{m}_j)\}$, producing a degeneration of the performance of OPARBCDA. A possible motivation of this behavior is as follows: for a given $\mathbf{x}^\ell$, when we compare the vector $\overline{\mathbf{x}}_{j,S_j}$ obtained extracting the portion $\mathbf{x}_{T_j}^k$ from the approximation on the enlarged tile S_j and the vector $\overline{\mathbf{x}}_{j,T_j}$ obtained as approximation of the minimum of F_ϵ restricted to T_j, we observe that $F_\epsilon(\overline{\mathbf{x}}_{j,T_j}; (\mathbf{x}^\ell)_{|\lambda - T_j}) \leq F_\epsilon(\overline{\mathbf{x}}_{j,S_j}; (\mathbf{x}^\ell)_{|\lambda - T_j})$, since $\overline{\mathbf{x}}_{j,T_j} \sim \mathrm{argmin}_{\mathbf{x}_{T_j} \in T_j} f_{\mathbf{x}_{T_j}}(\mathbf{x}_{T_j}; \mathbf{x}_{B_j^0})$; nevertheless, on the boundary of T_j, the

vector $\overline{\mathbf{x}}_{j,S_j}$ matches better than $\overline{\mathbf{x}}_{j,T_j}$ with the vectors related to adjacent tiles, since it is obtained taking into account of the behavior of F_ϵ on ν rows/columns of these adjacent tiles. Thus we have $F_\epsilon(\widetilde{\mathbf{m}}) \leq \min_j\{F_\epsilon(\mathbf{m}_j)\}$ and few iterations are enough to obtain satisfactory results.

4.2. Parallel implementation

As we can see from Algorithm 1 of Section 3, the main computational bottleneck of BCDA algorithm is the inexact minimization step of the functional restricted to one of the blocks $(\mathbf{u}, \mathbf{s}, \mathbf{z})$: few iterations of a PCG algorithm are required for the evaluation of the descent direction.

For taking advantage of all the cores in a commodity CPU, parallel linear algebra libraries can be then adopted to speed up the segmentation problem. To this extent we adopted Thrust [19] library: that offers the possibility to target different architectures by selecting, at compile-time, the parallelization backend. Besides GPU support, two multicore CPU approaches are offered: OpenMP and TBB; in this approach we used the former. The library takes care of the parallelization of linear algebra routines: in this BCDA implementation we then used Thrust provided implementation of vector norms and vector updates (axpy).
Concerning the evaluation of the elements of matrices $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, custom code has been developed and parallelized through OpenMP `parallel for` directive.
Diagonal preconditioner and linear CG are already offered by the library that, in turns, offers the possibility to introduce custom preconditioners. This chance is exploited to develop a block tridiagonal symmetric preconditioner class: factorization step can be easily parallelized, since we are facing a block-diagonal structure: for each block it's LDL factorization is independent and can be concurrently performed.
Matrix-vector product routines involve banded matrices: a tabular structure is used to memorize diagonals in compact vectors. Implementation relies on Cusp [20] library, that provides dedicated `dia_matrix` class. We make note that underlying implementation resides on aforementioned Thrust blas routines.
This first approach allows one to implement a version of BCDA that contains parallel matrix vector product subroutines, plus a number of blas-1 subroutines for norms and dot products. When varying the number of threads, memory–bound nature of this problem inhibits a satisfactory decrease of the overall computational time: parallelization speedup is low due to poor local data reuse. It is worth noting that, even if each subroutine is able to split the computational task among more than one core, there is no data reuse: each matrix-vector evaluation involves a complete scan of $(\mathbf{u}, \mathbf{s}, \mathbf{z})$ vectors, no temporal locality on data access is exploited. As a consequence, when large images are considered, no speedup is achieved: an example of the behaviour of this approach can be found on Table 7 of Section 5.
Results point out the need to find an approach that increases data locality: algorithm 2 offers this kind of feature by partitioning data and variables and considering independent subproblems.
In this second approach we are facing a decrease of data dimensionality: vari-

ables are more likely to fit in the hardware cache, thus leveraging the impact of an extensive memory access.

Concerning Algorithm 2, the intrinsic parallel features of Steps 2 and 3 are exploited in two different ways.

Tiling technique previously described is exploited in order to generate, at Step 2, a number of independent tasks that can be concurrently solved; concerning Step 3, OpenMP `parallel for` directive is used when evaluating the objective function.

Due to iterative nature of inner BCDA solver (Step 2), different running times are expected for the solution of separate subproblems: to overcome this disadvance we adopted manager/workers pattern [21] that ensures run-time distribution of independent tasks among POSIX threads.

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 collects subproblems computed solutions. Mutex-protected queues collect both task input and output results, as a consequence two different queues are present in the implementation:

- a job queue: a single manager is the producer of the queue elements, while all workers are consumers;
- a results queue: in this case each worker fills the queue with results of assigned subproblems, while the manager is responsible to insert them in the overall segmentation variables $(\mathbf{u}, \mathbf{s}, \mathbf{z})$.

Both cases can be handled by the same implementation that provides:

- a thread–safe interface for insert/remove operations;
- a signaling mechanism for the communication of available resources.

We provide a simple C++ class that stores resources in a private `std::queue<T>` variable, while exposes only two methods `push` and `pop` for resource insertion and removal, respectively. This implementation can be used in conjunction of POSIX Threads [22], since additional private members are present:

- a mutex variable of type `pthread_mutex_t`, used a safeguard for the shared resource;
- a condition variable of type `pthread_cond_t`, associated to previously mentioned mutex, for signaling procedures.

Such implementation choice allows for a mutually exclusive access to internal queue in multi-threading environment. Moreover, through the adoption of a condition variable, producer threads can communicate information about the state of shared data: for example to signal that a queue is no longer empty; an exaustive description of this approach can be found on Chapter 3 of [22]. In order to provide a reliable queue implementation even in presence of exceptions,

RAII [23] programming idiom is adopted when locking/unlocking operations are executed on a mutex.

Job queue is used to communicate both commands and data from master to workers: in this implementation, only two basic job types are used. First job type contains a complete description of one of the tasks (references to subproblem local data, objective function parameters and algorithm parameters) generated in Step 2 of Algorithm 2. A second type of job is used by master thread in order to ensure the clean termination of workers threads. Each worker thread code is structured as a while loop: as long as the thread can pick a subproblem description, it solves it and puts the results on results queue; when a termination job is picked, the thread exits.

