

# DOPE: Distributed Optimization for Pairwise Energies

Jose Dolz

jose.dolz@livia.etsmtl.ca

Ismail Ben Ayed

ismail.benayed@etsmtl.ca

Christian Desrosiers

Laboratory for Imagery, Vision and Artificial Intelligence  
 Ecole de Technologie Supérieure, Montreal, Canada

christian.desrosiers@etsmtl.ca

## Abstract

We formulate an Alternating Direction Method of Multipliers (ADMM) that systematically distributes the computations of any technique for optimizing pairwise functions, including non-submodular potentials. Such discrete functions are very useful in segmentation and a breadth of other vision problems. Our method decomposes the problem into a large set of small sub-problems, each involving a sub-region of the image domain, which can be solved in parallel. We achieve consistency between the sub-problems through a novel constraint that can be used for a large class of pairwise functions. We give an iterative numerical solution that alternates between solving the sub-problems and updating consistency variables, until convergence. We report comprehensive experiments, which demonstrate the benefit of our general distributed solution in the case of the popular serial algorithm of Boykov and Kolmogorov (BK algorithm) and, also, in the context of non-submodular functions.

## 1. Introduction

A mainstay in computer vision, regularization serves a breadth of applications and problems including segmentation [27, 29], optical flow [18], shape fitting [17], stereo matching [15], deconvolution [10], high-dimensional clustering [28], among many others [3, 13]. For instance, in the discrete setting, segmentation problems are commonly stated as optimizing a regularization-based functional<sup>1</sup> of the following general form [1, 4, 16]:

$$E(\mathbf{y}) = \sum_{i \in \Omega} u_i y_i + \lambda \sum_{i,j \in \Omega^2} w_{ij} |y_i - y_j| \quad (1)$$

where  $\Omega$  is the image domain and  $\mathbf{y} = (y_1, y_2, \dots, y_n)^\top \in \{0, 1\}^{|\Omega|}$  is a binary vector indicating a possible foreground-

<sup>1</sup>We give a binary (two-region) segmentation functional for simplicity but the discussion extends to multi-region segmentation.

background segmentation:  $y_i = 1$  if pixel  $i$  belongs to the foreground class, otherwise  $y_i = 0$ .  $\lambda \geq 0$  controls the relative importance of each term. The first term is a sum of unary potentials typically defined via log posteriors:

$$u_i = \log p(y_i = 0 | \mathbf{x}_i) - \log p(y_i = 1 | \mathbf{x}_i) \quad (2)$$

with  $\mathbf{x}_i \in \mathbb{R}^M$  denoting the feature vector of pixel  $i \in \Omega$  (e.g., color). The second term in (1) is a general form of pairwise regularization. The second-order Potts model [4] is an important example of pairwise regularization, and is very popular in computer vision: given a neighborhood  $\mathcal{N}(i)$  for pixel  $i$ ,  $w_{ij} > 0$  if  $j \in \mathcal{N}(i)$  and 0 elsewhere. In this case,  $w_{ij}$  is a penalty for assigning different labels to neighboring pixels  $i$  and  $j$ . Such a penalty can be either a constant, in which case the regularization term measures the length of segment boundary, or a decreasing function of feature (e.g., color) difference  $\|\mathbf{x}_i - \mathbf{x}_j\|$ , which attracts the segment boundary towards strong feature edges [4]. Potts regularization belongs to an important family of discrete pairwise functions, *submodular functions*<sup>2</sup>, which were instrumental in the development of various efficient computer vision algorithms. The global optimum of a function containing unary and submodular pairwise potentials can be computed exactly in polynomial time using graph cut (or max-flow) algorithms [5]. Other examples of pairwise terms of the general form in (1) include non-submodular functions, which arise in problems such as curvature regularization [10, 22], surface registration [13], deconvolution [10] and inpainting [13]. It also includes dense (fully connected) models [16], where pairwise penalties  $w_{ij}$  are not restricted to neighbouring pixels. These are only few examples of pairwise-function problems widely used in combination with popular optimization techniques such as LP relaxation [13] or mean-field inference [1, 16]. Finally, it is worth mentioning that total-variation (TV) terms can be

<sup>2</sup>A function  $f$  defined over a pair of discrete binary variables is submodular if and only if  $f(1, 0) + f(0, 1) \geq f(1, 1) + f(0, 0)$ .

viewed as the continuous counterpart of Potts regularization, and were the subject of a large number of vision works in recent years [7, 23, 24, 30].

Recently, there have been significant research efforts focusing on designing parallel (or distributed) formulations for optimizing pairwise functions [1, 19, 25]. Distributing computations would be beneficial not only to high-resolution images and massive 3D grids but also to difficult high-order models [11, 14, 26], which require approximate solutions solving a large number of problems of the form (1). For instance, continuous convex relaxation techniques have been gaining popularity recently due to their ability to accommodate parallel implementations [7, 23, 24, 30]. Unfortunately such techniques are restricted to TV regularization terms. Mean-field inference techniques [1, 16] also attracted significant attention recently as they can be parallelized, albeit at the cost of convergence guarantees [1].

In the context of submodular functions, several studies focused specifically on parallel formulations of the max-flow/graph-cut algorithm of Boykov and Kolmogorov (BK algorithm) [5], which has made a substantial impact in computer vision. The BK algorithm yields a state-of-the-art empirical performance for typical vision problems such as 2D segmentation. Even though this augmenting path algorithm is serial, it uses heuristics that handle efficiently sparse 2D grids, outperforming top parallel push-relabel max-flow algorithms [8]. Unfortunately, distributing the computations for the BK algorithm is not a trivial problem<sup>3</sup>, and the efficiency of the algorithm may decrease when moving from 2D to 3D (or higher-dimensional) grids. Therefore, parallel/distributed computations for this algorithm would be of substantial benefit to the community, and several works addressed the problem [2, 19, 25]. For instance, the method in [19] investigated a bottom-up approach to parallelize the BK algorithm using two subsequent phases: the first stage partitions the graph into several sub-graphs and processes them in parallel, whereas the second stage gradually merges the subgraphs so as to involve longer paths, until a global minimum is reached. Unfortunately, this technique requires a shared-memory model, which does not accommodate distributed computations. The method in [25] wrote the max-flow (graph cut) problem as a linear program, and viewed the objective function as a sum of two functions, each involving a sub-graph. Then, they used a dual decomposition formulation to process each of the two sub-graphs independently. However, it is not clear how to split the problem into a large number of sub-graphs (for faster computations) as this would increase exponentially the number of constraints (w.r.t. the sub-graphs). The method in [2] proposed a linear program formulation of the BK algorithm, via a  $L_1$  minimization statement. Solving the problem via Newton it-

<sup>3</sup>Augmenting-path max-flow algorithms are based on global operations and, therefore, do not accommodate parallel/distributed implementations

erations yields matrix-vector multiplications, which can be evaluated in parallel. This method, however, is not significantly faster than the serial BK algorithm [25].

