Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.2013.mssvm09

Vol. **6**, No. 1, pp. 169-198
February 2013

Multi-Label Markov Random Fields as an Efficient and Effective Tool for Image Segmentation, Total Variations and Regularization

Dorit S. Hochbaum*

Department of Industrial Engineering and Operations Research, University of California, Berkeley, Ca 94720, USA.

Received 6 December 2011; Accepted (in revised version) 12 September 2012

Available online 11 January 2013

Abstract. One of the classical optimization models for image segmentation is the well known Markov Random Fields (MRF) model. This model is a discrete optimization problem, which is shown here to formulate many continuous models used in image segmentation. In spite of the presence of MRF in the literature, the dominant perception has been that the model is not effective for image segmentation. We show here that the reason for the non-effectiveness is due to the lack of access to the optimal solution. Instead of solving optimally, heuristics have been engaged. Those heuristic methods cannot guarantee the quality of the solution nor the running time of the algorithm. Worse still, heuristics do not link directly the input functions and parameters to the output thus obscuring what would be ideal choices of parameters and functions which are to be selected by users in each particular application context.

We describe here how MRF can model and solve efficiently several known continuous models for image segmentation and describe briefly a very efficient polynomial time algorithm, which is provably fastest possible, to solve optimally the MRF problem. The MRF algorithm is enhanced here compared to the algorithm in Hochbaum (2001) by allowing the set of assigned labels to be any discrete set. Other enhancements include dynamic features that permit adjustments to the input parameters and solves optimally for these changes with minimal computation time. Several new theoretical results on the properties of the algorithm are proved here and are demonstrated for images in the context of medical and biological imaging. An interactive implementation tool for MRF is described, and its performance and flexibility in practice are demonstrated via computational experiments.

We conclude that many continuous models common in image segmentation have discrete analogs to various special cases of MRF and as such are solved optimally and efficiently, rather than with the use of continuous techniques, such as PDE methods, that restrict the type of functions used and furthermore, can only guarantee convergence to a local minimum.

AMS subject classifications: 05C21, 05C85, 68W40, 68U10

Key words: Total variation, Markov random fields, image segmentation, parametric cuts.

*Corresponding author. *Email address:* hochbaum@ieor.berkeley.edu (D. S. Hochbaum)

1. Introduction

Partitioning and grouping of similar objects plays a fundamental role in image segmentation and in clustering problems. In such problems the goals are to group together similar objects, or pixels in the case of image processing. Given an input image, the objective of image segmentation is to recognize the salient features in the image. Each feature set is grouped together in one segment represented by some uniform color area.

A noisy or corrupted image is characterized by lacking uniform color areas, which are assumed to characterize a true image. Rather, in such image there are adjacent pixels of different color areas. To achieve higher degree of uniform color areas, it is reasonable to assign a penalty to neighboring pixels that have different colors associated with them. On the other hand, the purpose of the segmentation is to represent the "true" image. For that purpose the given assignment of colors in the input image is considered to be the "priors" on the colors of the pixels, and as such, the best estimate available on their true labels. Therefore, any change in those priors is assigned a penalty for deviating from the priors.

The Markov Random Fields problem for image segmentation is to assign colors to the pixels so that the total penalty is minimized. The penalty consists of two terms. One is the *separation* penalty, or *smoothing* term, and the second is the *deviation* penalty, or *fidelity* term. For this reason we refer to this penalty minimization problem also as the *separation-deviation* problem. This problem has been extensively studied over the past two decades, see, e.g., [4,7,19,20,28,31].

The input to the problem is a graph $G=(V,E)$, where in the case of image segmentation each pixel is represented as a node in V. Let $N(i)$ be the set of neighbors of node $i \in V$. For each $j \in N(i)$ the pair of nodes $\{i,j\}$ have an edge $[i,j] \in E$ connecting them. Each node $i \in V$ has a deviation function $G_i()$ associated with it, and each edge $[i,j] \in E$ has an associated separation function $F_{ij}()$. The problem formulation, described in full detail in Section 5 is,

$$\text{(MRF)} \quad \min \sum_{i \in V} G_i(x_i) + \sum_{i \in V} \sum_{j \in N(i)} F_{ij}(x_i - x_j)$$
$$\text{subject to} \quad x_i \in X_i, \quad \forall i \in V.$$

The sets X_i consists of a collection of discrete labels, $\{a_1, a_2, \cdots, \mathbf{a}_n\}$, that can be assumed by variables x_i in a feasible solution. That is, MRF is a *multi-label* assignment. It is noted that the concept of "colors" associated with pixels can be replaced by any other scalar characterization of pixels or voxels, such as texture. We refer here to colors as a representation of such characterizations.

The complexity of MRF depends on the form of the penalty functions. This complexity of MRF was fully resolved and classified according to the properties of the penalty functions in [26] for sets of consecutive integers $X_i = \{\ell_i, \ell_{i+1}, \cdots, u_i\}$. For convex penalty functions MRF is polynomial time solvable, and for non-convex the problem is NP-hard. The cases when the deviation penalty functions are convex and the separation penalty functions are (bi-)linear (defined below) was shown by Hochbaum [26] to be solvable in polynomial

time using a parametric cut procedure. Furthermore, it was shown there that the complexity of the algorithm is the *fastest possible*. The case where both separation and deviation penalty functions are convex were also shown to be solvable very efficiently, and within a multiplicative log factor of fastest possible, by [1, 2]. For non-convex deviation functions and convex separation functions the problem is solved in *pseudo-polynomial time*, that depends on the number of label values, or the range of the variables. This running time is the time required to solve a minimum cut problem on a graph with number of nodes that depends on the number of labels [1]. When both type of penalty functions are non-convex the MRF problem is NP-hard. These results, all for multi-labels MRF, are summarized in Table 1.

Table 1: Complexity of MRF problems.

Deviation function $G_i()$	**Separation function** $F_{ij}()$	**Complexity**	**Reference**
Convex	Convex	$O(mn\log\frac{n^2}{m}\log nU)$	Ahuja et al. [2]
Convex	Bilinear	$O(mn\log\frac{n^2}{m}+n\log U)$	Hochbaum [26]
Quadratic	Bilinear	$O(mn\log\frac{n^2}{m})$	Hochbaum [26]
General (nonlinear)	convex	$O(mnU^2\log\frac{n^2U}{m})$	Ahuja et al. [1]
Linear	Nonlinear	NP-hard	Hochbaum [26]

In Table 1 and henceforth we let $U=\max_i|X_i|$ be the number of labels. In referring to complexity of algorithms here, we will use the standard notation of $n=|V|$ the number of nodes in the graph, $m=|E|$ the number of edges (or arcs, in case of directed graphs) in the graph, and $T(n,m)$ the complexity of the minimum s,t-cut problem on a graph with n nodes and m arcs or edges. A *bi-linear* separation function is a (two) piecewise linear function with a linear function in the range $x_i\geq x_j$ and a linear function in the range $x_j\geq x_i$, of the form:

$$F_{ij}(x_i-x_j)=\begin{cases}u_{ij}(x_i-x_j), & \text{if } x_i>x_j,\\ 0, & \text{if } x_i=x_j,\\ u_{ji}(x_j-x_i), & \text{if } x_i<x_j.\end{cases}$$

Comments

1. When the set of discrete values for each variable consists of two values, $U=2$, then the MRF problem (for any deviation or separation functions) is the *binary* MRF. The link of binary MRF to the minimum cut problem was first presented by Greig et al. [22], and later from a different perspective, as an s-excess problem, in [25, 26].
2. The complexities of the algorithms of [26] and [1] stated in Table 1 for convex-bilinear, quadratic-bilinear and nonlinear-convex, are fastest possible.
3. The complexity $O(mn\log(n^2/m))$ is replaced by the respective complexity of a minimum s,t-cut procedure that can be implemented efficiently as parametric. To date, the only known parametric cut procedures are the HPF (Hochbaum's PseudoFlow) algorithm and push-relabel algorithm. For these algorithms $T(n,m)$ is $O(mn\log(n^2/m))$, or parametric HPF can also be implemented in $O(n^3)$, [23].

4. The MRF problems in the table include those with convex functions such as $F_{ij}(x_i - x_j) = (x_i - x_j - a_{ij})^2$. Such functions do *not* assume the value 0 for $x_i - x_j = 0$. This is notable, since in segmentation applications both $G_i()$ and $F_{ij}()$ are assumed to map the zero argument to 0. The algorithms referenced in Table 1 that solve the respective MRF problems with either convex separation or nonlinear or convex deviation apply also to functions that do not assume the value 0 for arguments equal to 0, and therefore solve these more general problems.
5. The MRF algorithms cited can all be applied to *directed* graphs.
6. The number of labels given U is presented here as a set of consecutive integers. (See Remark 1.1 as to why it is sufficient to consider integers.) However, as proved here in Section 5 the algorithm for convex-bilinear given here can apply to *any* discrete set of U values, that are not required to be consecutive (or converted to integers).
7. The running time for nonlinear-convex is "pseudo-polynomial" as it depends on the number of labels U. The number U is not a part of the input. It is sufficient to indicate the length of the interval of the values, which requires $\log U$. Solving this problem provably requires at least pseudo-polynomial running time, as shown in Lemma 1.1 below. In that sense this problem is harder than e.g., the (weakly) NP-hard Knapsack problem, which can be solved in pseudo-polynomial time, but may be polynomial time solvable if $NP = P$. A pseudo-polynomial time algorithm is therefore best possible in the sense that it cannot be improved to polynomial time algorithm, that would depend on $\log U$ only.

Remark 1.1. It is always assumed that the input data describing a problem instance is given in integers. This is since digital computers can only process integers, or equivalently, rational numbers. If the input is given in rational numbers, there exists a scalar large enough so the input data can be scaled and converted to an equivalent instance in integers. When one considers nonlinear functions the solution may be irrational. Indeed such solutions cannot be expressed as the output of a computer algorithm, and can be at best attained within pre-specified accuracy, or number of significant digits. An ϵ-accurate solution is a solution that is within norm max distance of ϵ from the true solution. For more on this subject the reader is referred to [27].