Finally, as regards Step 3, OpenMP compiler directive `omp parallel for` is used for evaluation of $F_\epsilon(\widetilde{\mathbf{m}})$. While this step is performed, worker threads are waiting on a POSIX condition variable, without requiring cpu time.

5. Numerical experiments

Experiments are performed on a dual–head commodity PC equipped with two 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. Total number of cores is 16, and Intel(R) Hyper-Threading Technology is disabled. For each configuration settings, displayed computational time is obtained as the average of ten different runs; limit wall clock time is limited to 4 hours.

5.1. Test Problems

In order to evaluate the effectiveness of the proposed parallel approach, we consider a Trento DSM [24] of size 2020×2020, spatial resolution of 1 mt and range of data $\mathbf{g}$ between 183.770 and 971.916. For assessing the behaviour of the method when increasing image size a second test is presented: we choose a 16184×15984 image QuickBird [25], whose pixel values are in 0 and $5.22 \cdot 10^3$ range. For each dataset, a 600×600 portion of input image is displayed in Figure 2.

Both tests reference solutions are obtained by applying the BCDA method to each image, using stop tolerance $\theta_{outer} = 5 \cdot 10^{-4}$. In order to compare the effectiveness of BCDA with respect to OPARBCDA, the latter is stopped at the first outer iteration that shows an objective function value lower than the one obtained by BCDA.

Concerning the first dataset, ground truth solution is obtained again with BCDA, fixed stopping threshold as $\theta_{outer} = 10^{-10}$ and selecting a tolerance of 10^{-12} for inner PCG. The size of second dataset inhibits the possibility of having an estimated ground truth: indeed in double precision floating point arithmetic, this second test requires approximately 2 GB only for storing the measured image.

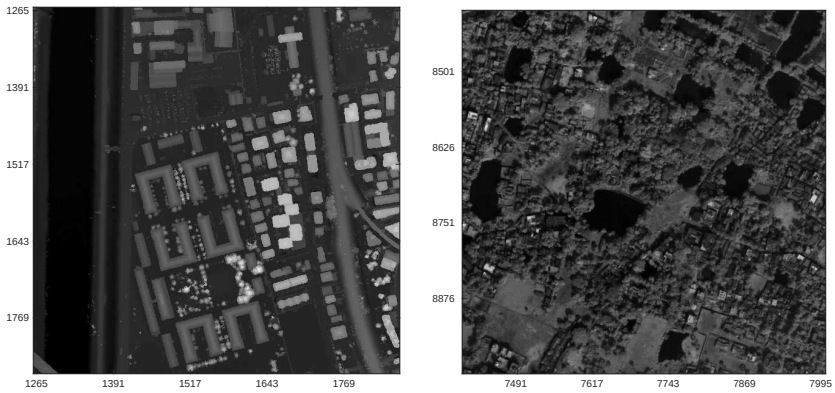

Figure 2: Left panel: portion of Trento image, right panel: portion of QuickBird image.

5.2. Small–size dataset segmentation

For test problem Trento, BCDA and OPARBCDA solutions are compared with a computed ground truth image. Table 1 collects the final objective values, the relative error on energy w.r.t. ground truth objective function value, along with both external iterations and running times in seconds. BCDA serial implementation is used, while parallel OPARBCDA, with a number of $C = 15$ workers is selected. In Table 1 p is the number of tiles in the considered grid and ν is the number of overlapping pixels. While all proposed combinations of tile grid and ν can provide similar function values at end, the introduction of overlap among threads can help on reduce both the external iteration number and the computational time.

	F_ϵ	rel.err	ext. it.	time [s]
ground truth solution	$8.693{\cdot}10^{7}$			
BCDA	$8.736{\cdot}10^{7}$	$5.006{\cdot}10^{-3}$		102.4
$p = 8 \times 8$				
OPARBCDA $\nu = 0$	$8.732{\cdot}10^{7}$	$4.511{\cdot}10^{-3}$	7	31.9 (C=15)
OPARBCDA $\nu = 4$	$8.709{\cdot}10^{7}$	$1.929{\cdot}10^{-3}$	2	25.8 (C=15)
OPARBCDA $\nu = 8$	$8.708{\cdot}10^{7}$	$1.760{\cdot}10^{-3}$	2	27.9 (C=15)
$p = 12 \times 12$				
OPARBCDA $\nu = 0$	$8.736{\cdot}10^{7}$	$4.973{\cdot}10^{-3}$	22	32.7 (C=15)
OPARBCDA $\nu = 4$	$8.729{\cdot}10^{7}$	$4.145{\cdot}10^{-3}$	2	17.0 (C=15)
OPARBCDA $\nu = 8$	$8.721{\cdot}10^{7}$	$3.289{\cdot}10^{-3}$	2	19.2 (C=15)
$p = 16 \times 16$				
OPARBCDA $\nu = 0$	$8.735{\cdot}10^{7}$	$4.858{\cdot}10^{-3}$	74	52.5 (C=15)
OPARBCDA $\nu = 4$	$8.710{\cdot}10^{7}$	$1.957{\cdot}10^{-3}$	3	15.1 (C=15)
OPARBCDA $\nu = 8$	$8.710{\cdot}10^{7}$	$1.955{\cdot}10^{-3}$	3	17.4 (C=15)

Table 1: Trento test problem: comparison of objective function values and computational times obtained by BCDA and OPARBCDA.

It is interesting to note the effect of the choice of the number of overlapping pixels ν to the localization of differences on blocks of variables. To this aim we compare $(\mathbf{s}^p, \mathbf{z}^p, \mathbf{u}^p)$ OPARBCDA solutions to the ground thruth $(\mathbf{s}^*, \mathbf{z}^*, \mathbf{u}^*)$

images: in Figure 3, we show the central portions of the differences $|\mathbf{s}^p - \mathbf{s}^*|$, $|\mathbf{z}^p - \mathbf{z}^*|$ obtained by OPARBCDA with a $t = 8 \times 8$ tile grid and $\nu = 0, 4, 8$ overlap pixels. When no overlap is selected, differences are clustered around tile junctions.