In general, the existing distributed/parallel formulations for optimizing pairwise functions are technique-specific. For instance, the methods in [2, 19, 25] were tailored for the BK algorithm, and it is not clear how to extend these methods beyond the context of max-flow formulations and submodular functions. In this study, we formulate an Alternating Direction Method of Multipliers (ADMM), which systematically distributes the computations of any technique for optimizing pairwise functions, including non-submodular potentials. Our method decomposes the problem into a large set of small sub-problems, each involving a sub-region of the image domain (i.e., block), which can be solved in parallel. We achieve consistency between the sub-problems through a novel constraint that can be used in conjunction with any functional of the form (1). We give an iterative numerical solution that alternates between solving the sub-problems and updating consistency variables, until convergence. Our method can be viewed as a variant of the alternating projections algorithm to find a point in the intersection of two convex sets and, therefore, is well suited to distributed convex optimization. We report comprehensive experiments, which demonstrate the benefit of our general solution in the case of the popular BK algorithm and, also, in the context of non-submodular functions.

## 2. Formulation

Let  $\mathbf{u} = (u_1, u_2, \dots, u_n)^\top \in \mathbb{R}^{|\Omega|}$  be the vector of unary penalties and  $W \in \mathbb{R}^{|\Omega| \times |\Omega|}$  the matrix of pairwise penalties  $w_{ij}$ . It is easy to show that the general segmentation problem in (1) can be expressed in matrix form as follows:

$$\arg \min_{\mathbf{y} \in \{0,1\}^{|\Omega|}} \mathbf{u}^\top \mathbf{y} + \lambda \mathbf{y}^\top L \mathbf{y}. \quad (3)$$

Here,  $L = D - W$  is the Laplacian matrix corresponding to  $W$ , and  $D$  is a diagonal matrix such that  $d_{ii} = \sum_j w_{ij}$ .

Let us divide a large image  $\Omega$  into  $K$  blocks  $\Omega_k$  ( $k = 1, \dots, K$ ) that can overlap, which allows pixels of image  $\Omega$  to be simultaneously located in multiple blocks. Let  $\hat{\mathbf{y}}_k \in \{0,1\}^{|\Omega_k|}$  denote the segmentation vector of block  $k$ . Our goal is to reformulate problem (3) in a way that the tasks of segmenting blocks are not directly coupled, thus allowing them to be performed simultaneously. To achieve this, we connect them through the segmentation vector  $\mathbf{y} \in \{0,1\}^{|\Omega|}$  of the whole image  $\Omega$ , by imposing linear constraints  $\hat{\mathbf{y}}_k = S_k \mathbf{y}$ ,  $k = 1, \dots, K$ , where  $S_k$  is a  $|\Omega_k| \times |\Omega|$  matrix selecting the pixels of block  $k$ .

Given the segmentation vectors of each block, global segmentation  $\mathbf{y}$  can be expressed using the following proposition.

**Proposition 1.** If each pixel of  $\Omega$  belongs to at least one block, i.e.  $\cup_{k=1}^K \Omega_k = \Omega$ , and  $\hat{\mathbf{y}}_k = S_k \mathbf{y}$ ,  $k = 1, \dots, K$ , then the following relationship holds:

$$\mathbf{y} = \left( \sum_{k=1}^K S_k^\top S_k \right)^{-1} \left( \sum_{k=1}^K S_k^\top \hat{\mathbf{y}}_k \right). \quad (4)$$

Here,  $Q = \sum_k S_k^\top S_k$  is a diagonal matrix such that  $q_{ii}$  is the number of blocks containing pixel  $i$ . In short, this proposition states that, if the block segmentation vectors are consistent (i.e., pixels have the same label across blocks containing them), then  $y_i$  is simply the mean label of pixel  $i$  within the blocks containing this pixel. This property also applies when relaxing the integer constraints on  $\mathbf{y}$ , allowing us to develop an efficient optimization strategy.

In the following theorem, we show that segmentation problem (3) can be reformulated as a sum of similar sub-problems, one for each block  $k$ , connected together through a relaxed global segmentation vector  $\mathbf{y}$ .

**Theorem 1.** Denote as  $\hat{\mathbf{u}}_k = S_k Q^{-1} \mathbf{u}$  and  $\hat{W}_k = S_k Q^{-1} W Q^{-1} S_k^\top$  the unary and binary potential weights of block  $k$ , adjusted to consider the occurrence of pixels in multiple blocks. Moreover, let  $\hat{D}_k$  be the diagonal matrix such that  $[\hat{D}_k]_{ii} = \sum_j [\hat{W}_k]_{ij}$ , and  $\hat{L}_k = \hat{D}_k - \hat{W}_k$  be the Laplacian of  $\hat{W}_k$ . If  $\hat{\mathbf{y}}_k = S_k \mathbf{y}$ ,  $k = 1, \dots, K$ , then problem (3) can be reformulated as

$$\arg \min_{\substack{\mathbf{y} \in \mathbb{R}^{|\Omega|} \\ \hat{\mathbf{y}}_k \in \{0,1\}^{|\Omega_k|}}} \sum_{k=1}^K (\hat{\mathbf{u}}_k + \lambda C_k \mathbf{y})^\top \hat{\mathbf{y}}_k + \lambda \sum_{k=1}^K \hat{\mathbf{y}}_k^\top \hat{L}_k \hat{\mathbf{y}}_k, \quad (5)$$

where  $C_k = S_k Q^{-1} L - \hat{L}_k S_k$ .