Lemma 1.1. *The complexity of MRF with nonlinear deviation functions and no separation functions, all $F_{ij}() = 0$, is at least pseudo-polynomial.*

Proof. For $G_i()$ nonlinear consider the MRF problem,

$$\min \sum_{i \in V} G_i(x_i)$$
$$\text{subject to} \quad 0 \leq x_i \leq U \quad \text{integer} \quad \forall i \in V.$$

This problem is *decomposable* to n nonlinear optimization problems, for all $i = 1, \cdots, n$, of the form,

$$\min G_i(x_i)$$
$$\text{subject to} \quad 0 \leq x_i \leq U \quad \text{integer}.$$

Solving this problem is equivalent to finding the minimum entry in the array of length $U+1$, $(G_i(0), G_i(1), \cdots, G_i(U))$. Since the entries in this array are arbitrary, it is necessary to inspect all of them. Therefore the running time of solving this nonlinear problem is $\Omega(U)$, and the complexity of nonlinear deviation MRF is $\Omega(nU)$. □

We conclude that MRF with nonlinear deviation functions and convex separation functions is only harder than the respective MRF with no separation functions. Therefore pseudo-polynomial complexity is best possible for nonlinear deviation MRF. In that sense the complexity of nonlinear deviation convex separation given in Table 1 is fastest possible.

Even though, with the results above, the MRF problem is fully understood and solved, there is still much active work on attempting to solve certain image segmentation models using restricted forms of MRF and/or solving them heuristically. One class of such problems is the continuous models of Mumford-Shah, total variations and regularization.

Continuous optimization models of total variations and regularization have been utilized in image analysis for the purpose of *denoising* an image. In solving these models researchers employ continuous methodologies and heuristics. Often the functions and parameters selected are chosen so as to facilitate the solution techniques instead to reflect the properties of the segmented features. Even if the model selected is MRF it is often approached with continuous techniques. For example, a recent work by Pock et al. [34] provides approximate continuous methods for solving MRF with the Potts model in which the separation functions assume the value 0 only if $x_i = x_j$ and $F_{ij}(x_i - x_j) = 1$ if $x_i \neq x_j$. Although such F_{ij} functions are non-convex, and the respective MRF problem is NP-hard, Pock et al. [34] address that problem with continuous techniques also for the case of a bipartition (where the variables x_i are binary). The method proposed, that relaxes the integrality, may not converge to an optimal solution, and the running time cannot be determined in advance. This is surprising, given that the exact binary problem can be solved within guaranteed polynomial time complexity. Moreover, digital images are inherently discrete, and considering them as continuous causes loss of accuracy. The output of a continuous method must be mapped back to digital image information, entailing further loss of accuracy. Additional examples and details of this phenomenon are provided in Section 2. We demonstrate that several classical continuous models are better represented with the MRF model and thus benefit from the algorithmic efficiency, described here, of solving it.

The main contributions here include: The introduction of the efficient use of the MRF algorithm for continuous and discrete models in vision; a new result allowing to solve convex-bilinear MRF for any discrete set of labels; new theoretical results on parametric cut; the implementation of the MRF algorithm with new flexibility features, the MRF tool, presented here for the first time; and experimental results that demonstrate the capabilities of the MRF tool.

The next section reviews representative research to date that addressed algorithms for MRF for vision problems or continuous models modeled as MRF. In Section 3 we describe the HPF algorithm for maximum flow and minimum cut, which is a main subroutine for the MRF algorithms described. Section 4 describes a few known continuous models that were previously addressed with continuous techniques, and their representation as MRF,

resulting in efficient algorithms. In Section 5 we provide a sketch of the MRF polynomial time algorithm and theoretical results on the parametric cut algorithm. This section includes a new result in Lemma 5.3 on solving for any discrete set of permissible labels, not necessarily consecutive integers, without an increase in the running time. Section 6 describes a user-interactive tool that runs the convex-bilinear MRF algorithm and its properties. That tool allows for several flexibility features that include changing the balance between separation and deviation, and allowing to increase the fidelity of the output with respect to a subset of the given colors. In Section 7 we present experimental results that show how the MRF algorithm's output depends on the choice of the parameters and the available selected features. In this section we also provide comparisons with the image segmentation algorithm based on Shi and Malik's eigenvector approach [37]. Finally we provide conclusions and brief summary in Section 8.

2. Related literature on algorithms for MRF

As noted already, the binary MRF was linked to the minimum cut problem originally by Greig et al. [22], and later from a different perspective, as an s-excess problem, in [25,26] where it was shown equivalent to a minimum s,t-cut problem. In that sense the binary MRF problem has been fully resolved. The MRF problem has been further studied over the past two decades, see, e.g., [4,7,19,20,28]. These studies present either heuristic algorithms or non-polynomial optimization algorithms for the MRF problem. Some attention has been devoted to binary MRF problems with *submodular* separation functions which do not appear to fall directly within the formulation of MRF given here, since those energy function are non-separable. However, Kolmogorov and Zabih, [30], demonstrated that such functions can be represented by a graph. This means that it is possible to formulate the submodular MRF problem as a simple binary MRF problem. Alternatively, this means that such submodular functions *are* separable and can be treated as any other binary MRF problem, and consequently solved with a minimum cut procedure on the respective graph.

Ishikawa presented in [29] a "multi-labels" MRF algorithm with convex separation and arbitrary nonlinear deviation functions. Although the complexity of Ishikawa's algorithm is not stated, it appear to have at least pseudo-polynomial complexity as the running time depends on the number of labels. Ishikawa was not aware that a more efficient pseudo-polynomial time algorithm already existed for the same problem. The nonlinear deviation algorithm was discussed in [26] (and referenced there as a 1999 manuscript) and described in detail in [1]. That algorithm, listed in Table 1, solves the nonlinear deviation MRF problem on U labels with a *single* minimum s,t-cut procedure on a graph on nU nodes and mU edges. As shown in Lemma 1.1 the run time of the algorithm of Ahuja et al. [1] is fastest possible in the sense that the dependence on U in the running time cannot be removed.

When the deviation functions are convex as well, which is the case in all the applications discussed in the literature, that run time is shown to be polynomial: Ahuja et al. [1] present a more efficient version of this algorithm that runs in $\log U$ applications of minimum s,t-cut on graphs of polynomial size (rather than pseudo-polynomial) which

improves on the run time of Ishikawa's algorithm. Moreover, for arbitrary convex penalty functions the run time of the algorithm in [2] is polynomial and the fastest known for MRF problems, $O(mn \log n_2/m \log nU)$

Bae and Tai presented in [3] a simplification of the Mumford-Shah model that reduces to an MRF problem on multiple labels (as opposed to a binary MRF problem). The problem formulation for setting $\phi_i = u_i$ is:

$$\min \sum_{i \in V} \delta^2 (u_i - u_i^0)^2 + \nu \sum_{[i,j] \in E} \frac{1}{2} w_{ij} |u_i - u_j|. \tag{2.1}$$

Bae and Tai [3] solved the problem by using, at each call to such an MRF problem, Ishikawa's algorithm [29]. As pointed above, that algorithm is less efficient than the respective algorithms in [1, 2, 26]. Moreover, problem (2.1) has bilinear separation terms and quadratic deviation terms. Thus there is no need to employ an algorithm to solve the harder problem with nonlinear deviation functions. Such a problem was shown in [26] to be solved in strongly polynomial time equal to $T(n,m)$ for a graph on n nodes and m adjacencies (edges). That running time is also the fastest possible in that the problem is at least as hard as a minimum s,t-cut problem and therefore if of complexity $\Omega(T(n,m))$. Therefore Ishikawa's algorithm is inappropriate to use in this context. Bae and Tai [3] further refer to Darbon and Sigelle [15] as being able to solve problem (2.1) by a sequence of min-cuts. As pointed out above, this problem can be solved by a single minimum cut procedure.

In [15], Darbon and Sigelle address, as here, a discrete version of the total variation problem. They present the problem as a collection of binary MRF problems, and therefore not of polynomial time complexity. They considered the deviation terms to be either linear or convex quadratic and the separation terms are bi-linear corresponding to absolute value. Therefore this problem is solved in the run time of a single minimum cut and is strongly polynomial (independent of the number of labels), as shown in [26]. Although aware of this algorithm, they do not use it, since "Hochbaum in [26] *did not present numerical results*". One of our goals here is to present numerical results, in addition to the theoretical foundations and algorithms for MRF.

Pock et al. in [34] claim to show that in the isotropic, and thus non-separable, case with Potts penalty functions, the continuous approach dominates the discrete one. A simplification of the problem they address is cast as MRF, and therefore separable problem. They refer to Ishikawa's algorithm for solving that MRF exactly. As noted already, this algorithm is not the best even for nonlinear deviation terms and is dominated by the algorithm in [1]. Although unclear, it appears that Pock et al. use convex deviation terms rather than nonlinear deviation. And since they use for the separation terms absolute value (bilinear) functions, such an MRF problem is solved in polynomial time and very efficiently, with the algorithm described in [26].

Another issue is that Pock et al. also claim that the size of the graph makes it computationally impossible to compute the minimum cut. They do not state however which algorithm was used to solve the cut problem. Recently, a popular cut algorithm used in

vision problem is by Boykov and Kolmogorov, [6]. But that algorithm does not work well for the setting here. This is because the neighborhood in the constructed graph is different from a grid and Boykov Kolmogorov algorithm does not run as efficiently as HPF except for low connectivity low degree graphs. A study comparing the performance of Boykov Kolmogorov algorithm to HPF for vision problems is described in [16] and demonstrates that other than small size low connectivity vision graphs, where that algorithm is slightly faster, HPF provides substantial improvement for vision problems solving the minimum cut problem and is the fastest algorithm to date for cut problems and vision problems.

Based on the combination of Ishikawa's algorithm with a choice of minimum cut algorithm that is suboptimal, Pock et al. conclude that approaching their vision problem with a continuous method yields better results and is more efficient. However, the use of the MRF algorithm of [26], along with the HPF algorithm would have provided results much more efficiently, and based on this author's experience, the results are likely to have been of better quality than those achieved with the continuous method. Until tested properly, the claim of Pock et al. that the continuous method performs better remains unsettled.

In [24] Hochbaum et al. conduct an experimentation with the parametric cut algorithm HPF for the convex-deviation bilinear-separation MRF. The results indicate that the MRF is a fast and effective method at denoising medical images as well as segmenting tissue types, organs, lesions, and other features within medical images.

3. HPF — the pseudoflow algorithm

The pseudoflow algorithm, HPF, plays an important role here, both in terms of efficiency, and in its adaptability to solve the parametric cut problem.

3.1. Definitions and notation

Let G_{st} be a graph (V_{st}, A_{st}), where $V_{st} = V \cup \{s,t\}$ and $A_{st} = A \cup A_s \cup A_t$ in which A_s and A_t are the source-adjacent and sink-adjacent arcs respectively. The number of nodes $|V_{st}|$ is denoted by n, while the number of arcs $|A_{st}|$ is denoted by m. A flow $f = \{f_{ij}\}_{(i,j)\in A_{st}}$ is said to be *feasible* if it satisfies

(i) Flow balance constraints: for each $j \in V$, $\sum_{(i,j)\in A_{st}} f_{ij} = \sum_{(j,k)\in A_{st}} f_{jk}$ (i.e., inflow(j) = outflow(j)), and

(ii) Capacity constraints: the flow value is between the lower bound and upper bound capacity of the arc, i.e., $\ell_{ij} \le f_{ij} \le u_{ij}$. We assume henceforth w.l.o.g that $\ell_{ij} = 0$.