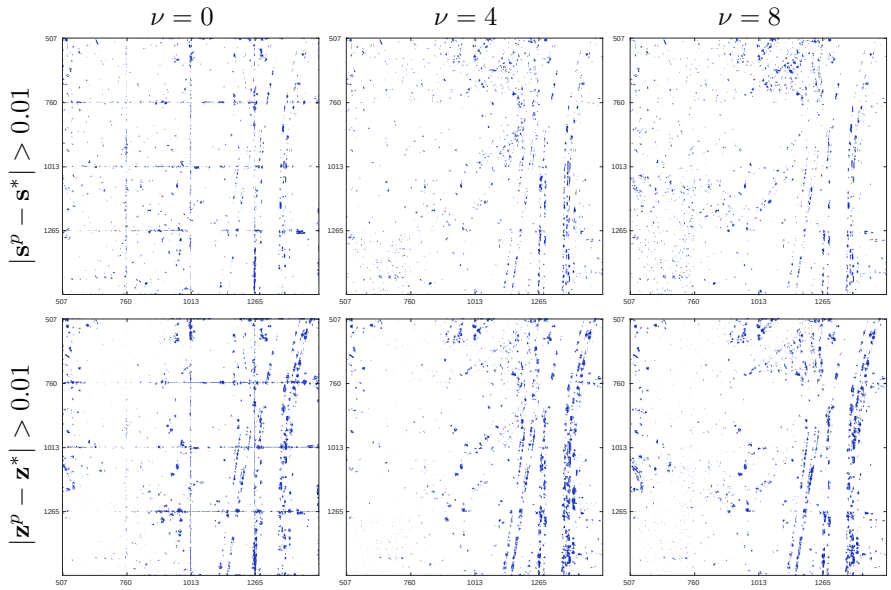

Figure 3: Trento test problem: for visualization purposes, only the central 1010×1010 portion of image is considered; parallel approach tile cuts are located at rows/columns [507 760 1013 1265]. Subfigures on first row show the entries of $|\mathbf{s}^p - \mathbf{s}^*| > 0.01$, while second row shows the entries of $|\mathbf{z}^p - \mathbf{z}^*| > 0.01$. For each row, right figure is obtained with no overlap pixels ($\nu = 0$), while second and third exploit $\nu = 4$ and $\nu = 8$, respectively.

For each algorithm, we then report mean and standard deviation of difference maps between ground truth and achieved solutions and the percentage of entries of $|\mathbf{u}^* - \mathbf{u}|$ that are less than $\sigma_1 = 0.03 * g_{max}$ and $\sigma_2 = 0.01 * g_{max}$, where g_{max} is the maximum value of input image; thresholds for $|\mathbf{s}^* - \mathbf{s}|$ and $|\mathbf{z}^* - \mathbf{z}|$ are $\sigma_1 = 0.03$ and $\sigma_2 = 0.01$. We recall that $\mathbf{s}$ and $\mathbf{z}$ assume values in $[0, 1]$. In order to assess the impact of tile grid position on the reconstruction errors, this analysis is carried on both in the whole domain Ω and in strip regions Υ of wideness 20 pixels centered on tile boundaries. As expected, serial BCDA performance summarized on Table 2 shows no relevant difference between the measures performed on different sets of pixels (Ω and Υ); thus these values can be taken as a reference.

We first consider a fixed tile grid (8×8) and different values of the parameter ν that measures the number of overlapping pixels on neighbouring tiles. From Table 3, we can observe that a value of $\nu = 4$ is sufficient for obtaining a OPARBCDA solution comparable to serial approach, while the choice of $\nu = 0$ locates the error around tile junctions, as already underlined by a visual

	Ω				Υ	
	mean	std. dev.	$< \sigma_1$	$< \sigma_2$	$< \sigma_1$	$< \sigma_2$
BCDA $F_\epsilon(\mathbf{x}^s)$=8.736·10^7 rel.err.=5.006·10^{-3}						
$\mathbf{u}^s - \mathbf{u}^*$	-7.27e-05	1.99e-01	99.66%	97.95%	99.68%	97.99%
$\mathbf{s}^s - \mathbf{s}^*$	-4.09e-04	1.47e-02	98.12%	95.56%	98.16%	95.59%
$\mathbf{z}^s - \mathbf{z}^*$	-7.25e-03	8.80e-02	96.75%	94.14%	96.62%	93.91%

Table 2: Trento test problem: error analysis for BCDA with respect to a ground truth $\mathbf{x}^*$: rel.err. denotes the relative error of $F_\epsilon(\mathbf{x}^s)$ with respect to $F_\epsilon(\mathbf{x}^*)$.

inspection of difference maps.

	Ω				Υ	
	mean	std. dev.	$< \sigma_1$	$< \sigma_2$	$< \sigma_1$	$< \sigma_2$
OPARBCDA $p = 8 \times 8$ $\nu = 0$ $F_\epsilon(\mathbf{x}^p)$=8.732·10^7 rel.err.=4.511·10^{-3}						
$\mathbf{u}^p - \mathbf{u}^*$	-1.96·10^{-3}	2.32·10^{-1}	99.79%	98.83%	99.00%	94.77%
$\mathbf{s}^p - \mathbf{s}^*$	-1.05·10^{-4}	1.24·10^{-2}	99.12%	97.90%	96.26%	91.30%
$\mathbf{z}^p - \mathbf{z}^*$	-2.70·10^{-3}	6.13·10^{-2}	98.18%	96.76%	92.14%	87.29%
OPARBCDA $p = 8 \times 8$ $\nu = 4$ $F_\epsilon(\mathbf{x}^p)$=8.709·10^7 rel.err.=1.929·10^{-3}						
$\mathbf{u}^p - \mathbf{u}^*$	-7.82·10^{-5}	1.37·10^{-1}	99.84%	99.06%	99.79%	98.77%
$\mathbf{s}^p - \mathbf{s}^*$	-1.56·10^{-4}	9.96·10^{-3}	99.20%	98.10%	98.97%	97.48%
$\mathbf{z}^p - \mathbf{z}^*$	-2.58·10^{-3}	5.88·10^{-2}	98.45%	97.06%	97.75%	95.74%
OPARBCDA $p = 8 \times 8$ $\nu = 8$ $F_\epsilon(\mathbf{x}^p)$=8.708·10^7 rel.err.=1.760·10^{-3}						
$\mathbf{u}^p - \mathbf{u}^*$	-6.74·10^{-5}	1.36·10^{-1}	99.83%	99.07%	99.84%	99.01%
$\mathbf{s}^p - \mathbf{s}^*$	-1.41·10^{-4}	1.00·10^{-2}	99.18%	98.08%	99.13%	97.92%
$\mathbf{z}^p - \mathbf{z}^*$	-2.47·10^{-3}	5.80·10^{-2}	98.54%	97.22%	98.25%	96.72%

Table 3: Trento test problem: error analysis for OPARBCDA (tile grid 8×8) with respect to a ground truth $\mathbf{x}^*$: rel.err denotes the relative error of $F_\epsilon(\mathbf{x}^p)$ with respect to $F_\epsilon(\mathbf{x}^*)$.