*Proof.* See details in Appendix A.  $\blacksquare$

We solve problem (5) with an ADMM approach. Moving constraints  $\hat{\mathbf{y}}_k = S_k \mathbf{y}$ ,  $k = 1, \dots, K$ , into the functional via augmented Lagrangian terms [12] (with multiplier  $\mathbf{a}_k$ ) gives:

$$\begin{aligned} & \arg \min_{\substack{\mathbf{y} \in \mathbb{R}^{|\Omega|}, \hat{\mathbf{y}}_k \in \{0,1\}^{|\Omega_k|} \\ \mathbf{a}_k \in \mathbb{R}^{|\Omega_k|}}} \sum_{k=1}^K (\hat{\mathbf{u}}_k + \lambda C_k \mathbf{y})^\top \hat{\mathbf{y}}_k \\ & + \lambda \sum_{k=1}^K \hat{\mathbf{y}}_k^\top \hat{L}_k \hat{\mathbf{y}}_k + \frac{\mu}{2} \sum_{k=1}^K \|\hat{\mathbf{y}}_k - S_k \mathbf{y} + \mathbf{a}_k\|_2^2. \end{aligned} \quad (6)$$

In this equation, augmented Lagrangian parameter  $\mu \geq 0$  controls the trade-off between the original functional and satisfying the constraints. In general, ADMM methods are not overly sensitive to this parameter and converge if  $\mu$  is large enough [6]. In practice,  $\mu$  is initialized using a small value and increased at each iteration by a given factor. To solve problem (6), we note that the functional is convex with

respect to each parameter  $\hat{\mathbf{y}}_k$ ,  $\mathbf{y}$  and  $\mathbf{a}_k$ . We thus update these parameters alternatively, until convergence is reached (i.e., the constraints are satisfied up to a given  $\epsilon$ ).

Given  $\mathbf{y}$ , the segmentation vectors of each block  $k$  can be updated independently, in parallel, by solving the following problem:

$$\begin{aligned} & \arg \min_{\substack{\hat{\mathbf{y}}_k \in \{0,1\}^{|\Omega_k|}}} (\hat{\mathbf{u}}_k + \lambda C_k \mathbf{y})^\top \hat{\mathbf{y}}_k + \lambda \hat{\mathbf{y}}_k^\top \hat{L}_k \hat{\mathbf{y}}_k \\ & + \frac{\mu}{2} \|\hat{\mathbf{y}}_k - (S_k \mathbf{y} - \mathbf{a}_k)\|_2^2 \end{aligned} \quad (7)$$

Using the fact that  $\|\hat{\mathbf{y}}_k\|_2^2 = \mathbf{1}^\top \hat{\mathbf{y}}_k$  for binary vector  $\hat{\mathbf{y}}_k$ , we reformulate the problem as:

$$\begin{aligned} & \arg \min_{\substack{\hat{\mathbf{y}}_k \in \{0,1\}^{|\Omega_k|}}} \left( \hat{\mathbf{u}}_k + \lambda C_k \mathbf{y} + \mu (\mathbf{a}_k - S_k \mathbf{y} + \frac{1}{2}) \right)^\top \hat{\mathbf{y}}_k \\ & + \lambda \hat{\mathbf{y}}_k^\top \hat{L}_k \hat{\mathbf{y}}_k. \end{aligned} \quad (8)$$

Notice that for  $\mathbf{y}$  fixed, this block problem corresponds to a sum of unary and pairwise potentials. Therefore, as discussed in the introduction, it can be solved with one of the popular techniques<sup>4</sup>, e.g., the BK algorithm [5]

Once all block segmentation vectors have been computed, we can update the global segmentation  $\mathbf{y}$  by solving the following problem:

$$\arg \min_{\mathbf{y} \in \mathbb{R}^{|\Omega|}} \lambda \sum_{k=1}^K \hat{\mathbf{y}}_k^\top C_k \mathbf{y} + \frac{\mu}{2} \sum_{k=1}^K \|S_k \mathbf{y} - (\hat{\mathbf{y}}_k + \mathbf{a}_k)\|_2^2. \quad (9)$$

Since we have relaxed the integer constraints on  $\mathbf{y}$ , this corresponds to a unconstrained least-square problem, whose solution is given by:

$$\mathbf{y} = \frac{1}{\mu} Q^{-1} \sum_{k=1}^K \left( \mu S_k^\top (\hat{\mathbf{y}}_k + \mathbf{a}_k) - \lambda C_k^\top \hat{\mathbf{y}}_k \right). \quad (10)$$

Note that since  $Q$  is diagonal, computing its inverse is trivial.

Finally, the Lagrangian multipliers are updated as in standard ADMM methods:

$$\mathbf{a}_k = \mathbf{a}_k + (\hat{\mathbf{y}}_k - S_k \mathbf{y}), \quad k = 1, \dots, K. \quad (11)$$

The pseudo-code for implementing our DOPE method is given in Algorithm 1. In a first step, the algorithm computes the unary and pairwise potentials of the global image, and divides the image into possibly overlapping blocks  $\Omega_k$ ,  $k = 1, \dots, K$ , based on a given partition scheme. For each block  $k$ , the algorithm pre-computes parameters  $S_k$ ,  $C_k$ ,  $\hat{W}_k$  and  $\hat{\mathbf{u}}_k$ . Note that these parameters can be computed in parallel. In the main loop, the algorithm then simultaneously recomputes segmentation vectors  $\hat{\mathbf{y}}_k$  of each block, and uses them to update the global segmentation vector  $\mathbf{y}$ . This process is repeated until constraints linking the block

<sup>4</sup>The choice depends on the form of the matrix of pairwise potentials.

segmentation to the global segmentation are satisfied, up to a given  $\epsilon \geq 0$ . As mentioned earlier, the algorithm's convergence is facilitated by increasing the ADMM parameter  $\mu$  by a factor of  $\mu_{\text{fact}}$  at each iteration.

---

**Algorithm 1:** DOPE segmentation algorithm

---

**Input:** The input image  $\Omega$  and pixel features  $\mathbf{x}$ ;  
**Input:** Block partition scheme;  
**Input:** Parameters  $\lambda, \epsilon, \mu_0, \mu_{\text{fact}}$ ;  
**Output:** The segmentation vector  $\mathbf{y}$ ;

*Initialization:*  
 Compute global image unary and pairwise potentials;  
 Compute blocks and their corresponding parameters;  
 Set  $y_i := \frac{1}{2}, i = 1, \dots, |\Omega|$ ;  
 Set  $\mathbf{a}_k := \mathbf{0}, k = 1, \dots, K$ ;  
 Set  $\mu := \mu_0$ ;

*Main loop:*  
**while**  $\exists k, s.t. \|\hat{\mathbf{y}}_k - S_k \mathbf{y}\|_2 > \epsilon$  **do**  
 | **In parallel:** update  $\hat{\mathbf{y}}_k, k = 1, \dots, K$ , by solving (8);  
 | Update  $\mathbf{y}$  by Eq. (10);  
 | Update  $\mathbf{a}_k, k = 1, \dots, K$ , based on Eq. (11);  
 | Set  $\mu := \mu \times \mu_{\text{fact}}$ ;  
**return**  $\mathbf{y}$ .

