Fusion Moves for Markov Random Field Optimization

Victor Lempitsky Carsten Rother Stefan Roth Andrew Blake

Abstract—The efficient application of graph cuts to Markov Random Fields (MRFs) with multiple discrete or continuous labels remains an open question. In this paper, we demonstrate one possible way of achieving this by using graph cuts to combine pairs of suboptimal labelings or solutions. We call this combination process the fusion move. By employing recently developed graph cut based algorithms (so-called QPBO-graph cut), the fusion move can efficiently combine two proposal labelings in a theoretically sound way, which is in practice often globally optimal.

We demonstrate that fusion moves generalize many previous graph cut approaches, which allows them to be used as building block within a broader variety of optimization schemes than were considered before. In particular, we propose new optimization schemes for computer vision MRFs with applications to image restoration, stereo, and optical flow, among others. Within these schemes the fusion moves are used 1) for the parallelization of MRF optimization into several threads; 2) for fast MRF optimization by combining cheap-to-compute solutions; and 3) for the optimization of highly non-convex continuous-labeled MRFs with 2D labels. Our final example is a non-vision MRF concerned with cartographic label placement, where fusion moves can be used to improve the performance of a standard inference method (loopy belief propagation).

Index Terms—Markov random fields, Computer vision, Combinatorial algorithms, Graph algorithms, Stereo, Motion, Image restoration.

1 Introduction

Many computer vision and image processing tasks are cast as optimization problems. Quite often these optimization problems arise from the need to perform maximum a-posteriori (MAP) estimation in probabilistic models. The minimization of energies associated with Markov Random Fields (MRF energies) has enjoyed particular popularity, especially in low-level vision applications. These include stereo matching, image denoising, image inpainting, image segmentation, image super-resolution and others.

Over recent decades, a number of optimization algorithms have been suggested for MRF energies that commonly arise in (low-level) vision problems [1]. Graph cut approaches, which minimize pairwise MRF energies by solving mincut/maxflow problems on special graph constructions are among the most successful methods. Graph cut methods naturally apply to binary-labeled pairwise MRFs, for which they can often find globally optimal solutions [2]–[5].

This paper focuses on minimizing energies from pairwise MRFs with either multiple (> 2) discrete or continuous labels. The majority of optimization problems for such MRFs are NP-hard, and thus various approximate algorithms (such as loopy belief propagation [6] in case of discrete labels, and gradient descent in case of continuous labels) have been suggested. Graph cuts have also been applied to discrete multi-label MRFs by either been reducing them to large binary-label MRFs [7]–[9], or by applying graph cuts iteratively to binary-labeled subproblems [10]–[13]. While these approaches yield competitive results when compared to other approximate optimization techniques, their computational demands grow fast with the number of labels.

Here we take a different approach to these problems: we exploit the fact that a variety of approximate solutions to a particular MRF optimization problem can be produced by using various approximate (or local) optimization algorithms, different initializations, parameter settings or other factors. The question we ask is whether different suboptimal solutions can be combined in a principled manner in order to produce a new, better labeling with lower energy.

We show that given a pair of suboptimal labelings (or *proposals*), the problem of their combination into a better solution can be expression as a binary-labeled MRF minimization problem. This auxiliary binary-labeled MRF can be optimized efficiently using graph cuts, which yields a new combined (*'fused'*) labeling with decreased (or equal) energy. We call this operation the *fusion move* and, after introducing it, suggest several optimization schemes based on fusion moves for various pairwise MRFs. We also discuss how the fusion move generalizes several previous graph cut algorithms for multi-label discrete MRFs [10], [11]. In general, fusion moves allow us to apply graph cuts to a wider class of MRFs, including those with continuous labels. Moreover, in the case of discrete labels, fusion moves can be used to accelerate graph cut minimization dramatically.

Our approach builds on two important components from the literature: First, we rely on the QPBO-mincut procedure from [4], introduced to computer vision in [14]. It can optimize non-submodular binary-labeled MRFs that are subproblems of our approach, and we refer to this procedure as non-submodular graph cut.

Second, we use the idea of graph cut moves from [10], [11] and extend it to perform fusions of proposal solutions that themselves are computed by some domain-specific method.

We demonstrate the versatility and applicability of fusion moves with four different *hybrid* optimization schemes, where fusion moves are used to combine proposal solutions coming from very different processes. First, we demonstrate how fusion moves can be used to efficiently parallelize previous graph cut-based algorithms for discrete-label MRFs — the well known alpha-expansion [10] algorithm — with little parallelization overhead. In this case, the proposal labelings that are to be fused are generated by the alpha-expansion algorithm itself. Next, we suggest a new graph cut algorithm called *LogCut* that generates labelings for typical low-level vision MRFs with a large number of labels much faster than alpha-expansion. We observe that by fusing several suboptimal labelings produced by LogCut using fusion moves, we can find MRF labelings with very low energy much faster than state-of-the-art algorithms.

After that, we apply fusion moves to a continuous-valued MRF for optical flow computation, and observe that fusing proposals obtained with classical continuous algorithms can yield optical flow fields with state-of-the-art accuracy, which by far exceeds that of the original proposals (*FusionFlow* approach). The LogCut and FusionFlow approaches were previously introduced in two conference papers [15], [16]. Here we focus on the aspects of these approaches that relate to fusion moves.