Table 4 shows that three different grid sizes (8×8, 12×12 and 16×16) with fixed $\nu = 4$ provide similar reconstruction, in terms of the previously described error measures.

By limiting the visualization on selected 600×600 portion shown in Figure 2, a visual comparison of the solution can be carried out: Figure 4 collects, on left column, solution $(\mathbf{u}^s, \mathbf{s}^s, \mathbf{z}^s)$ evaluated by BCDA, while on right column, solution $(\mathbf{u}^p, \mathbf{s}^p, \mathbf{z}^p)$ provided by OPARBCDA with a 16×16 tile grid and $\nu = 8$.

For this preliminary test, an acceptable decrease of the computational time has been acquired: when selecting a number of workers C greater or equal 2, parallel computational time of OPARBCDA is lower than the serial BCDA time, as sketched on left panel of Figure 5. Right panel shows the speedup of the parallel implementation for a selection of proposed tile grids, when varying the number of workers C.

5.3. Large–size dataset segmentation

In the second test problem data size is approximately eight times (for each side) than the previous test problem. Thus tile grid is increased of a factor 8, obtaining similar tile sizes.

Since no ground truth image is available, we present only a brief comparison between BCDA and OPARBCDA solutions. Table 5 summarizes the behaviour

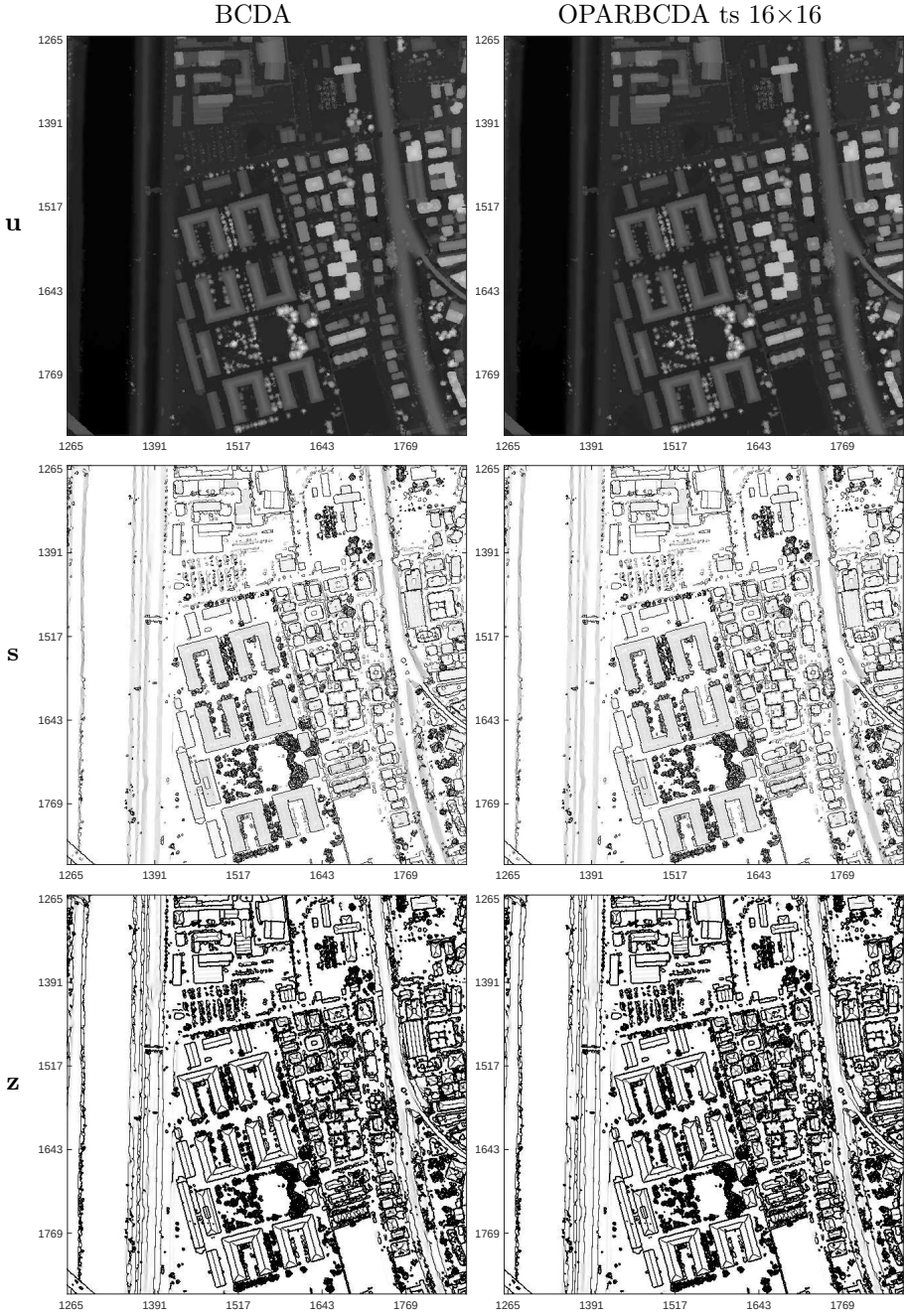

Figure 4: Left column: portion of Trento test image solution $(\mathbf{u}^s, \mathbf{s}^s, \mathbf{z}^s)$ evaluated by BCDA. Right column, solution $(\mathbf{u}^p, \mathbf{s}^p, \mathbf{z}^p)$ provided by OPARBCDA with a 16×16 tile grid and $\nu = 8$.