---

### 3. Experiments

The main goal of our experiments is to demonstrate that our DOPE formulation can distribute the computations of powerful serial algorithms without affecting the quality of the energies at convergence. First, we prove that our formulation can achieve segmentation results consistent with the popular serial graph cut (sGC) algorithm of Boykov-Kolmogorov [5], while allowing distributed computations. We illustrate the usefulness of our method on the task of segmenting high-resolution 2D multi-channel images (Section 3.1) and 3D MRI brain volumes (Section 3.2) using the second-order Potts model, and compare its accuracy, obtained energies and efficiency to those corresponding to sGC. The consistency between our method's segmentation and sGC is measured using Dice score coefficient (DSC) and relative energy differences:

$$\Delta E(\%) = \frac{E_{GC} - E_{distReg}}{E_{GC}} \times 100 \quad (12)$$

where  $E_{GC}$  is the energy of serial GC and  $E_{distReg}$  is the energy given by our distributed regularization formulation.

Another objective of these experiments is to assess the impact of our method's parameters on computation time and segmentation accuracy. In particular, we evaluated how the partitioning scheme (i.e., block size and overlap) affects the method's performance. If blocks are small, a greater level of parallelism can be achieved, but segmentation consistency across blocks might be harder to satisfy. Con-

versely, using larger blocks with more overlap encourages global consistency of the segmentation, but might increase the run times. In our experiments, we considered three partitioning schemes, dividing images into  $K = 32, 64$  or  $128$  even-sized blocks. For each of these, we tested three levels of overlap. In the first one, denoted by  $\text{Size}_{00}$ , images were split into  $K$  non-overlapping blocks covering the whole image (i.e., each pixel/voxel is in exactly one block). The size of these blocks was then increased by 10% and 25%, leading to larger blocks with greater overlap. We denote these two overlapping partitions by  $\text{Size}_{10}$  and  $\text{Size}_{25}$ , respectively. Furthermore, we investigated the impact of neighbourhood size (i.e., the number of non-zero pairwise potentials  $w_{ij}$ ) on segmentation performance. Using larger neighbourhoods, as defined by the kernel, can lead to a finer segmentation but significantly increases run times. Kernel sizes of 3, 5, 7 and 9 pixels/voxels were considered in our experiments. We used square kernels for 2D images, and spherical kernels in the 3D setting. The regularization parameter  $\lambda$  was selected per image (typically, its values are proportional to image size). Note that the same  $\lambda$  was used for computing the energy of both our method and sGC. Finally, the ADMM parameter was initialized to  $\mu_0 = 100$  for 2D images and  $\mu_0 = 500$  for 3D volumes, and increased by a factor of  $\mu_{\text{fact}} = 1.05$  at each iteration.

Finally, we report curvature regularization experiments to show the use of our formulation in the case of non-submodular pairwise functions (Section 3.3). These experiments involve distributing the computations of the trust region (LSA-TR) method in [10], a serial non-submodular optimizer that recently obtained competitive<sup>5</sup> performances in a wide range of applications (deconvolution, inpainting, among others). This shows that our formulation can be readily used in these applications.

Our method was implemented in MATLAB R2015b, and all experiments were performed on a server with the following hardware specifications: 64 Intel(R) Xeon(R) 2.30GHz CPUs with 8 cores, and 128 GB RAM. For sGC, we used the publicly available B-K MATLAB tool<sup>6</sup>, which implements the max-flow algorithm. In the next sections, we present the results obtained for high resolution 2D multi-channel images, 3D MRI data and a squared curvature regularization example.

#### 3.1. High-resolution 2D multi-channel images

We first tested our method on 10 high-resolution 2D multi-channel images, with resolution ranging between  $2000 \times 3000$  and  $2600 \times 3900$  pixels. As in [4], we drew seeds to generate color model priors for the foreground and

<sup>5</sup>LSA-TR outperforms significantly popular non-submodular optimization techniques such as TRWS and QPBO; See the comparative energy plots in [10].

<sup>6</sup><http://vision.csd.uwo.ca/code/>

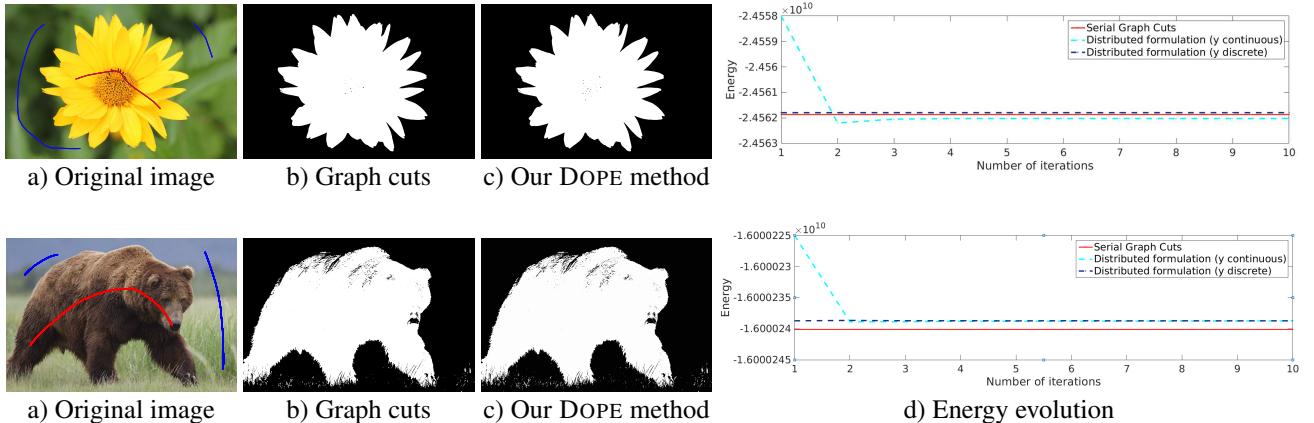


Figure 1: Visual example of 2D image segmentation by sGC and DOPE approaches (*left*) and the evolution of the segmentation energy in both formulations with respect to the number of iterations (*right*).

background regions (see Fig. 1). The k-means algorithm [21] was employed to group foreground/background seed pixels into 5 clusters, which were then used to compute the log posteriors of unseeded pixels.