The *maximum flow* or *max-flow* problem on a directed capacitated graph with two distinguished nodes—a source and a sink—is to find a feasible flow f^* that maximizes the amount of flow that can be sent from the source to the sink while satisfying flow balance constraints and capacity constraints.

A *cut* is a partition of nodes $S \cup T = V$, $T = \bar{S}$, with $s \in S, t \in T$. The set S is referred to as the *source set* and T as the *sink set*. The capacity of a cut is the sum of capacities of arcs that go from S to T denoted by $C(S,T) = \sum_{i\in S, j\in T, (i,j)\in A} u_{ij}$. The *minimum s-t*

cut problem, henceforth referred to as the *min-cut* problem, is to find a bi-partition of nodes—one containing the source and the other containing the sink—such that the sum of capacities of arcs from the source set to the sink set is minimized. In 1956, Ford and Fulkerson [17] established the *max-flow min-cut theorem*, which states that the value of a max-flow in any network is equal to the value of a min-cut.

Given a capacity-feasible flow, hence a flow that satisfies (ii), an arc (i,j) is said to be a *residual arc* if $(i,j) \in A_{st}$ and $f_{ij} < u_{ij}$ or $(j,i) \in A_{st}$ and $f_{ji} > 0$. For $(i,j) \in A_{st}$, the residual capacity of arc (i,j) with respect to the flow f is $c_{ij}^f = u_{ij} - f_{ij}$, and the residual capacity of the reverse arc (j,i) is $c_{ji}^f = f_{ij}$. Let A^f denote the set of residual arcs with respect to flow f in G_{st} which consists of all arcs or reverse arcs with positive residual capacity.

A *preflow* is a relaxation of a flow that satisfies capacity constraints, but inflow into a node is allowed to exceed the outflow. The *excess* of a node $v \in V$ is the inflow into that node minus the outflow denoted by $e(v) = \sum_{(u,v)\in A_{st}} f_{uv} - \sum_{(v,w)\in A_{st}} f_{vw}$. Thus a preflow may have nonnegative excess.

A *pseudoflow* is a flow vector that satisfies capacity constraints but may violate flow balance in either direction (inflow into a node needs not to be equal outflow). A negative excess is called a *deficit*.

3.2. Hochbaum's pseudo-flow algorithm

Hochbaum's pseudoflow algorithm, HPF, [25] works with pseudoflows that can violate flow balance constraints. HPF has a strongly polynomial complexity of $O(nm \log \frac{n^2}{m})$ [23]. It was shown to be fast in theory [25] and in practice [11] for general benchmark problems. In [16] HPF's performance for vision problems was compared to that of the push-relabel algorithm and its improvement, PAR, [21] and to Boykov and Kolmogorov's algorithm [6]. The results demonstrated that the superior performance of HPF for vision problems, compared to that of the other methods, makes it the most effective algorithm for large vision problems.

At each iteration of the algorithm a spanning forest is maintained: Each node in $v \in V$ is associated with at most one *parent* node w such that the *current arc*, (w,v), is in A^f; the corresponding *parent node* of v is denoted by $\text{parent}(v) = w$. The algorithm also associates with each node with a *root* that is defined constructively as follows: starting with node v, generate the sequence of nodes $\{v, v_1, v_2, \cdots, v_r\}$ defined by the current arcs $(v_1,v),(v_2,v_1),\cdots,(v_r,v_{r-1})$ until v_r has no current arc. Such root node v_r always exists [23,25]. Let the unique root of node v be denoted by $\text{root}(v)$. Note that if node v has no current arc, then $\text{root}(v) = v$. For each root node, the connected component it belongs to forms a tree, and is called a *branch*.

HPF algorithm can be initiated with any arbitrary initial *pseudoflow* (i.e., flow vector that may violate flow balance in either direction) that saturates source adjacent and sink-adjacent arcs. The *simple initialization* is generated by saturating all source-adjacent and sink-adjacent arcs, $A_s \cup A_t$, and setting all other arcs to have zero flow. This creates a set of source-adjacent nodes with excess, and a set of sink-adjacent nodes with deficit. All

other arcs have zero flow, and the set of initial current arcs is empty. Thus, each node is a singleton component of the forest for which it serves as a tree and the root of the tree.

The algorithm associates each node $v \in V$ with a distance label $d(v)$. For distance labels to be *valid*, they must satisfy,

(i) $d(t) = 0$;
(ii) for all $(i, j) \in A^f$, $d(i) \leq d(j) + 1$;
(iii) if i a non-root node $d(i) \geq d(\text{parent}(i))$.

Condition (iii) implies that in each branch the distance labels can only increase with the distance from the root of the branch. Distance labels are initialized at $d(i) = 1$ for all $i \in V$, $d(s) = n$ and $d(t) = 0$. Throughout the algorithm the distance labels maintain the invariant that they can only be lower than the shortest distance to t in the residual graph.

A residual arc (w, v) is said to be *admissible* if $d(w) = d(v) + 1$.

A node is said to be *active* if it has strictly positive excess. Given an admissible arc (w, v) with nodes w and v in different components, an *admissible path* is the path from root(w) to root(v) along the set of current arcs from root(w) to w, the arc (w, v), and the set of current arcs (in the reverse direction) from v to root(v).

An iteration of HPF consists of choosing an active component, with root node label $< n$ and searching for an admissible arc from a *lowest labeled* node w in this component. Choosing a lowest labeled node for processing ensures that an admissible arc is never between two nodes of the same component.

By construction (see [25]), the root is the lowest labeled node in a component and node labels are non-decreasing with their distance from the root of the component. Thus, all the lowest labeled nodes within a component form a branch (sub-tree) rooted at the root of the component. Once an active component is identified, all the lowest labeled nodes within the component are examined for admissible arcs by performing a depth-first-search in the branch starting at the root.

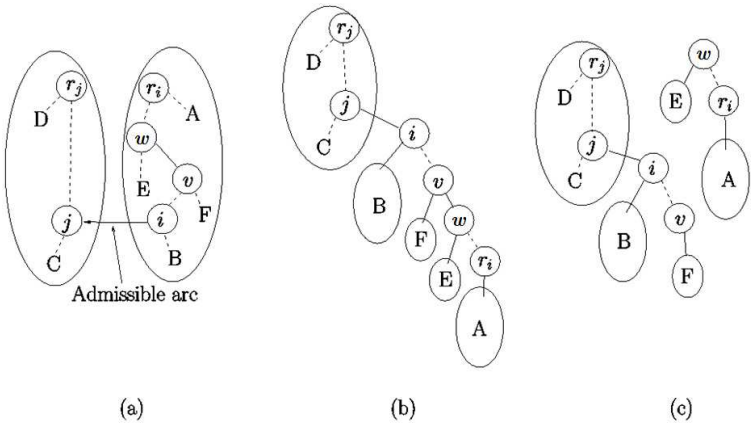

Figure 1: (a) Components before merger. (b) Before pushing flow along the admissible path from r_i to r_j. (c) New components generated when arc (w, v) leaves the current forest due to insufficient residual capacity.

If an admissible arc (w,v) is found, a *merger* operation is performed. The merger operation consists of pushing the entire excess of root(w) towards root(v) along the admissible path and updating the excesses and the arcs in the current forest. A schematic description of the merger operation is shown in Fig. 1. The pseudocode is given in Fig. 2.

```
/*
Min-cut stage of HPF algorithm.
*/
procedure HPF (V_st, A_st, c):
    begin
        SimpleInit (A_s, A_t, c);
        while ∃ an active component T with root r, where d(r) < n, do
            w ← r;
            while w ≠ ∅ do
                if ∃ admissible arc (w, v) do
                    Merger (root(w), ··· , w, v, ··· , root(v));
                    w ← ∅;
                else do
                    if ∃y ∈ T : (parent(y) = w) ∧ (d(y) = d(w)) do
                        w ← y;
                    else do {relabel}
                        d(w) ← d(w) + 1;
                        w ← parent(w);
    end
```

Figure 2: The min-cut stage of the HPF algorithm. At termination all nodes in label-n components are the source set of the min-cut.

If no admissible arc is found, $d(w)$ is increased by 1 unit for all lowest label nodes w in the component. The algorithm terminates when there are no active nodes with label $< n$. At termination all n labeled nodes form the source set of the min-cut.

The active component to be processed in each iteration can be selected arbitrarily. There are two variants of HPF with specific selection rules: (i) the lowest label HPF, where an active component with the lowest labeled root is processed at each iteration; and (ii) the highest label HPF, where an active component with the highest labeled root node is processed at each iteration.

The first stage of HPF terminates with the min-cut and a pseudoflow. The second stage converts this pseudoflow to a maximum feasible flow using *flow decomposition*. The running time of the second stage $O(m \log n)$ is dominated by that of the first stage. The experiments on vision problems in [16], and the experiments for general flow problems in [11], indicate that the time spent in flow recovery is small compared to the time to find the min-cut. Moreover, for vision problems, it is only necessary to identify the minimum cut in the network.

An important property of HPF is *monotonicity*, which is utilized in the parametric HPF

algorithm: Given two sets of capacities of arcs adjacent to source A_s with set 1 having all capacities equal or lesser than those of set 2. There are two corresponding sets of capacities of arcs adjacent to sink A_t with set 1 having all capacities equal or *larger* than set 2. Suppose HPF terminated with a minimum cut for set 1. The update, or change, of the capacities of A_s from set 1 to set 2 is upwards, and can only create newly active nodes. The update of the capacities of A_t from set 1 to set 2 is downwards, and can only increase the distance of nodes from the sink. For distance labels to be valid they need to be a lower bound on the actual distance to the sink. Therefore, this form of update maintains valid distance labels, but the current solution may no longer be optimal if there are active nodes. HPF then solves for set 2 of the capacities by continuing from the optimal solution of set 1. Since the complexity of the algorithm is dominated by a potential function that depends on the distance labels, this continuation and solving for set 2 is still within the complexity of a single minimum cut solution.

This property of the HPF algorithm is also used to prove that the source set of the min-cut for set 1 is *contained* in the source set of the min-cut for set 2-the *nestedness property*. This is because the nodes in the source set for set 1 all have label at least n. The update of the capacities does not change their status. Further processing may only add nodes to the source set by applying the relabel operation to some of the nodes.