	Ω				Υ	
	mean	std. dev.	$< \sigma_1$	$< \sigma_2$	$< \sigma_1$	$< \sigma_2$
OPARBCDA $p = 8 \times 8$ $\nu = 4$ $F_\epsilon(\mathbf{x}^p)$=8.709·10^7 rel.err.=1.929·10^{-3}						
$\mathbf{u}^p - \mathbf{u}^*$	-7.82·10^{-5}	1.37·10^{-1}	99.84%	99.06%	99.79%	98.77%
$\mathbf{s}^p - \mathbf{s}^*$	-1.56·10^{-4}	9.96·10^{-3}	99.20%	98.10%	98.97%	97.48%
$\mathbf{z}^p - \mathbf{z}^*$	-2.58·10^{-3}	5.88·10^{-2}	98.45%	97.06%	97.75%	95.74%
OPARBCDA $p = 12 \times 12$ $\nu = 4$ $F_\epsilon(\mathbf{x}^p)$=8.729·10^7 rel.err.=4.145·10^{-3}						
$\mathbf{u}^p - \mathbf{u}^*$	-1.14·10^{-4}	1.79·10^{-1}	99.72%	98.37%	99.71%	98.21%
$\mathbf{s}^p - \mathbf{s}^*$	-2.92·10^{-4}	1.32·10^{-2}	98.50%	96.47%	98.37%	96.03%
$\mathbf{z}^p - \mathbf{z}^*$	-5.53·10^{-3}	7.85·10^{-2}	97.37%	95.26%	96.83%	94.27%
OPARBCDA $p = 16 \times 16$ $\nu = 4$ $F_\epsilon(\mathbf{x}^p)$=8.710·10^7 rel.err.=1.957·10^{-3}						
$\mathbf{u}^p - \mathbf{u}^*$	-8.74·10^{-5}	1.47·10^{-1}	99.81%	98.92%	99.77%	98.68%
$\mathbf{s}^p - \mathbf{s}^*$	-1.63·10^{-4}	1.07·10^{-2}	99.07%	97.78%	98.89%	97.32%
$\mathbf{z}^p - \mathbf{z}^*$	-2.67·10^{-3}	6.24·10^{-2}	98.23%	96.64%	97.77%	95.75%

Table 4: Trento test problem: error analysis for OPARBCDA ($\nu = 4$) with respect to a ground truth $\mathbf{x}^*$: rel.err denotes the relative error of $F_\epsilon(\mathbf{x}^p)$ with respect to $F_\epsilon(\mathbf{x}^*)$.

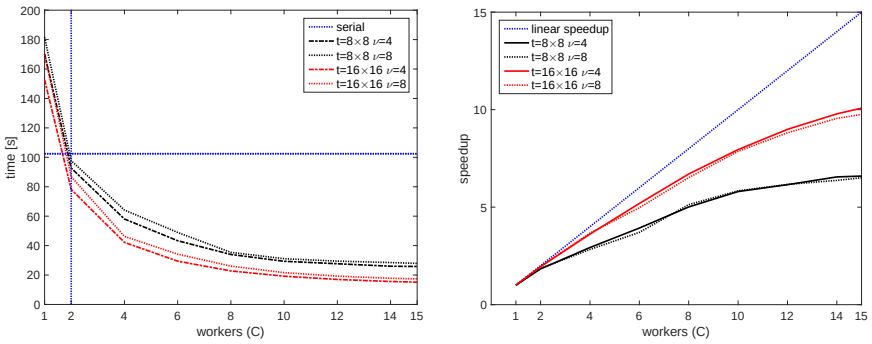

Figure 5: Trento test problem: computational time for parallel OPARBCDA with respect to the number C of workers. Horizontal line at 102.4 s. is serial BCDA time. On the right, speedups when varying both tile grid and number of overlap pixels are sketched.

of previously introduced difference maps, when considering both total pixel domain (Ω) and by focusing our investigation on the proximity of tile junctions (Υ).

	Ω				Υ	
	mean	std. dev.	$< \sigma_1$	$< \sigma_2$	$< \sigma_1$	$< \sigma_2$
OPARBCDA $p = 64 \times 64$ $\nu = 4$						
$\mathbf{u}^p - \mathbf{u}^s$	$-2.97 \cdot 10^{-4}$	$3.94e \cdot 10^{0}$	90.28%	80.92%	87.66%	76.19%
$\mathbf{s}^p - \mathbf{s}^s$	$4.55 \cdot 10^{-4}$	$5.61e \cdot 10^{-2}$	95.49%	90.32%	94.19%	87.55%
$\mathbf{z}^p - \mathbf{z}^s$	$2.57 \cdot 10^{-3}$	$8.13e \cdot 10^{-2}$	97.50%	94.92%	96.63%	93.23%
OPARBCDA $p = 64 \times 64$ $\nu = 8$						
$\mathbf{u}^p - \mathbf{u}^s$	$1.04 \cdot 10^{-5}$	$4.74 \cdot 10^{0}$	87.93%	77.34%	87.43%	76.40%
$\mathbf{s}^p - \mathbf{s}^s$	$7.31 \cdot 10^{-4}$	$6.61 \cdot 10^{-2}$	94.44%	88.48%	94.24%	88.02%
$\mathbf{z}^p - \mathbf{z}^s$	$4.29 \cdot 10^{-3}$	$9.48 \cdot 10^{-2}$	97.01%	94.01%	96.84%	93.69%
OPARBCDA $p = 96 \times 96$ $\nu = 4$						
$\mathbf{u}^p - \mathbf{u}^s$	$-1.46 \cdot 10^{-4}$	$4.26 \cdot 10^{0}$	89.61%	79.81%	87.92%	76.92%
$\mathbf{s}^p - \mathbf{s}^s$	$-3.43 \cdot 10^{-5}$	$5.94 \cdot 10^{-2}$	95.26%	89.89%	94.46%	88.29%
$\mathbf{z}^p - \mathbf{z}^s$	$2.25 \cdot 10^{-3}$	$8.54 \cdot 10^{-2}$	97.40%	94.72%	96.91%	93.79%
OPARBCDA $p = 96 \times 96$ $\nu = 8$						
$\mathbf{u}^p - \mathbf{u}^s$	$-1.15 \cdot 10^{-5}$	$5.18 \cdot 10^{0}$	86.59%	75.24%	86.26%	74.63%
$\mathbf{s}^p - \mathbf{s}^s$	$1.65 \cdot 10^{-4}$	$7.13 \cdot 10^{-2}$	93.87%	87.41%	93.72%	87.09%
$\mathbf{z}^p - \mathbf{z}^s$	$4.06 \cdot 10^{-3}$	$1.02 \cdot 10^{-1}$	96.71%	93.45%	96.58%	93.21%
OPARBCDA $p = 128 \times 128$ $\nu = 4$						
$\mathbf{u}^p - \mathbf{u}^s$	$-1.08 \cdot 10^{-4}$	$4.62 \cdot 10^{0}$	88.12%	77.34%	86.85%	75.30%
$\mathbf{s}^p - \mathbf{s}^s$	$-7.48 \cdot 10^{-4}$	$6.41 \cdot 10^{-2}$	94.59%	88.58%	93.99%	87.42%
$\mathbf{z}^p - \mathbf{z}^s$	$1.47 \cdot 10^{-3}$	$9.20 \cdot 10^{-2}$	97.05%	94.07%	96.71%	93.43%
OPARBCDA $p = 128 \times 128$ $\nu = 8$						
$\mathbf{u}^p - \mathbf{u}^s$	$-6.64 \cdot 10^{-5}$	$5.51 \cdot 10^{0}$	84.57%	71.89%	84.13%	71.17%
$\mathbf{s}^p - \mathbf{s}^s$	$-1.18 \cdot 10^{-3}$	$7.61 \cdot 10^{-2}$	92.91%	85.55%	92.68%	85.13%
$\mathbf{z}^p - \mathbf{z}^s$	$2.11 \cdot 10^{-3}$	$1.09 \cdot 10^{-1}$	96.17%	92.45%	95.99%	92.14%