Figure 2 gives the average relative energy difference (%), relative time difference and Dice similarity coefficient (DSC) between the segmentation of our DOPE method and sGC, computed over the 10 images. In the left-side plots, the number of blocks was set to 128, but we varied the size of these blocks and the kernel. Conversely, the right-side plots compare our method with sGC for various numbers of blocks and kernel sizes, while keeping the block size fixed to  $\text{Size}_{25}$ . Values are reported for one and three ADMM iterations of our method. While a more detailed analysis is presented below, we found that three iterations were often sufficient to achieve convergence in the case of 2D images (e.g., see Fig. 1). We observe that energy differences are quite small in all tested configurations, with values around 0.1% for one ADMM iteration and 0.01% for three ADMM iterations. With respect to block size, we observe no difference between the tested configurations, for the same number of ADMM iterations. For a single ADMM iteration, increasing the overlap seems to result in higher energy differences. However, these differences disappear when using three iterations, suggesting that having a greater overlap requires more iterations to converge.

As expected, segmentation times varied proportionally to the number and size of blocks. However, doubling the number of blocks did not lead to reduction in processing time by the same factor. This is in part due to pre-processing operations, such as computing the unary and pairwise potentials for the whole image, which need to be performed regardless of the image partitioning scheme used. Another trend that can be observed is that the speed-up provided by our method increases with the kernel size. Thus, for kernel sizes of 7 or

more, our method obtained nearly identical segmentation results up to 5 times faster than sGC. Additionally, allowing the algorithm to run three iterations did not increase run times significantly for larger kernels, suggesting that most operations are performed in pre-processing steps.

In terms of segmentation consistency, it can be seen that our DOPE method obtains segmentation results quite similar to those of sGC, with DSC values above 0.99. In most cases, increasing the number of blocks decreases DSC values, although this difference is not significant. A similar effect can be observed when employing larger blocks and kernels. Overall, the segmentation results obtained by our method are consistent with those of sGC, for all tested configurations.

Figure 1 gives two examples of segmentations obtained by sGC and our DOPE method. The first column shows the image to be segmented with foreground/background scribbles, whereas the second and third columns give the segmentation result of sGC and our method, respectively. The evolution of the segmentation energy is also shown in Figure 1 (*right*). We observe that our approach converges rapidly, requiring only two iterations to achieve near-zero energy differences.

### 3.2. 3D MRI volumes

Segmentation efficiency is particularly important in the case of 3D volumes, where computational and memory requirements often exceed the capacity of current methods. As a second experiment, we tested our DOPE method on a 3D MRI brain volume of size  $200 \times 200 \times 100$ . For this experiment, we considered the task of segmenting sub-cortical brain regions, and used the soft probability map generated by a 3D convolutional neural network [19]<sup>7</sup> as unary potentials in the energy function.

<sup>7</sup><https://github.com/josedolz/LiviaNET>

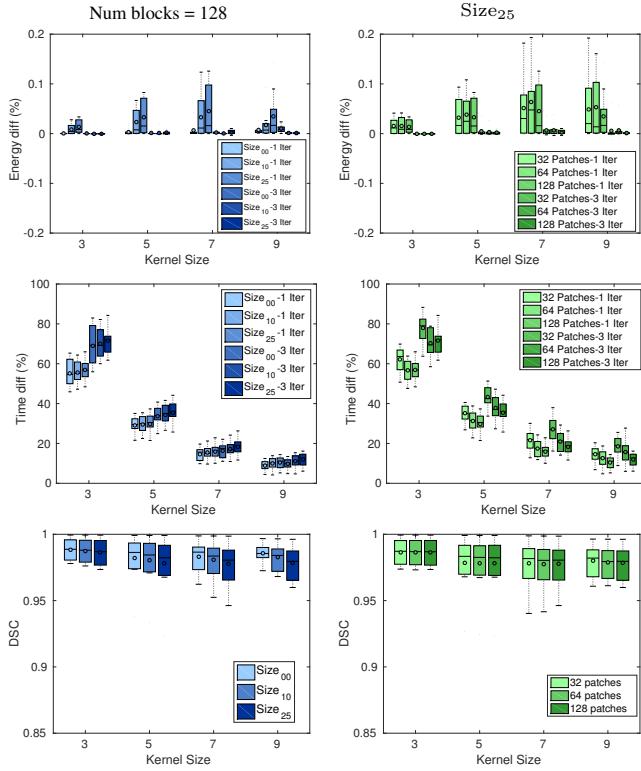


Figure 2: Mean relative energy difference (%), relative time difference (%) and Dice similarity coefficient (DSC) between the segmentation of sGC and our DOPE method, computed over the 10 high-definition 2D images. In the left-side plots, the number of blocks is fixed to 128, and values are reported for different block and kernel sizes. In right-side plots, the block size is fixed to Size<sub>25</sub>, and values are reported for different number of blocks and kernel sizes.

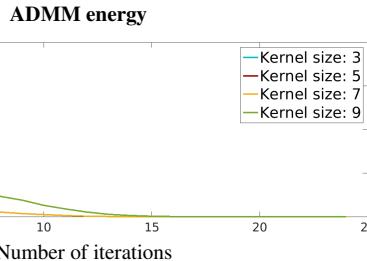


Figure 3: Evolution of the ADMM energy (augmented Lagrangian terms) for a partitioning comprised of 128 blocks with Size<sub>10</sub> and different kernel sizes.