4. Casting image segmentation continuous models as MRF

4.1. Total variations

In the total variation method [33,36] the recorded image is represented by the function which maps each pixel to its label (color). It is assumed that u_0 can be decomposed as $u_0 = u + v$ where u contains homogeneous regions with sharp prominent edges, and v contains additional texture and noise. The goal of the total variation method is to find u by minimizing the functional

$$\int_{\Omega} |\nabla u| dx dy + \alpha \|u - u_0\|.$$

This functional is defined on the plane, where (x, y) designate the position of each pixel in the image.

Although not immediately apparent, there is a connection between this problem and the MRF problem: The term $|\nabla u|$ captures the difference between each pixel and its neighborhood. The neighborhood can be set to any desirable set — it is not restricted to the commonly used grid neighborhood. This gradient term is thus the *separation* term. The second term $\alpha\|u - u_0\|$ is the deviation of the mapped function u from the recorded image u_0.

This total variation problem is solved by continuous techniques. One such method solves the associated Euler-Lagrange equation

$$u = u_0 + \frac{1}{2\alpha} \nabla \cdot \Big(\frac{\nabla u}{\|\nabla u\|} \Big).$$

In contrast to MRF, this method does not guarantee to deliver an optimal solution and its complexity is undetermined. For this quadratic deviation and absolution value separation MRF does deliver an optimal solution in strongly polynomial time.

In a more general set-up, the total variation regularization problem (TVR) the image is represented as $s(x)$ — a given function defined on an open subset Ω, and $f(x)$ is its *regularized* version, or for images, it is called the *denoising* of $s()$. We define two real functions $\gamma : R \to [0,\infty)$ and $\beta : R \to [0,\infty)$ which assume the value 0 for the argument of 0,

$$F(f) = \int_{\Omega} \gamma(f(x) - s(x))dx.$$

In the denoising literature F is called a *fidelity* term since it measures deviation from $s()$ which could be a noisy grayscale image. In our terminology, the fidelity term is the *deviation*.

A second function is the total variations on f, $TV(f)$: The discrete form of the total variations function is represented as a function f on a grid of discrete values in Ω and associated with a defined *neighborhood* of each grid point. Let the set of neighboring pairs be denoted by E. Then the total variation of f is $\sum_{[i,j]\in E} \beta(f(i) - f(j))$ for a function β often selected as the absolute value function: $\beta(x) = \max\{0, x\}$. For a constant α the *total variation regularization* of $s()$ is the function f that minimizes the weighted combination of the total variations and fidelity of f:

$$\min TV(f) + \alpha F(f).$$

Rudin, Osher and Fatemi [36] have studied TVRs of F where $\gamma(y) = y^2$, and Chan and Esedoglu [10] studied $\gamma(y) = |y|$.

Since MRF is solved in polynomial time for convex γ and convex β, consequently, the problem of Chan and Esedoglu is a special case solved by parametric cut, and the problem of Rudin et al. is a special case solved by the quadratic convex dual of min cost network flow. Both cases are efficiently solvable and the MRF algorithm guarantees an optimal solution in polynomial time.

4.2. Classes of the Mumford-Shah problem

Computational difficulties with traditional active contour methods were addressed with so-called *level set methods*, by Chan and Vese (2001) and by Osher and Fedkiw (2003) [9, 33]. The idea used in these methods is viewing contours as level sets of a function. For example, a contour in 2D may be viewed as all points (x, y) satisfying $f(x, y) = C$ for some function f and a given constant C. Chan-Vese considered the Mumford Shah functional but assumed a mapping to a simplified function that is piecewise constant and that can only attain two values, c_1 and c_2. Let Ω denote the image domain and let Ω_1 and Ω_2 be two subsets of Ω such that $\Omega = \Omega_1 \cup \Omega_2$. Define B as the boundary separating Ω_1 and Ω_2. The goal is to minimize the Mumford Shah functional which in the case of partition to two

subsets is:

$$F(c_1, c_2, f)$$
$$= \int_\Omega [c_1 - u_0(x,y)]^2 H(f) + [c_2 - u_0(x,y)]^2 (1 - H(f)) + \beta |\nabla H(f)| dx dy,$$

where $H(f) = 1$ if $(x,y) \in \Omega_1$ and 0 if $\in \Omega_2$. This model is a special case of MRF with two labels c_1 and c_2. That is, this is a binary MRF. Assume, w.l.o.g. that $c_2 > c_1$. We let each pixel position (x,y) be represented by a single index i. The variable x_i indicating the side of the bipartition $(S, \bar{S})$ that pixel i belongs to,

$$x_i = \begin{cases} 1, & \text{if } i \in S, \\ 0, & \text{if } i \in T = \bar{S}. \end{cases}$$

We let $w_i^1 = [c_1 - u_0(i)]^2$ if $i \in S$ and $w_i^2 = [c_2 - u_0(i)]^2$ if $i \in \bar{S}$. Also, let $W = \sum_{i \in \Omega} w_i^2$. Observe that,

$$\sum_{i \in S} w_i^1 x_i + \sum_{i \in \bar{S}} w_i^2 (1 - x_i) = W + \sum_{i \in \Omega} (w_i^1 - w_i^2) x_i.$$

The problem is then equivalently stated as the minimization problem:

$$W + \min_{S \subset \Omega} \sum_{i \in \Omega} (w_i^1 - w_i^2) x_i + \beta \sum_{[i,j] \in E} (c_2 - c_1) \cdot |x_i - x_j|.$$

This discrete version corresponds to anisotropic total variation in the continuous version, i.e., $|\nabla u|_1 = |u_x| + |u_y|$.

We let $\beta' = \beta(c_2 - c_1)$ noting that the model

$$\min_{S \subset \Omega} \sum_{i \in \Omega} (w_i^1 - w_i^2) x_i + \beta' \sum_{[i,j] \in E} |x_i - x_j|$$

is a particularly simple case of MRF. The functions $G_i(x_i) = (w_i^1 - w_i^2) x_i$ are linear, with positive or negative coefficients. For each adjacency edge $[i,j]$ the function $F_{ij}(x_i - x_j)$ is the absolute value, $|x_i - x_j|$. This is a special case of the bilinear MRF with the derivatives of the deviation terms equal to constants (for the use of the derivatives see the next section, Section 5). This type of MRF problem is solved by a single minimum s,t cut problem (non-parametric) on a graph on $|\Omega|$ nodes, [26] as explained next in Section 5.

5. The methodology

For the separation-deviation model of the image segmentation problem the input is an image constituting of a set of pixels each with a given color and a neighborhood relation between pairs of pixels. The decision is to assign each pixel a color assignment, that may be different from the given color of the pixel, so that neighboring pixels will tend to have the same color assignment. The aim is to modify the given color values as little as possible

while penalizing changes in color between neighboring pixels. The penalty function thus has two components: the deviation cost that accounts for modifying the color assignment of each pixel, and the separation cost that penalizes the extent of pairwise discontinuities in color assignment for each pair of neighboring pixels.

Formally, we are given an image which is a set of pixels V, with a real-valued intensity r_i for each pixel $i \in V$. The neighborhood of pixel i, which contains pixels adjacent to i, is denoted by $N(i)$. We wish to assign each pixel $i \in V$ an intensity x_i that belongs to a discrete finite set $X = \{i_1, i_2, \cdots, i_k\}$ so that the sum over all pixels of the *deviation* cost $G_i(\cdot)$ and the *separation* cost $F_{ij}(\cdot)$ is minimized. The deviation function depends on the deviation of the assigned color from the given intensity $G_i(x_i - r_i)$. The separation is a function of the difference in assigned intensities between adjacent pixels $F_{ij}(x_i - x_j)$. This is the MRF problem,

$$\min \sum_{i \in V} G_i(x_i) + \sum_{i \in V} \sum_{j \in N(i)} F_{ij}(x_i - x_j)$$
$$\text{subject to} \quad x_i \in X \quad \forall i \in V.$$

A variant of the problem with each variable x_i taking an integer value in an interval $[\ell_i, u_i]$ is the separation-deviation problem. We will show that it extends to the problem with $x_i \in X$ for any set of values X. This is important when segmenting an image with a restricted subset of colors.

Since the model of convex-bilinear MRF is the most commonly used in vision, we devote the remainder of this section to providing details on the efficient algorithm that solves this problem.

5.1. Solving the convex deviation bilinear separation MRF

Let ℓ and u be the smallest and largest values respectively in the set of feasible values X, $\ell = \min_{x \in X} x$ and $u = \max_{x \in X} x$. For a parameter value $\alpha \in (\ell, u)$ we construct an s,t-graph $G_\alpha = (V_{st}, A_{st})$ from the adjacency graph of the image (V,A) where V is the set of pixels and A the set of adjacency arcs. For $F_{ij}()$ bilinear we have two opposite arcs between i and j with u_{ij} the arc capacity of arc (i,j) and u_{ji} the arc capacity of arc (j,i).

We add to the set of nodes V a source s and sink t, $V_{st} = V \cup \{s,t\}$. For each $i \in V$ let $G_i'(\alpha)$ be the derivative, or the subgradient, at α of $G_i()$. Each node $i \in V$ has two arcs adjacent to it, one from source and the other one directed to sink. Assign arc (s,i) the capacity $\max\{0, G_i'(\alpha)\}$ and assign arc (i,t) the capacity $\max\{0, -G_i'(\alpha)\}\}$. Notice that one of these two capacities for each i is always zero. Let the set of arcs of positive capacity adjacent to the source be denoted by A_s, and the set of arcs of positive capacity adjacent to the sink, A_t. Let the minimum cut $(\{s\} \cup S, \bar{S} \cup \{t\})$ in the graph G_α partition V to $S = S_\alpha$ and $V \setminus S = \bar{S}_\alpha$. The graph G_α is illustrated in Fig. 3 for a grid graph (V,A). Note however that the algorithm described works for *any* type of graph, rather than for grid graphs only.

Let the optimal solution to MRF be $\mathbf{x}^* = (x_j^*)_j$. The key to the efficiency of the algorithm solving the problem is the following threshold theorem:

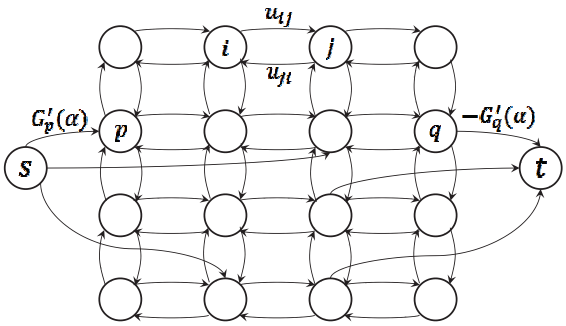

Figure 3: G_α.