Table 5: QuickBird test problem: error analysis for OPARBCDA with respect to BCDA solution $\mathbf{x}^s$. Thresholds are $\sigma_1 = 0.03$ and $\sigma_2 = 0.01$ for $|\mathbf{s}^p - \mathbf{s}^s|$ and $|\mathbf{z}^p - \mathbf{z}^s|$, while $\sigma_1 = 3 \cdot 10^{-4} * g_{max}$ and $\sigma_2 = 1 \cdot 10^{-4} * g_{max}$ for $|\mathbf{u}^p - \mathbf{u}^s|$.

Selected thresholds are $\sigma_1 = 0.03$ and $\sigma_2 = 0.01$ for $|\mathbf{s}^s - \mathbf{s}^p|$ and $|\mathbf{z}^s - \mathbf{z}^p|$, while $\sigma_1 = 3 \cdot 10^{-4} * g_{max}$ and $\sigma_2 = 1 \cdot 10^{-4} * g_{max}$ (where g_{max} is the maximum value of input image) for $|\mathbf{u}^s - \mathbf{u}^p|$. Both tile grid size and overlapping pixel number ν marginally impact the difference between solutions, as we can see from visual inspection (see Figure 6) and difference maps on a portion of the solutions (Figure 7).

Obtained objective function values, when varying both the overlap size ν among tiles and the tile grid, are summarized in Table 6; here we show the behaviour of both external iteration number and computational time when varying tile grid and overlap.In accordance with first test results, tile size around 126×124 (obtained with 128×128 tile grid), while sligtly increasing the number of external iterations, provides the lower computational time among tested configuration settings.

A brief comparison between the proposed parallel approaches can be drawn by selecting the best result, in terms of computational time, acquired by BCDA with parallel linear algebra, whose behaviour when varying the number of threads is reported on Table 7. In this case, OPARBCDA (tile grid 128×128, $\nu = 4$)