Figure 3 shows the energy related to the augmented Lagrangian terms in Eq. (6), for a partitioning composed of 128 blocks with Size<sub>10</sub>, and kernel sizes corresponding to 3, 5, 7 and 9. Recall that this energy corresponds to segmentation consistency across different blocks. We observe that the number of iterations required to achieve convergence increases with kernel size, probably due to the

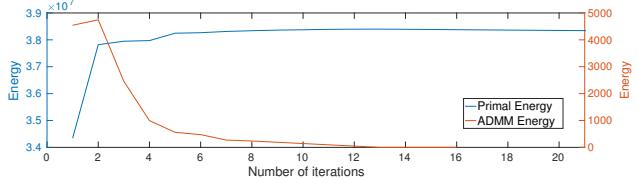


Figure 4: Evolution of the segmentation (primal) energy and ADMM (augmented Lagrangian terms) energy during optimization, for a partitioning comprised of 128 blocks with Size<sub>10</sub> and a kernel of size 7.

broader interaction between blocks for larger kernels. However, our method converged in less than 10 iterations, for all kernel sizes. These observations are confirmed by Figure 4, which also shows the variation of segmentation energy (unary and pairwise potentials) when employing 128 blocks with Size<sub>10</sub> and a kernel of size 7. We notice that the segmentation energy increases with the number of iterations. This can be explained by the fact that this energy is computed using the integer-relaxed segmentation vector  $\mathbf{y}$ , which gets increasingly restricted to a binary solution over time. Segmentation convergence is illustrated in Fig. 5, which shows the evolution of  $\mathbf{y}$  for a random 2D slice of the volume.

Analyzing detailed results, we see that mean relative energy differences increase with kernel size, ranging from 0.1% for kernels of size 3 to 2.5% when employing kernel of size 9. Moreover, for a fixed kernel size, having a greater overlap leads to smaller energy differences (e.g., energy difference of 1.5% for Size<sub>20</sub> compared to 2.25% for Size<sub>00</sub>, for a kernel size of 9 and 128 blocks). This suggests the greater usefulness of having overlapping blocks in the segmentation of 3D volumes. However, when overlap is allowed, larger block sizes reported slightly higher energy differences. As was the case for 2D image segmentation, the speed-up of our DOPE method depends on kernel size and the block number/size. Hence, for 64 blocks with Size<sub>10</sub> and kernel size of 7, our method was about 3 times faster than sGC, while a speed-up of 4 was achieved using 128 blocks with Size<sub>00</sub> and a kernel size of 9. For segmentation consistency, DSC values ranged from 0.95 to 0.99 in all tested configurations. While a few iterations were sufficient for small kernels, larger kernels required more iterations to converge.

Figure 6 shows the segmentation of white matter tissues in three axial slices, obtained by our DOPE method using 128 blocks with Size<sub>10</sub> and kernel size of 7 (*left column*) and sGC (*middle column*). Segmentation differences are shown in the *rightmost column*. It can be observed that the two segmentations are very similar, with a DSC equal to 0.9638 and an energy difference of only -1.33%. In Figure 7, we illustrate 3D segmentation results by showing the

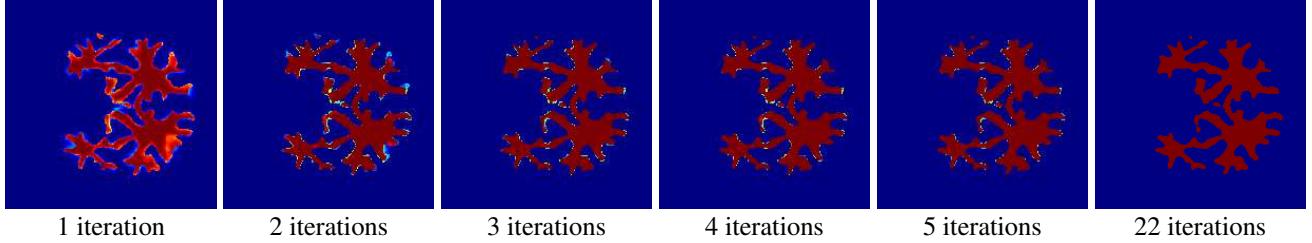


Figure 5: Evolution of the segmentation results (relaxed  $y$ ) with respect to the number of iterations.

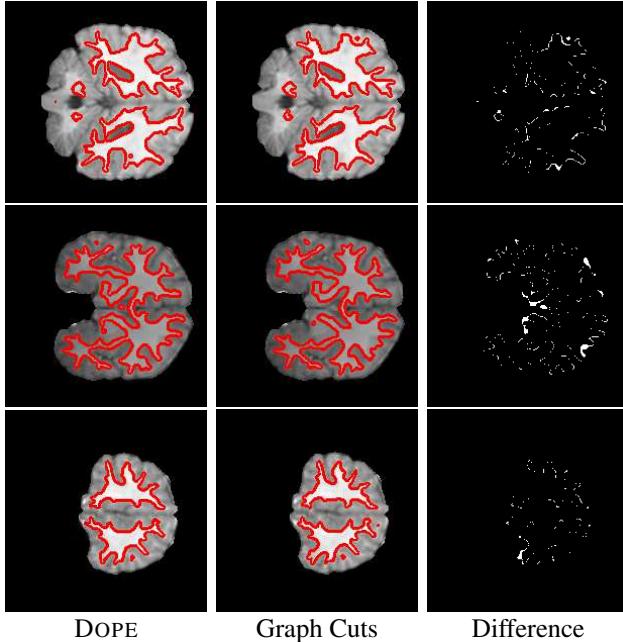


Figure 6: Several 2D images in axial view of the 3D segmentation of white matter generated by our DOPE approach (*left*) and serial graph cuts (*middle*). Differences between both segmentations are shown in the last column.

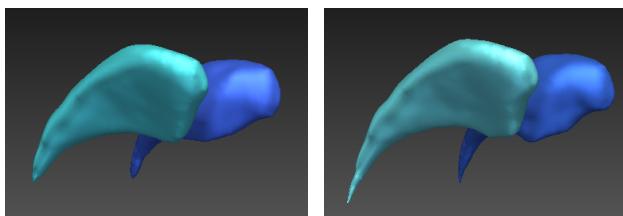


Figure 7: 3D volumes segmented by serial GC (*left*) and our DOPE approach.

surface of the left and right putamen regions, extracted by our method and sGC. Note that, in this case, the probability maps of these specific regions were used as unary potentials in the energy function.