Theorem 5.1 (The threshold theorem [26]). *The optimal solution* $\mathbf{x}^*$ *to convex bilinear satisfies* $x_j^* < \alpha$ *for all* $j \in S_\alpha$, *and* $x_j^* \geq \alpha$ *for all* $j \in \bar{S_\alpha}$.

The threshold theorem means that for each node we can determine whether the corresponding variable's value in an optimal solution is $< \alpha$ or $\geq \alpha$, depending on whether the respective node belongs to the source or the sink set of the cut. See Fig. 4 for illustration.

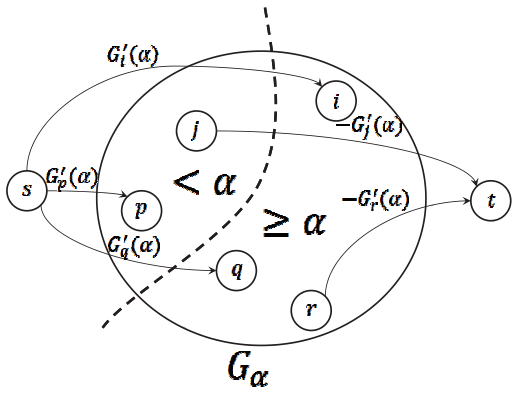

Figure 4: The threshold theorem: The dashed line traverses and intersects the arcs of the cut.

By solving for each value of α in the range, the threshold theorem can be used to establish a partition of the nodes in the graph, and the corresponding variables, to sets where in each set all variables get the same value (and same color) in an optimal solution. So if the set X is of small cardinality, one can apply the minimum cut algorithm $|X|$ times and determine the values of all variables. While this is possible, such an algorithm is not polynomial, e.g., for the case that X can be specified as the integer values in an interval $[\ell, u]$.

We can however solve the problem in polynomial time by noting several important properties of the cut function. Let S_{λ_q} be the *minimal* source set obtained by solving the minimum cut problem in the graph corresponding to parameter $\alpha = \lambda_q$. Then, for a sequence of monotone increasing values of the parameter, $\lambda_0 < \lambda_1 < \cdots < \lambda_p$, we get a

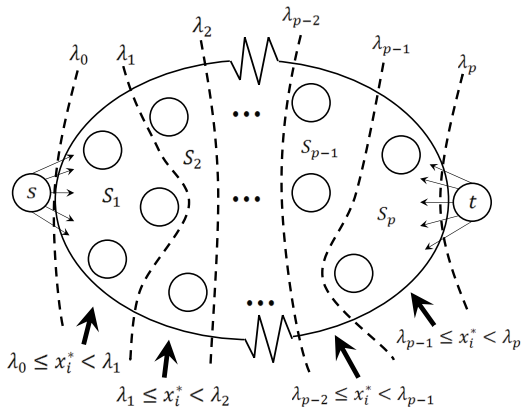

Figure 5: The parametric cut.

nested collection of source sets of the respective minimum cuts: $\{s\} = S_{\lambda_1} \subseteq \cdots \subseteq S_{\lambda_p} \subseteq V$. See Fig. 5 for illustration. When $\lambda_0 \le \ell$ then the set of nodes of value $< \lambda_0$ is empty. For $\lambda_p \ge u$ the set of nodes of value $< \lambda_p$ is V. From the threshold theorem it follows that in an optimal solution all nodes in $S_q = S_{\lambda_q} \setminus S_{\lambda_{q-1}}$, $q = 2, \cdots, p$ have intensity strictly less than λ_q and greater or equal than λ_{q-1}.

The most efficient approach for solving the problem involves using the *parametric cut* algorithm. The parametric algorithm can be implemented with the push-relabel algorithm or HPF. No other algorithms are known that have the required "parametric" structure. This structure, loosely speaking, means that once a minimum cut solution has been found for some value $\alpha = \lambda_1$, it is possible to "continue" the algorithm to find a solution for $\lambda_2 > \lambda_1$ without restarting. This results in run time, for a sequence of increasing K parameter values, that is identical to that of a single cut $T(n,m)$ plus certain adjustment of excesses and deficits in the switch from one parameter value to the next that requires $O(Kn)$ steps. Again, if $K = |X|$ is a quantity greater than polynomial in n and m this would not be considered a polynomial time algorithm. In the next subsection we describe procedure **parametric** that solves the problem in polynomial time regardless of the value of K.

In all vision applications the number of parameter values is small and therefore the run time of the parametric algorithm is dominated by $O(T(n,m))$. We nevertheless include here, for completeness sake, also the algorithm that works in polynomial time regardless of the size of X.

5.2. The parametric cut algorithm

A directed graph G_{st} containing a source and sink nodes s and t is said to be *parametric* if the source-adjacent arcs are functions of a parameter λ that are monotone nondecreasing and the sink-adjacent arcs are functions of λ that are monotone nonincreasing.

Consider varying the value of λ in an interval $[\ell, u]$. As the value of λ increases, the source set becomes larger and contains the previous source sets corresponding to smaller

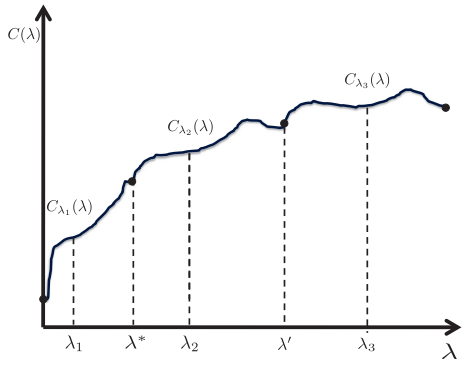

Figure 6: The cut capacity as a function of λ in a parametric cut.

values of λ. Specifically, for some $\lambda \leq \ell$ $S_\lambda = \{s\}$, and for some $\lambda \geq u$ $S_\lambda = V \cup \{s\}$. We call the smallest value of λ where S_λ strictly increases-a *node shifting breakpoint*, or *breakpoint*. For $\lambda_1 < \cdots < \lambda_\ell$ the set of all breakpoints we get a corresponding nested collection of source sets:

$$\{s\} = S_{\lambda_1} \subset S_{\lambda_2} \subset \cdots \subset S_{\lambda_\ell} = \{s\} \cup V.$$

Fig. 6 illustrates how the minimum cut capacity function $C(\lambda)$ changes with λ. In this figure λ^* and λ' are node shifting breakpoints. For any value of λ that lies strictly between any two consecutive breakpoints $\lambda^{(1)} < \lambda^{(2)}$, the function $C(\lambda)$ does not change, since the bi-partition remains the same and it is equal to the function $C_{\lambda^{(1)}}(\lambda)$. In Fig. 6, $C_{\lambda_2}(\lambda) = C_{\lambda^*}(\lambda)$, but $C_{\lambda'}(\lambda)$ is a different function from $C_{\lambda_2}(\lambda)$. The smallest value of λ where the bi-partition changes and at least one node is added to the source set, the cut function changes as well and that value is a breakpoint.

Certain facts concerning the parametric cut functions are of importance. An efficient parametric cut algorithm was first introduced in the seminal paper of Gallo et al. [18]. Although Gallo et al. addressed the parametric cut problem and its implementation with push-relabel algorithm, their discussion is restricted to *linear* parametric functions. This corresponds to deviation functions that are quadratic convex. For general nonlinear monotone parameter functions the following lemmas are critical. We let $C_{\lambda_1}(\lambda)$ be the min cut capacity function of the minimum cut for capacity functions' argument $\lambda = \lambda_1$.

The next lemma establishes that the function $C(\lambda)$ is *breakpoint-concave*. (This term is introduced here for the first time.) To motivate this concept consider the simple case where the parametric capacity functions are linear. In that case all the different cut function, e.g., $C_{\lambda_1}(\lambda)$, are linear. The cut capacity function is then piecewise linear. It is also concave as for $\lambda_1 < \lambda_2$, $C_{\lambda_1}(\lambda) - C_{\lambda_2}(\lambda)$ is a linear function with nonnegative slope. Equivalently, the slope of $C_{\lambda_1}(\lambda)$ is greater or equal to that of $C_{\lambda_2}(\lambda)$. Therefore the function $C(\lambda)$ is concave when the parametric functions are linear. This is clearly not the case for general parametric functions considered here, as between breakpoints the functions are sums of arbitrary monotone increasing (or nondecreasing) and monotone decreasing (or nonincreasing) functions.

Definition 5.1. *A function $C(\lambda)$ is* breakpoint-concave *if for $\lambda_1 < \lambda_2$, $C_{\lambda_1}(\lambda) - C_{\lambda_2}(\lambda)$ is a monotone nondecreasing function of λ.*

Lemma 5.1. *In a parametric graph, the cut capacity function $C(\lambda)$ is breakpoint-concave.*

Proof. By the definition, we need to show that for $\lambda_1 < \lambda_2$, $C_{\lambda_1}(\lambda) - C_{\lambda_2}(\lambda)$ is a monotone nondecreasing function of λ.

Let $(\{s\} \cup S_1, \{t\} \cup T_1)$, $(\{s\} \cup S_2, \{t\} \cup T_2)$ be the cuts corresponding to λ_1 and λ_2 respectively.

$$\begin{aligned} C_{\lambda_1}(\lambda) - C_{\lambda_2}(\lambda) =& C(S_1, T_1) - C(S_2, T_2) + C(\{s\}, T_1)(\lambda) \\ & - C(\{s\}, T_2)(\lambda) + C(S_1, \{t\})(\lambda) - C(S_2, \{t\})(\lambda) \\ =& K_{12} + C(\{s\}, T_1 \setminus T_2)(\lambda) - C(S_2 \setminus S_1, \{t\})(\lambda). \end{aligned}$$

Note that $K_{1,2}$ is a constant independent of λ. $C(\{s\}, T_1 \setminus T_2)(\lambda)$ is a monotone nondecreasing function, and $C(S_2 \setminus S_1, \{t\})(\lambda)$ is a monotone nonincreasing function. Therefore the difference between these two terms is monotone nondecreasing. □

Consider the interval $[\lambda_1, \lambda_2]$, the respective cut functions $C_{\lambda_1}(\lambda)$ and $C_{\lambda_2}(\lambda)$ and the value of λ at the intersection of these function, $\hat{\lambda}$. As a corollary to Lemma 5.1 we get that either:

1. the interval $[\lambda_1, \lambda_2]$ contains no breakpoints, and then the functions $C_{\lambda_1}(\lambda)$, $C_{\lambda_2}(\lambda)$ are identical, or,
2. there is exactly one breakpoint in the interval $[\lambda_1, \lambda_2]$ and this breakpoint is $\hat{\lambda}$ — the intersection of the two functions $C_{\lambda_1}(\hat{\lambda}) = C_{\lambda_2}(\hat{\lambda})$, or,
3. there are two or more breakpoints in the interval $[\lambda_1, \lambda_2]$, then $\hat{\lambda}$ "separates" them in the sense that $[\hat{\lambda}, \lambda_2]$ and $[\lambda_1, \hat{\lambda}]$ each contain at least one breakpoint.