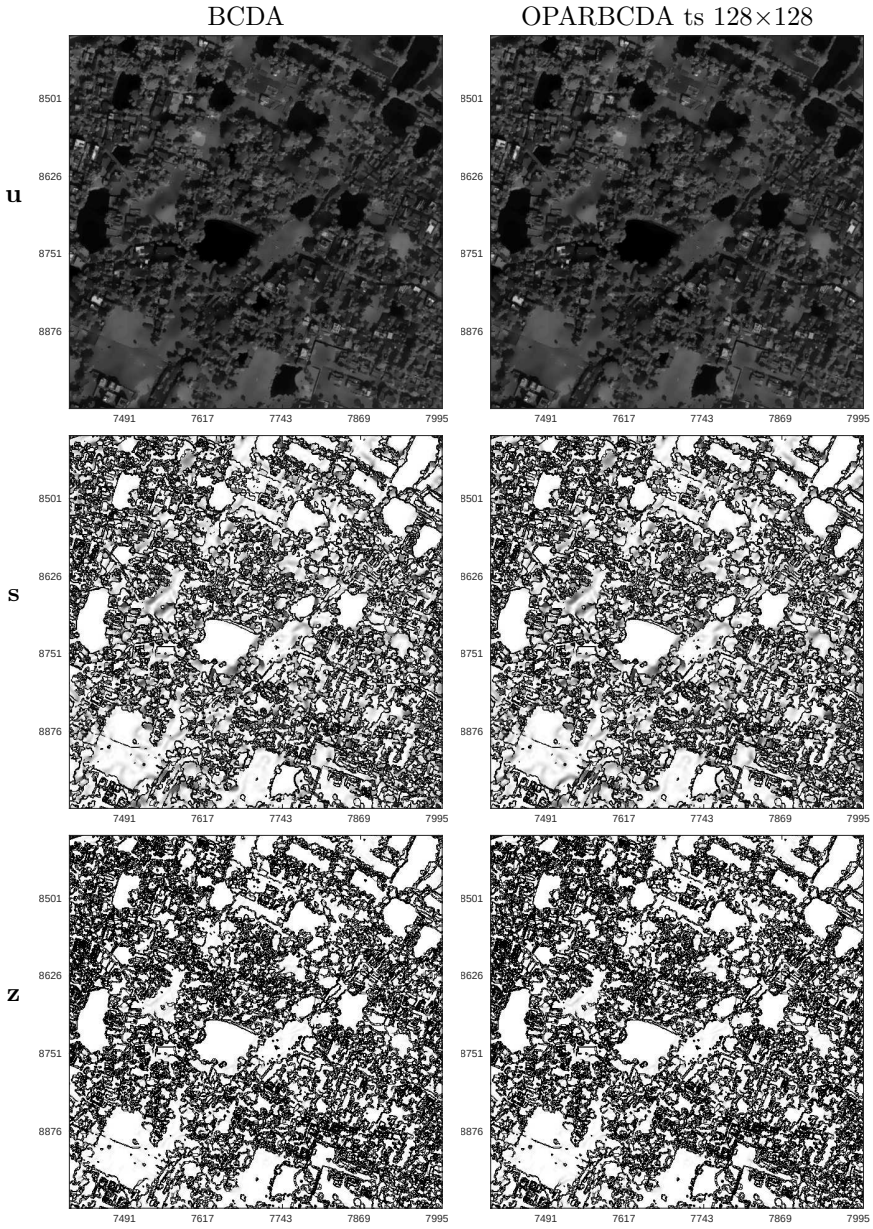

Figure 6: Left column: portion of QuickBird test image solution $(\mathbf{u}^s, \mathbf{s}^s, \mathbf{z}^s)$ evaluated by BCDA. Right column, solution $(\mathbf{u}^p, \mathbf{s}^p, \mathbf{z}^p)$ provided by OPARBCDA with a 128×128 tile grid and $\nu = 8$.

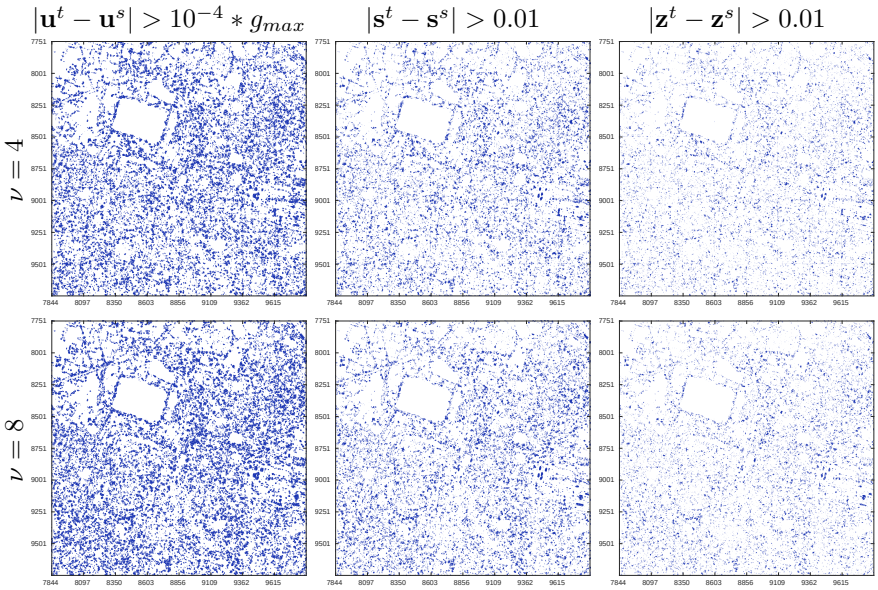

Figure 7: QuickBird test problem: difference maps for OPARBCDA with tile grid 64×64: first row collects the points where absolute difference between OPARBCDA $(\mathbf{u}^p, \mathbf{s}^p, \mathbf{z}^p)$ solution and BCDA $(\mathbf{u}^s, \mathbf{s}^s, \mathbf{z}^s)$ solution is above a fixed threshold, $10^{-4} * g_{max}$ for $\mathbf{u}$, while 10^{-2} for $\mathbf{s}$ and $\mathbf{z}$.

	F_ϵ	ext. it.	time [s]
BCDA (4 threads)	$5.221{\cdot}10^{10}$		12959 (T=4)
$p = 64 \times 64$			
OPARBCDA $\nu = 4$	$5.210{\cdot}10^{10}$	3	1782 (C=15)
OPARBCDA $\nu = 8$	$5.200{\cdot}10^{10}$	3	2639 (C=15)
$p = 96 \times 96$			
OPARBCDA $\nu = 4$	$5.211{\cdot}10^{10}$	4	1464 (C=15)
OPARBCDA $\nu = 8$	$5.200{\cdot}10^{10}$	4	2089 (C=15)
$p = 128 \times 128$			
OPARBCDA $\nu = 4$	$5.215{\cdot}10^{10}$	5	1062 (C=15)
OPARBCDA $\nu = 8$	$5.214{\cdot}10^{10}$	4	1329 (C=15)

Table 6: QuickBird test problem: comparison of objective function values and computational times obtained by the two approaches when tested on QuickBird image.

outcomes BCDA by a factor of 12, lowering the computational time from 3 hours and 35 minutes to around 17 minutes.

test	threads			
	1	4	8	16
QuickBird	14400*	12959	12944	13342

Table 7: QuickBird test problem: computational time in seconds for parallel BCDA with respect to the number of threads. A wall clock time of 4 hours is chosen: run times above this threshod are denoted with $*$.

6. Conclusions

In this paper we addressed the minimization of a second order elliptic approximation of the Blake-Zissermann functional, which is a state-of-the-art variational model to approach the image segmentation problem. Starting from the recently proposed BCDA method [18], this paper developed novel versions of BCDA able to both improve its original performance in terms of computational time and take into account large-size data. In particular we proposed to combine a decomposition technique of the image domain into tiles with a construction of the approximated solution as a proper connection of the sub-solutions computed by BCDA on each tile. The optimization problems related to any tile block of variables can be solved by a parallel strategy. We proved that the parallel (OPARBCDA) method, based on the decomposition of the image domain in tiles, generates a sequence of iterates which wholly converges to a critical point of the functional on the level set devised by the starting point. Commodity multicore CPU has been used in order to evaluate the efficiency of the parallel scheme on large images in terms of computational cost and effectiveness with respect to the behavior on the tile junctions. The numerical results showed the benefits which can be gained by applying the OPARBCDA scheme instead of the BCDA algorithm, not only in terms of computational time saved but also for the possibility of addressing segmentation problems with large-size input images.

Acknowledgements

This work was partially supported by INDAM-GNCS2016 and by MIUR under the project FIRB - Futuro in Ricerca 2012 contract RBFR12M3AC.