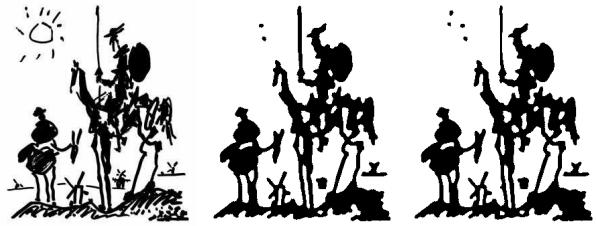
### 3.3. Curvature regularization

We demonstrated that our general formulation can distribute the computations of a powerful serial sub-modular optimization algorithm (i.e., BK) without affecting the quality of the energies at convergence. In this section, we report a curvature regularization experiment to illustrate how our method can also distribute the computations of non-submodular optimization techniques. Specifically, we focused on distributing the computations of the stat-of-the-art LSA-TR method in [10].

Table 1 reports the energies obtained by LSA-TR non-submodular optimization [10] and a distributed version based on our DOPE formulation. To obtain these energies, we employed the squared curvature model and the Picasso's ink drawing used in [22]. Fig. 8 depicts the results of this experiment. Notice that, by distributing the computations of LSA-TR with our DOPE formulation, we obtained a very similar result while reducing computation time.

	LSA-TR [10]	LSA-TR (DOPE) (8 sub-blocks)
Energy	$1.0906 \times 10^4$	$1.1115 \times 10^4$
Time	174 s	53 s

Table 1: Energies obtained by LSA-TR non-submodular optimization [10] and our formulation.



Original      LSA-TR [10]      LSA-TR(DOPE)  
Figure 8: Curvature regularization results of Picasso's ink drawing using the trust region (LSA-TR) method in [10] and our DOPE formulation.

## 4. Conclusion

In this work, we have formulated a general Alternating Direction Method of Multipliers (ADMM) technique that systematically distributes computations in the context of pairwise functions. Our method is scalable, allowing computations for large images by decomposing the problem into a large set of small sub-problems that can be solved in parallel. Unlike existing approaches, which distributes computation to a restricted number of cores [20], our algorithm can be easily adapted to the number of available cores. As shown in the results, another advantage of our technique is that it requires a small number of iterations to converge: about 3 iterations for high-definition 2D images and 5-10 iterations for 3D volumes. This allows our method to obtain segmentation results consistent with those of sGC, while allowing distributed computations.

While our experiments have focused on a standard Potts model, one of the main benefits of our formulation lies in its generality. This generality has been proven by distributing the computations in the context of a non-submodular function. Thus, our approach is not technique-specific, and can be applied to a wide variety of pairwise potentials, including dense (or fully connected) models. In future work, we plan to extend our method to such models, e.g., the dense pairwise potentials in [16], and to the case of multi-label segmentation.

## A. Proof of Theorem 1

Using relationship (4) in Eq. (3), and relaxing the integer constraints on  $\mathbf{y}$ , the general segmentation problem (3) can be reformulated as

$$\begin{aligned} \arg \min_{\substack{\mathbf{y} \in \mathbb{R}^{|\Omega|} \\ \hat{\mathbf{y}}_k \in \{0,1\}^{|\Omega_k|}}} & \sum_{k=1}^K \mathbf{u}^\top Q^{-1} S_k^\top \hat{\mathbf{y}}_k \\ & + \lambda \sum_{k=1}^K \sum_{l=1}^K \hat{\mathbf{y}}_k^\top S_k Q^{-1} L Q^{-1} S_l^\top \hat{\mathbf{y}}_l \end{aligned} \quad (13)$$

where  $\hat{\mathbf{y}}_k = S_k \mathbf{y}$  and  $k = 1, \dots, K$ . The unary term can be simplified by defining vectors  $\hat{\mathbf{u}}_k = S_k Q^{-1} \mathbf{u}$ , and then the problem can be written as

$$\begin{aligned} \arg \min_{\substack{\mathbf{y} \in \mathbb{R}^{|\Omega|} \\ \hat{\mathbf{y}}_k \in \{0,1\}^{|\Omega_k|}}} & \sum_{k=1}^K \hat{\mathbf{u}}_k^\top \hat{\mathbf{y}}_k + \lambda \sum_{k=1}^K \sum_{l=1}^K \hat{\mathbf{y}}_k^\top S_k Q^{-1} L Q^{-1} S_l^\top \hat{\mathbf{y}}_l. \end{aligned} \quad (14)$$

We notice that the segmentation vectors of each block are still coupled in the right-most term of this new cost function. In order to segment each block independently, we thus split this term in two: the cost of assigning labels to pixels in the

same block and in different blocks:

$$\begin{aligned} \arg \min_{\substack{\mathbf{y} \in \mathbb{R}^{|\Omega|} \\ \hat{\mathbf{y}}_k \in \{0,1\}^{|\Omega_k|}}} & \underbrace{\sum_{k=1}^K \hat{\mathbf{u}}_k^\top \hat{\mathbf{y}}_k}_{\textcircled{1}} + \underbrace{\lambda \sum_{k=1}^K \hat{\mathbf{y}}_k^\top S_k Q^{-1} L Q^{-1} S_k^\top \hat{\mathbf{y}}_k}_{\textcircled{2}} \\ & + \underbrace{\lambda \sum_{k=1}^K \sum_{l \neq k}^K \hat{\mathbf{y}}_k^\top S_k Q^{-1} L Q^{-1} S_l^\top \hat{\mathbf{y}}_l}_{\textcircled{3}}. \end{aligned} \quad (15)$$

Using the fact that  $L = D - W$  and  $\hat{\mathbf{y}}_k = S_k \mathbf{y}$ , we can transform  $\textcircled{2}$  as follows:

$$\begin{aligned} \textcircled{2} &= \lambda \sum_{k=1}^K \hat{\mathbf{y}}_k^\top S_k Q^{-1} (D - W) Q^{-1} S_k^\top \hat{\mathbf{y}}_k \\ &= \lambda \sum_{k=1}^K \hat{\mathbf{y}}_k^\top (S_k Q^{-1} D Q^{-1} S_k^\top - \widehat{W}_k + \widehat{D}_k - \widehat{D}_k) \hat{\mathbf{y}}_k \\ &= \lambda \sum_{k=1}^K \hat{\mathbf{y}}_k^\top \widehat{L}_k \hat{\mathbf{y}}_k + \hat{\mathbf{y}}_k^\top (S_k Q^{-1} D Q^{-1} S_k^\top - \widehat{D}_k) S_k \mathbf{y}, \end{aligned} \quad (16)$$