In the parametric cut algorithm the step of taking the intersection of two cut functions is repeated recursively. With each such call, either a breakpoint is found, or it is proved that there is no breakpoint, or else there is a partition to two intervals each containing at least one breakpoint.

Since the cut functions appear to be arbitrary, it is not obvious that finding the intersection of two cut functions can be accomplished easily or efficiently. However, we prove next that this step is no harder than finding the minimum of a convex function in a given interval.

Lemma 5.2. *The complexity of finding $\hat{\lambda}$ is the same as the complexity of finding that the sum of derivatives of convex functions is equal to a constant K (or finding the minimum of a convex function.)*

Proof. Since the capacities of source adjacent arcs are monotone nondecreasing, there exists, for each $j \in V$ a (nondecreasing) convex function $f_j()$ whose derivative $\max\{0, f_j'(\lambda)\}$

coincides with the parametric capacity function on arc (s, j). Similarly, the capacity functions of the sink adjacent arcs, (s, i), are monotone nonincreasing and therefore these coincide with the derivatives of (nonincreasing) convex functions $\min\{0, f_i'(\lambda)\}$.

Let $(\{s\} \cup S_1, \{t\} \cup T_1)$, $(\{s\} \cup S_2, \{t\} \cup T_2)$ be the cuts corresponding to λ_1 and λ_2 respectively, as in the proof to Lemma 5.1. Using that proof we write the intersection of the two functions λ as satisfying,

$$\begin{aligned} 0 =& C_{\lambda_1}(\lambda) - C_{\lambda_2}(\lambda) \\ =& K_{1,2} + C(\{s\}, T_1 \setminus T_2)(\lambda) - C(S_2 \setminus S_1, \{t\})(\lambda) \\ =& K_{1,2} + \sum_{j \in T_1 \setminus T_2} \max\{0, f_j'(\lambda)\} + \sum_{i \in S_2 \setminus S_1} \min\{0, f_i'(\lambda)\}. \end{aligned}$$

Observe that $T_1 \setminus T_2 = S_2 \setminus S_1$ therefore this sum is $K_{1,2} + \sum_{i \in S_2 \setminus S_1} f_i'(\lambda)$. Thus the argument $\hat{\lambda}$ that solves this equation is also the minimum argument of the sum of convex functions, $\sum_{i \in S_2 \setminus S_1} f_i(\lambda) + K_{1,2}\lambda$. Since sum of convex functions is convex this is equivalent to finding the minimum of a convex function. We note that the minimum (or maximum) of a convex (concave) function can be found, within accuracy ϵ, in time $\log U/\epsilon)$. This time cannot be improved in the sense that the complexity must depend on U/ϵ (For an extensive discussion of this subject, the reader is referred to [27]). □

The recursive procedure **parametric** finds all the breakpoints:

Procedure parametric ($\lambda_1, \lambda_2, S_{\lambda_1}, S_{\lambda_2}$)

Find the intersection $\hat{\lambda}$ of C_{λ_1} and C_{λ_2}, $C_{\lambda_1}(\hat{\lambda}) = C_{\lambda_2}(\hat{\lambda})$.
If $S_{\lambda_1} = S_{\hat{\lambda}}$, halt "no breakpoints in the interval, and λ_1 is a breakpoint".
Else
    Call **parametric** ($\lambda_1, \hat{\lambda}, S_{\lambda_1}, S_{\hat{\lambda}}$)
    Call **parametric** ($\hat{\lambda}, \lambda_2, S_{\hat{\lambda}}, S_{\lambda_2}$)
end

The complexity of **parametric** is $O(T(n, m) + n \log U/\epsilon)$, where U is the length of the interval between the smallest value of λ, where $S_\lambda = \{s\}$, and the largest value of λ, where $S_\lambda = \{s\} \cup V$. As proved in [26], this complexity is best possible.

For a set of breakpoints $\{\lambda_1 < \cdots < \lambda_\ell\}$, the bi-partition, and therefore the source set, does not change for any $\alpha \in [\lambda_{q-1}, \lambda_q)$, $q = 2, \cdots, \ell$. We conclude that for all $j \in S_q = S_{\lambda_q} \setminus S_{\lambda_{q-1}}$ x_j^* is equal to the largest value in X that is $< \lambda_q$. This is because for any λ in the interval the set $S_\lambda = S_{\lambda_{q-1}}$. Thus all nodes in the set are greater or equal to λ and strictly smaller than λ_q.

Example 5.1. Let $\lambda_1 < \lambda_2 < \lambda_3 < \lambda_4$ be four breakpoints so that $S_{\lambda_1} = \emptyset$, and let $x_1 < x_2 < x_3 < x_4 < x_5$ be the values in the set X so that, $\lambda_1 < \lambda_2 < x_1 < x_2 < \lambda_3 \leq x_3 < x_4 < x_5 < \lambda_4$. Given the source sets for the breakpoints, $\emptyset \subset S_{\lambda_2} \subset S_{\lambda_3} \subset S_{\lambda_4}$ we conclude

that $S_{x_1} = S_{x_2} = S_{\lambda_2} \subset S_{x_3} = S_{x_4} = S_{x_5} = S_{\lambda_3}$. Then, for all $i \in S_{\lambda_2}$, $x_i^* = x_2$ and for all $i \in S_{\lambda_3} \setminus S_{\lambda_2}$, $x_i^* = x_5$.

Therefore, once the set of breakpoints is available we can quickly generate the optimal solution for any discrete set X. Assuming that for any λ, the value $\max_{x_i \in X, x_i \leq \lambda} x_i$ can be found in $O(1)$, we get the following lemma:

Lemma 5.3. *Given the set of all breakpoints $\lambda_0 < \lambda_1 < \lambda_2 \cdots < \lambda_p$, the optimal solution to convex-bilinear MRF restricted to any set of discrete values X is generated in linear time.*

6. The interactive tool

Due to its speed, the convex-bilinear MRF algorithm can be applied in an interactive online mode. This is particularly suitable for applications where one is looking for hidden pathologies of specific type in noisy images. This is the case when searching for tumors or lesions in medical images, or in anti-terrorism or crime prevention applications. The MRF algorithm of [26] was implemented within a framework of an "imaging tool" with a user interface. The interactive tool's interface is shown in Fig. 7. It is implemented with several specific features.

The interactive tool solves the separation-deviation (SD) problem:

$$\text{(SD)} \min D \sum_{j \in V} G_j(r_j - x_j) + S \sum_{(i,j) \in A} u_{ij} |x_i - x_j|$$
$$\text{subject to} \quad x_i \in X \quad \text{for} \quad i \in V.$$

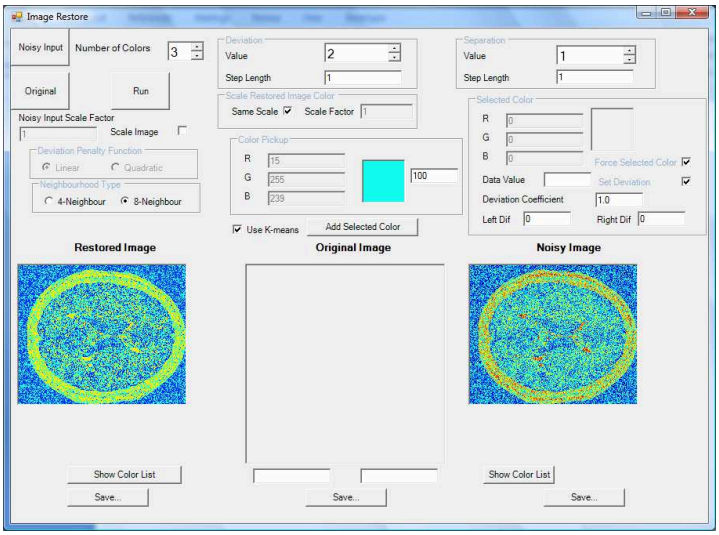

Figure 7: The interactive tool.

The functions G_j can be either quadratic or absolute value functions. D and S are constant integers multiplying the deviation and separation terms respectively. If $D = 0$ then the output is a single color label assigned to all nodes. If $S = 0$ then the output is the same as the input, if the colors set of the input is in X, otherwise each pixel is assigned a "nearest" color label in X.

The specific values of S and D are unimportant. Rather, it is the *ratio* S/D which is crucial in determining the degree of color uniformity in the image. The larger this ratio, the greater the color uniformity.

The empirical implementation of the MRF algorithm is using the parametric HPF code accessible for download at [12].

The current algorithm's interface can support the following interactive functions:

1. Segment image with a fixed number of automatically selected colors. For a chosen number of colors k the tool uses a k-means algorithm to select the color set X. Note that the number of colors is not equal to the number of segments as each color set is not necessarily a connected component.
2. Segment for a input color set X. Alternatively, the user can add or remove colors to an existing color set by clicking on any pixel in the image, or manually inserting the color code. As proved in Lemma 5.3 the output image can be generated by reading the existing output and without additional computation.
3. Uniform increase/decrease in deviation cost. The coefficient of all the deviation terms is denoted by D. The tool allows to increase all deviation costs by a selected constant factor.
4. Similarly, the tool allows to increase/decrease all separation costs by a selected constant factor, denoted by S. We note that the only parameter of interest is the *ratio*, D/S between the deviation and separation penalty terms. The tool allows to modify the coefficient of each by an integer factor. Consequently any rational number ratio can be generated.
5. Color restricted change in deviation. This feature is important in identifying hidden structures. When the user suspects that a certain color area may indicate an object of interest, it is possible to increase the deviation functions associated with this color only. So for selected color g the deviation function $G_i(x_i, g)$ is increased by the selected factor for all pixels with input color g. This guarantees that any pixel with input color equal to g is more likely to show in the segmented output, even though it is small and has unusual boundaries that otherwise would have been "cleaned" by the separation penalty dominance. Thus an object of this or similar color, is likely to be identified.
6. The color restricted change in deviation may lead to the pronounced appearance of this color also in areas where it is not of significance. For that purpose the tool allows to restrict the change in deviation or separation to a user defined rectangular window region.
7. Choose an 8-neighbor adjacency or a 4-neighbor adjacency.