References

References

[1] M. Zanetti, V. Ruggiero, M. Miranda Jr., Numerical minimization of a second-order functional for image segmentation, Commun. Nonlinear Sci. Numer. Simul. 36 (2016) 528–548.

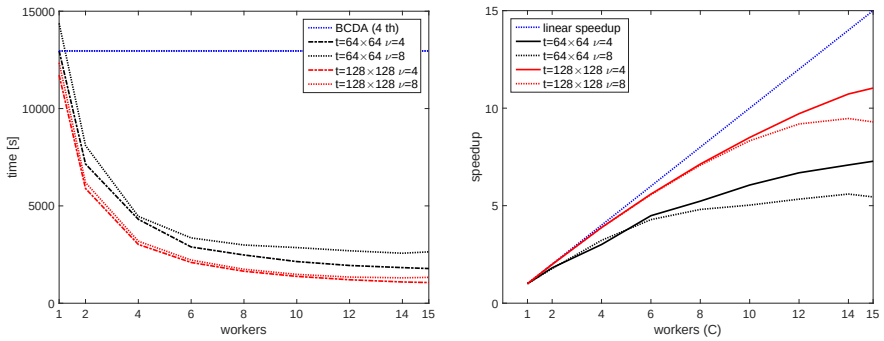

Figure 8: QuickBird test problem: computational time for parallel OPARBCDA with respect to the number C of workers. Horizontal line at 12959 s. is reference BCDA time. On the right: speedups when varying both tile grid and number of overlap pixels.

[2] D. Mumford, J. Shah, Optimal approximations by piecewise smooth functions and associated variational problems, Comm. Pure Appl. Math. 42 (5) (1989) 577–685. doi:10.1002/cpa.3160420503.

[3] A. Blake, A. Zisserman, Visual reconstruction, MIT Press Series in Artificial Intelligence, MIT Press, Cambridge, MA, 1987.

[4] M. Carriero, A. Leaci, F. Tomarelli, A second order model in image segmentation: Blake & Zisserman functional, in: Variational methods for discontinuous structures (Como, 1994), Vol. 25 of Progr. Nonlinear Differential Equations Appl., Birkhäuser, Basel, 1996, pp. 57–72.

[5] M. Carriero, A. Leaci, F. Tomarelli, A survey on the Blake–Zisserman functional, Milan Journal of Mathematics 83 (2) (2015) 397–420.

[6] L. Ambrosio, V. M. Tortorelli, Approximation of functionals depending on jumps by elliptic functionals via Γ-convergence, Comm. Pure Appl. Math. 43 (8) (1990) 999–1036. doi:10.1002/cpa.3160430805.

[7] P. D'Ambra, G. Tartaglione, Solution of Ambrosio–Tortorelli model for image segmentation by generalized relaxation method, Communications in Nonlinear Science and Numerical Simulation 20 (3) (2015) 819–831.

[8] G. Bellettini, A. Coscia, Approximation of a functional depending on jumps and corners, Bollettino Unione Matematica Italiana B (7) 8 (1) (1994) 151–181.

[9] G. Bellettini, A. Coscia, Discrete approximation of a free discontinuity problem, Numerical Functional Analysis and Optimization 15 (3-4) (1994) 201–224.

[10] L. Ambrosio, L. Faina, R. March, Variational approximation of a second order free discontinuity problem in computer vision, SIAM J. Math. Anal. 32 (6) (2001) 1171–1197 (electronic). doi:10.1137/S0036141000368326.

[11] L. Grippo, M. Sciandrone, Globally convergent block-coordinate techniques for unconstrained optimization, Optim. Methods Softw. 10 (4) (1999) 587–637. doi:10.1080/10556789908805730.

[12] M. Zanetti, R. Zanella, L. Bruzzone, A tiling procedure for second-order variational segmentation of large size remote sensing images, in: Proceedings of IEEE International Geoscience and Remote Sensing Symposium, 2016, pp. 6902–6905. doi:10.1109/IGARSS.2016.7730801.

[13] R. Zanella, M. Zanetti, V. Ruggiero, A parallel approach for image segmentation by numerical minimization of a second order functional, in: AIP Conference Proceedings, Vol. 1776, 2016, p. 090035.

[14] D. Noll, A. Rondepierre, Convergence of linesearch and trust-region methods using the Kurdyka–Łojasiewicz inequality, in: Computational and Analytical Mathematics, Vol. 50 of Springer Proceedings in Mathematics & Statistics, Springer, New York, 2013, pp. 593–611.

[15] H. Attouch, J. Bolte, B. F. Svaiter, Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss-Seidel methods, Math. Program., Ser. A 137 (2013) 91–129.

[16] J. Bolte, S. Sabach, M. Teboulle, Proximal alternating linearized minimization for nonconvex and nonsmooth problems, Math. Program., Ser. A 146 (2014) 459–494.

[17] M. Carriero, A. Leaci, F. Tomarelli, Splitting methods with variable metric for Kurdyka–Łojasiewicz functions and general convergence rates, Journal of Optimization Theory and Applications 165 (3) (2015) 874–900.

[18] O. L. Mangasarian, Parallel gradient distribution in unconstrained optimization, SIAM Journal on Optimization 33 (1995) 1916–1925.

[19] J. Hoberock, N. Bell, Thrust: A parallel template library.
URL `https://thrust.github.io`

[20] S. Dalton, N. Bell, L. Olson, M. Garland, Cusp: Generic parallel algorithms for sparse matrix and graph computations.
URL `http://cusplibrary.github.io`

[21] J. L. Ortega-Arjona, G. R. Roberts, Architectural patterns for parallel programming, Proceedings of the 3rd European Conference on Pattern Languages of Programming and Computing (EuroPLoP98), Kloster Irsee, Germany, 1988.

[22] D. R. Butenhof, Programming with POSIX Threads, Addison-Wesley Professional, 1997.

[23] H. Sutter, A. Alexandrescu, CPP Coding Standards: 101 Rules, Guidelines and Best Practices, Addison-Wesley Longman, 2004.

[24] www.territorio.provincia.tn.it/portal/server.pt/community/lidar/847/lidar/23954.

[25] DigitalGlobe (2005), QuickBird scene 000000185940_01_P001, Level Standard 2A, DigitalGlobe, Longmont, Colorado, 11/21/2002.