where  $\widehat{W}_k = S_k Q^{-1} W Q^{-1} S_k^\top$  and  $\widehat{L}_k = \widehat{D}_k - \widehat{W}_k$  is the Laplacian of  $\widehat{W}_k$ . Likewise, since  $\sum_{l \neq k} S_l^\top S_l = Q - S_k^\top S_k$ , the rightmost term in (15) becomes

$$\begin{aligned} \textcircled{3} &= \lambda \sum_{k=1}^K \hat{\mathbf{y}}_k^\top S_k Q^{-1} L Q^{-1} \sum_{l \neq k} S_l^\top S_l \mathbf{y} \\ &= \lambda \sum_{k=1}^K \hat{\mathbf{y}}_k^\top S_k Q^{-1} L (I - Q^{-1} S_k^\top S_k) \mathbf{y} \\ &= \lambda \sum_{k=1}^K \hat{\mathbf{y}}_k^\top S_k Q^{-1} L \mathbf{y} + \hat{\mathbf{y}}_k^\top (\widehat{W}_k - S_k Q^{-1} D Q^{-1} S_k^\top) S_k \mathbf{y}. \end{aligned} \quad (17)$$

Replacing  $\textcircled{2}$  and  $\textcircled{3}$  by these equalities, the problem reduces to

$$\begin{aligned} \arg \min_{\substack{\mathbf{y} \in \mathbb{R}^{|\Omega|} \\ \hat{\mathbf{y}}_k \in \{0,1\}^{|\Omega_k|}}} & \sum_{k=1}^K \hat{\mathbf{u}}_k^\top \hat{\mathbf{y}}_k + \lambda \sum_{k=1}^K \hat{\mathbf{y}}_k^\top \widehat{L}_k \hat{\mathbf{y}}_k \\ & + \lambda \sum_{k=1}^K \hat{\mathbf{y}}_k^\top (S_k Q^{-1} L - \widehat{L}_k S_k) \mathbf{y}, \end{aligned} \quad (18)$$

which is the formulation of Theorem 1.

## References

- [1] P. Baque, T. M. Bagautdinov, F. Fleuret, and P. Fua. Principled parallel mean-field inference for discrete random fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5848–5857, 2016. 1, 2
- [2] A. Bhushnurmeh and C. J. Taylor. Graph cuts via  $l_1$  norm minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(10):1866–1871, 2008. 2
- [3] A. Blake, P. Kohli, and C. Rother. *Markov Random Fields for Vision and Image Processing*. MIT press, 2011. 1
- [4] Y. Boykov and G. Funka Lea. Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006. 1, 4
- [5] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004. 1, 2, 3, 4
- [6] N. B. Bras, J. Bioucas-Dias, R. C. Martins, and A. C. Serra. An alternating direction algorithm for total variation reconstruction of distributed parameters. *IEEE Transactions on Image Processing*, 21(6):3004–3016, 2012. 3
- [7] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011. 2
- [8] A. Delong and Y. Boykov. A scalable graph-cut algorithm for N-D grids. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 2
- [9] J. Dolz, C. Desrosiers, and I. B. Ayed. 3D fully convolutional networks for subcortical segmentation in MRI: A large-scale study. *arXiv preprint arXiv:1612.03925*, 2016. 5
- [10] L. Gorelick, Y. Boykov, O. Veksler, I. B. Ayed, and A. Delong. Submodularization for binary pairwise energies. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1154–1161, 2014. 1, 4, 7
- [11] L. Gorelick, F. R. Schmidt, and Y. Boykov. Fast trust region for segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1714–1721, 2013. 2
- [12] M. R. Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969. 3
- [13] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, T. Kröger, J. Lellmann, N. Komodakis, B. Savchynskyy, and C. Rother. A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision*, 115(2):155–184, 2015. 1
- [14] Y. Kee, M. Souiai, D. Cremers, and J. Kim. Sequential convex relaxation for mutual information-based unsupervised figure-ground segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4082–4089, 2014. 2
- [15] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions via graph cuts. In *IEEE International Conference on Computer Vision (ICCV)*, pages 508–515, 2001. 1
- [16] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *Advances in Neural Information Processing Systems (NIPS)*, pages 109–117, 2011. 1, 2, 8
- [17] V. S. Lempitsky and Y. Boykov. Global optimization for shape fitting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 1
- [18] V. S. Lempitsky, S. Roth, and C. Rother. FusionFlow: Discrete-continuous optimization for optical flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 1
- [19] J. Liu and J. Sun. Parallel graph-cuts by adaptive bottom-up merging. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2181–2188, 2010. 2
- [20] J. Liu, J. Sun, and H.-Y. Shum. Paint selection. In *ACM Transactions on Graphics (ToG)*, volume 28, pages 69:1–7. ACM, 2009. 8
- [21] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967. 5
- [22] C. Nieuwenhuis, E. Töpke, L. Gorelick, O. Veksler, and Y. Boykov. Efficient squared curvature. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4098–4105, 2014. 1, 7
- [23] K. Punithakumar, J. Yuan, I. B. Ayed, S. Li, and Y. Boykov. A convex max-flow approach to distribution-based figure-ground separation. *SIAM J. Imaging Sciences*, 5(4):1333–1354, 2012. 2
- [24] M. Souiai, M. R. Oswald, Y. Kee, J. Kim, M. Pollefeys, and D. Cremers. Entropy minimization for convex relaxation approaches. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1778–1786, 2015. 2

- [25] P. Strandmark and F. Kahl. Parallel and distributed graph cuts by dual decomposition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2085–2092, 2010. 2
- [26] M. Tang, I. Ben Ayed, and Y. Boykov. Pseudo-bound optimization for binary energies. In *European Conference on Computer Vision (ECCV)*, pages 691–707, 2014. 2
- [27] M. Tang, I. Ben Ayed, D. Marin, and Y. Boykov. Secrets of GrabCut and kernel k-means. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1555–1563, 2015. 1
- [28] M. Tang, I. Ben Ayed, D. Marin, and Y. Boykov. Normalized cut meets MRF. In *European Conference on Computer Vision (ECCV)*, 2016. 1
- [29] T. Taniai, Y. Matsushita, and T. Naemura. Superdifferential cuts for binary energies. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2030–2038, 2015. 1
- [30] J. Yuan, E. Bae, and X.-C. Tai. A study on continuous max-flow and min-cut approaches. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 2