7. Experimental results

In order to assess the performance of the algorithm we use a simulation on medical images of thorax and brain: For a "true" image, noise is randomly added to obscure the

image and simulate an input "noisy" image. The objective, when processing true images, is to segment correctly the salient features in the image. These features are determined as per their clinical significance. For noisy images, the objective is to de-blur them without losing the salient features of the image.

7.1. The effect of changing the number of colors in the output

Here we demonstrate the feature of choosing a fixed number of colors, k, from the interval of colors of the input noisy image. The set of k colors is selected automatically by using k-means algorithm that chooses k numbers in the range of input colors $[0,1,2,\cdots,255]$ that minimize the total distance on the real line from all the pixels colors. This choice may generate colors that do not even exist in the input image, but are nevertheless among the optimal median set.

This is illustrated here in an image of the thorax. The true and noisy images are presented in Fig. 8. In Fig. 9 we show the results of the segmentation with linear absolute value functions for both the deviation and separation penalties. Note that for the choices of $k = 5$ and $k = 6$ the lesion color has been selected and the lesion appears as one of the segments in the output image.

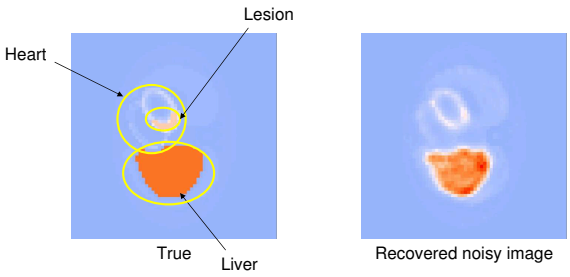

Figure 8: The "true" and noisy thorax images.

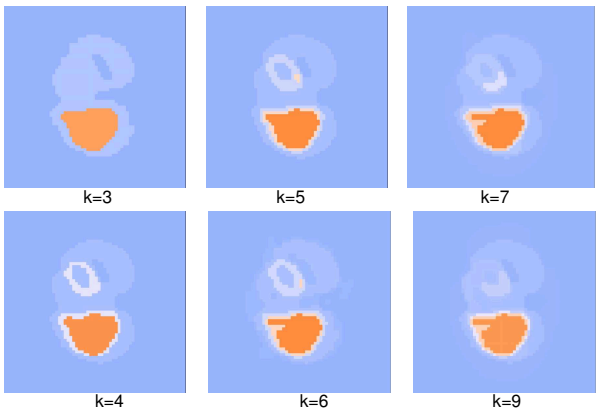

Figure 9: The output for different values of k — number of colors — selected automatically by a k-means algorithm.

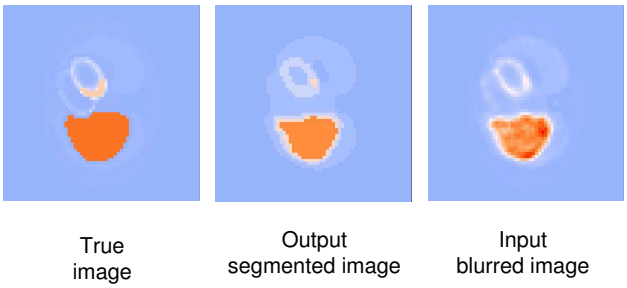

Figure 10: Thorax input, output and true comparison.

To demonstrate the segmented image appearance of the lesion compared to the noisy and true, we give them side-by-side in Fig. 10.

7.2. Modifying the ratio between the separation and deviation penalties

The effect of modifying the ratio S/D is illustrated here for two examples of brain images. The first set of true and noisy images are given in Fig. 11. In that image there are four small lesions. We then apply the separation-deviation algorithm with $D = 2$ and for increasing values of S, as shown in Fig. 12. The lesions show very clearly in the high separation images in yellow color.

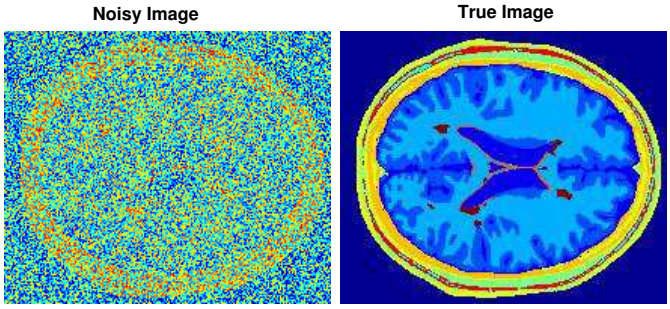

Figure 11: Brain image 1, true and noisy.

The second set is similar, with an elongated form of lesion in the brain image. The true and noisy are shown in Fig. 13.

Again the high separation coefficient images in Fig. 14 clearly segment the elongated lesion tissue.

7.3. Increasing deviation for a selected color

The tool allows to select a particular color, either by the color code, or by clicking on a pixel that has the desired color. The deviation penalty is then increased for all integer

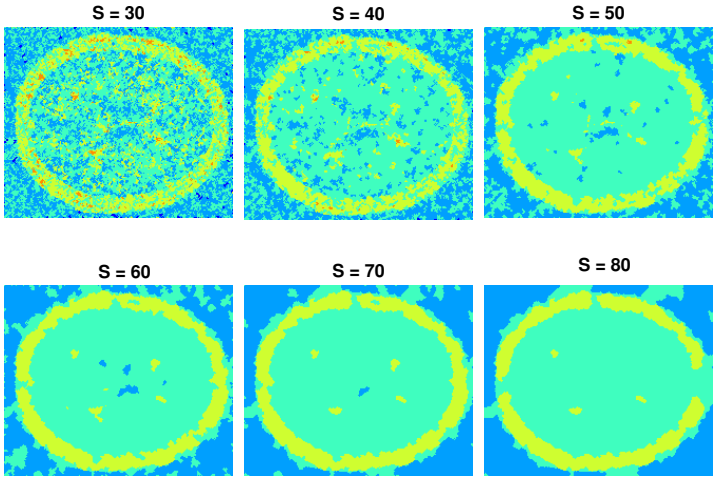

Figure 12: The output for increasing values of S when applied to noisy brain image 1.

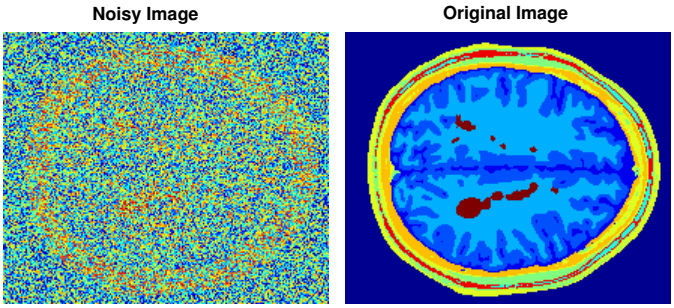

Figure 13: Brain image 2, true and noisy.

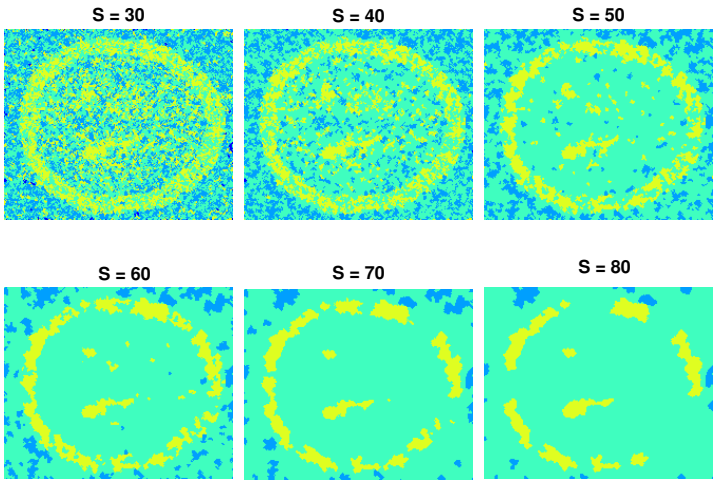

Figure 14: The output for increasing values of S when applied to noisy brain image 2.

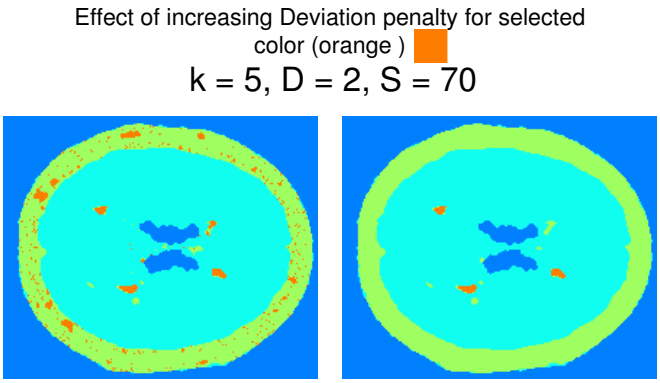

Figure 15: Increased deviation penalty for a selected color in brain image 1.

color codes in a small interval around the selected color. For color code q the interval is $[q-5, q+5]$. The size of this interval can be adjusted by the user.

We show here, for brain image 1, that if the color orange is selected, then it show as the color of 3 out of the 4 lesions. When the deviation for that color is increased the lesions become better segmented and more prominent. Of course, the color orange also appears in other areas of the brain shell where it is of no clinical significance. This issue will be addressed in the next prototype of the interactive tool, where the deviation increase will apply only in a user-defined window.

7.4. Comparison of image segmentation with separation-deviation to normalized cuts approach

We now compare our software for image segmentation with the normalized cut approach introduced by Shi and Malik [37]. This normalized cut approach utilizes the spectral technique in finding the Fiedler eigenvalue and the corresponding eigenvector. The method is described and Shi's software implementation is provided in: `http://www.cis.upenn.edu/~jshi/software/`.

The input to that code is the number of desired segments in the output image. The code preprocesses the input image, first by converting it to gray scale and then resizing it to 160×160. The algorithm is then applied to the preprocessed image. We show here the segmentation of the thorax image, for 5 segments and for 12 segments. The running time increases approximately linearly with the number of segments. It appears that a much larger number of segments will be required before the lesion will be segmented.

Similar results apply to brain image 2 where the true image is segmented. This is shown in Fig. 17. Only the 20 segments partition begins to show the lesion area, yet still this segmentation does not delineate the lesion correctly.

Note that the software of Shi requires to convert the image first to gray scale, which is why it is not presented in color.

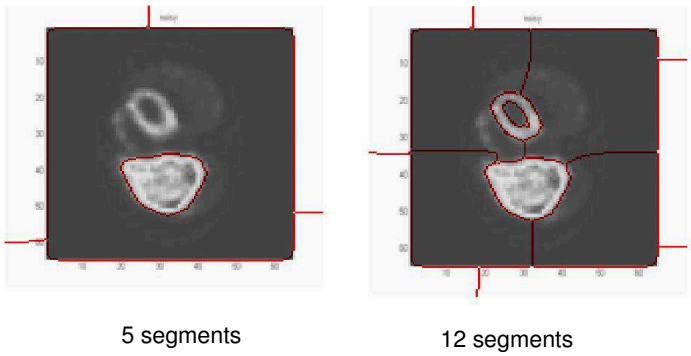

Figure 16: Normalized cut software segmentation of the noisy thorax image for 5 and 12 segments.

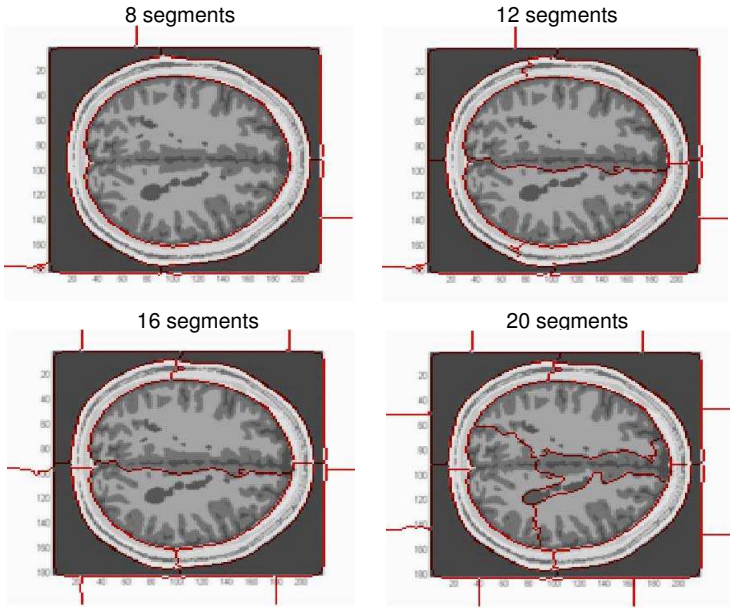

Figure 17: Normalized cut software segmentation of true brain image 2 for 8, 12, 16 and 20 segments.

8. Conclusions

We demonstrate here that the MRF algorithm is an effective technique for regularization and denoising of images, that has not been utilized to date to its full potential. The advantage of the algorithm is both in theory and in practice. Since the MRF algorithm delivers an optimal solution, and is provably fastest possible, it gives better quality results than any alternative methodology, in terms of minimizing the objective function. This is in contrast to using heuristics, where the solution generated is not directly linked to a particular optimization criterion. The algorithm is shown here to segment successfully in

practice the salient features in true images, and to be able to identify hidden important features and de-blur noisy images. These capabilities together with the efficiency of the MRF algorithm, renders it a useful addition to a segmentation tool box.

Acknowledgments This research was supported in part by NSF awards No. CMMI-1200592 and CBET-0736232.

References

[1] R. K. Ahuja, D. S. Hochbaum and J. B. Orlin, *A cut-based algorithm for the convex dual of the minimum cost network flow problem*, Algorithmica, 39(3) (2004) pp. 189–208. Also, UC Berkeley manuscript, (1999).

[2] R. K. Ahuja, D. S. Hochbaum and J. B. Orlin, *Solving the convex cost integer dual network flow problem*, Management Science, 49(7) (2003), pp. 950–964. Extended abstract in, Proceedings of IPCO'99, G. Cornuejols, R. E. Burkard and G. J. Woeginger (Eds.), Lecture Notes in Computer Science, 1610 (1999), pp. 31–34.

[3] E. Bae and X.-C. Tai, *Graph cut optimization for the piecewise constant level set method applied to multiphase image segmentation*, Scale Space and Variational Methods in Computer Vision (SSVM 2009), X.-C. Tai and K. Mórken and M. Lysaker and K.-A. Lie eds. LNCS, 5567 (2009), pp. 1–13.

[4] A. Blake and A. Zisserman, Visual Reconstruction, MIT Press, 1987.

[5] Y. Boykov and M. P. Jolly, *Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images*, International Conference on Computer Vision, (ICCV), I (2001), pp. 105–112.

[6] Y. Boykov and V. Kolmogorov, *An experimental comparison of min- cut/max-flow algorithms for energy minimization in vision*, IEEE Trans. Pattern Anal. Machine Intelligence, (PAMI), 26(9) (2004), pp. 1124–1113.

[7] Y. Boykov, O. Veksler and R. Zabih, *Markov random fields with efficient approximations*, Proc IEEE Conference CVPR, Santa Barbara CA, (1998), pp. 648–655.

[8] Y. Boykov, O. Veksler and R. Zabih, *Fast approximate energy minimization via graph cuts*, Proc 7th IEEE International Conference on Computer Vision, (1999), pp. 377–384.

[9] T. F. Chan and L. A. Vese, *Active contours without edges*, IEEE Trans. Image Processing, 10(2) (2001), pp. 266–277.

[10] T. F. Chan and S. Esedoglu, *Aspects of total variation regularized l1 function approximation*, SIAM J. Appl. Math., 65(5) (2005), pp. 1817–1837.

[11] B. G. Chandran and D. S. Hochbaum, *A computational study of the pseudoflow and push-relabel algorithms for the maximum flow Problem*, Operations Research, 57(2) (2009), pp. 358–376.

[12] B. G. Chandran and D. S. Hochbaum, Pseudoflow Solver, accessed January 2012, `http://riot.ieor.berkeley.edu/Applications/Pseudoflow/maxflow.html`.

[13] D. L. Collins, A. P. Zijdenbos, V. Kollokian, J. G. Sled, N. J. Kabani, C. J. Holmes and A. C. Evans, *Design and construction of a realistic digital brain phantom*, IEEE Trans. Med. Imaging, 17(3) (1998), pp. 463–468.

[14] I. J. Cox, S. B. Rao and Y. Zhong, *Ratio regions: a technique for image segmentation*, Proc. Int. Conf. Pattern Recognition, B (1996), pp. 557–564.

[15] J. Darbon and M. Sigelle, *Image Restoration with discrete constrained total variation Part I: Fast and exact optimization*, J. Math. Imaging Vis., 26(3) (2006), pp. 261–276.

[16] B. Fishbain D. S. Hochbaum and S. Mueller, *The Pseudoflow algorithm for minimum-Cut in vision problems*, `http://arxiv.org/abs/1007.4531` and UC Berkeley manuscript, 2011.

[17] L. R. Ford and D. R. Fulkerson, *Maximal flow through a network*, Canadian J. Math., 8(3) (1956), pp. 339–404.

[18] G. Gallo, M. D. Grigoriadis and R. E. Tarjan, *A fast parametric maximum flow algorithm and applications*, SIAM J. Comput., 18 (1989), pp. 30–55.

[19] D. Geiger and F. Girosi, *Parallel and deterministic algorithms for MRFs: surface reconstruction*, IEEE Trans. Pattern Anal. Machine Interlligence, PAMI, 13 (1991), pp. 401–412.

[20] S. Geman and D. Geman, *Stochastic relaxation, Gibbs distributions and the bayesian restoration of images*, IEEE Trans. Pattern Anal. Machine Intelligence, PAMI, 6 (1984), pp. 721–741.

[21] A. Goldberg, *The partial augmentÜelabel algorithm for the maximum flow problem*, ESA, (2008), pp. 466–477.

[22] D. M. Greig and B. T. Porteous and A. H. Seheult, *Exact maximum a posteriori estimation for binary images*, J. Royal Statis. Society Ser. B, 51(2) (1989), pp. 271–279.

[23] D. S. Hochbaum and J. B. Orlin, *Simplifications and speedups of the pseudoflow algorithm*, Networks, to appear 2012. Online in `http://onlinelibrary.wiley.com/doi/10.1002/net.21467/abstract`.

[24] D. S. Hochbaum, J. Qranfal and G. Tanoh, *A fast computational algorithm for segmentation of noisy medical images*, Algorithmic Operations Res., 6 (2011), pp. 79–90.

[25] D. S. Hochbaum, *The Pseudoflow algorithm: a new algorithm for the maximum flow problem*, Operations Res., 58(4) (2008), pp. 992–1009.

[26] D. S. Hochbaum, *An efficient algorithm for image segmentation, Markov Random Fields and related problems*, J. ACM, 48(4) (2001), pp. 686–701.

[27] D. S. Hochbaum, *Complexity and algorithms for nonlinear optimization problems*, Annal. Operations Res., 153 (2007), pp. 257–296.

[28] H. Ishikawa and D. Geiger, *Segmentation by grouping junctions*, IEEE Conference on Computer Vision and Pattern Recognition CVPR98, (1998), pp. 125–131.

[29] H. Ishikawa, *Exact optimization for Markov Random fields with convex priors*, IEEE Trans. Pattern Anal. Machine Intelligence, 25(10) (2003), pp. 1333–1336.

[30] V. Kolmogorov and R. Zabih, *What energy functions can be minimized via graph cuts*, IEEE Trans. Pattern Anal. Machine Intelligence, 26(2) (2004), pp. 147–159.

[31] S. Z. Li, K. L. Chan and H. Wang, *Bayesian image restoration and segmentation by constrained optimization*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96), (1996).

[32] J. Malik, S. Belongie, T. Leung and J. Shi, *Contour and texture analysis for image segmentation*, Int. J. Comput. Vision, 43 (2001), pp. 7–27.

[33] S. J. Osher and R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer New York, 2003.

[34] T. Pock, A. Chambolle, D. Cremers and H. Bischof, *A convex relaxation approach for computing minimal partitions*, IEEE Computer Society Conference on Computer Vision and Pattern

Recognition (CVPR'09), (2009) pp. 810–817.

[35] P. H. PRETORIUS, M. A. KING, B. M. W. TSUI, K. J. LACROIX AND W. XIA, *A mathematical model of motion of the heart for use in generating source and attenuation maps for simulating emission imaging*, Med. Phys., 26 (1999), pp. 2323–2332.

[36] L. I. RUDIN, S. J. OSHER AND E. FATEMI, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.

[37] J. SHI AND J. MALIK, *Normalized cuts and image segmentation*, IEEE Trans. Pattern Anal. Mach. Intell., 22(8) (2000), pp. 888–905.