Finally, we consider a class of non-vision MRFs found in geo-information systems (GIS) applications that are concerned with automatic label placements on a map. Such MRFs have a very different structure from the ones considered in the previous applications, and the optimization schemes popular in computer vision, such as alpha-expansion or message passing do not perform well. Here, we suggest an optimization scheme that obtains low-energy solution by fusing proposals obtained with message-passing (belief propagation).

The rest of the paper is organized as follows. In section 2 we introduce our notation, then review previous work on optimization in binary-labeled MRFs, which is essential to our approach; after that we introduce the fusion move idea, and finally relate it to other graph-cut based multi-label MRF optimization methods. In section 3, we discuss four new hybrid optimization schemes developed on the basis of fusion moves and give experimental results. Finally, we conclude with a discussion in section 4.

2 The Fusion Move

Before introducing the fusion move itself, we first introduce the basic setting. In this paper we consider energies associated with pairwise MRFs, which take the form:

$$E(\mathbf{x}) = \sum_{p\in\mathcal{V}} U_p(x_p) + \sum_{p,q\in\mathcal{N}} V_{pq}(x_p, x_q)\,, \quad \mathbf{x}\in\mathbf{L}^{\mathcal{V}}\,, \quad (1)$$

where $\mathcal{V}$ is a set of nodes, $\mathcal{N}$ is a set of undirected edges connecting pairs of nodes (in vision problems $\mathcal{V}$ often corresponds to the pixels in the image, while $\mathcal{N}$ contains pairs of adjacent pixels in 4- or 8-connected neighborhoods). The *labeling* $\mathbf{x}$ assigns a label x_p from some label space $\mathbf{L}$ to every node $p\in\mathcal{V}$. The family of real-valued functions $U_p : \mathbf{L}\to\mathbf{R}$ and $V_{p,q} : \mathbf{L}^2\to\mathbf{R}$, called *unary* and *pairwise potentials* respectively, define the energy of each labeling. The optimization problem then is to find the labeling with the smallest energy, which amounts to finding the labeling with the highest posterior probability, since energy and probability can be related through Boltzmann's law as $p(\mathbf{x}) = \exp\{-\frac{1}{T}E(\mathbf{x})\}$. Since for the majority of interesting combinations of $\mathbf{L}$, U_p, and V_{pq} this minimization problem is NP-complete, approximate optimization approaches are considered instead.

2.1 Previous work on binary-labeled MRFs

The minimization of (1) in the binary-label case ($\mathbf{L} = \{0,1\}$) has a number of applications in computer vision, such as image segmentation, image stitching, etc. and has thus received a lot of attention in the literature (e.g. [3], [5]). These techniques has long been considered in the optimization literature as well, where such minimization problems are known as *quadratic pseudo-boolean programming* (QPBO). It is known that certain QPBO problems can be solved via minimum cut computation in a specially constructed graph [2]. In the *submodular* case when all pairwise potentials V_{pq} obey $V_{pq}(0,0) + V_{pq}(1,1) \le V_{pq}(0,1) + V_{pq}(1,0)$ the globally optimal labeling $\hat{\mathbf{x}}$ can be found exactly [2], [3], [5].

It has been realized, however, relatively recently that non-submodular QPBO problems can still be reduced to minimum cut computations [4], [14]. In the general non-submodular case, the globally optimal labeling $\hat{\mathbf{x}}$ can be found only partially, however. This means that after the minimum cut computation, each node can be assigned the labels 0, 1, or '?' (we will refer to the nodes with labels 0 and 1 as *labeled* while the nodes with label '?' are called *unlabeled*). It is then known that the globally optimal labeling coincides with $\hat{\mathbf{x}}$ for all labeled nodes (*partial optimality*). Furthermore, it is known that taking an arbitrary labeling $\mathbf{x}'$ and replacing its labels with those of $\mathbf{x}$ for all nodes that have been labeled in $\hat{\mathbf{x}}$ is guaranteed to not increase the energy of $\mathbf{x}'$ (*persistence* property). In many practical scenarios, the number of unlabeled nodes may be negligible; in many other cases some or all of such nodes can be further labeled in a globally optimal way using certain search heuristics [17]–[19].

2.2 The fusion move

We now consider a more general case, where the MRF energy (1) is defined on a non-binary label space $\mathbf{L}$ (e.g. $\mathbf{L} = \{0,1,...,N\}$ or $\mathbf{L} = \mathbf{R}^2$). While the exact minimization of (1) is intractable for the absolute majority of cases, its approximate minimization is still of great use for

a broad variety of computational problems. The fusion move introduced here deals with the problem of optimally combining two (suboptimal) proposal labelings. As we shall see later, this methodology can be applied to the approximate minimization of (1) in a variety of ways.

Let us assume that we are given two labelings $\mathbf{x}^0 \in \mathbf{L}^{\mathcal{V}}$ and $\mathbf{x}^1 \in \mathbf{L}^{\mathcal{V}}$. We will now consider the set of labelings obtained by combining $\mathbf{x}^0$ and $\mathbf{x}^1$. Here, by combination we understand a labeling where the label of each node is taken either from $\mathbf{x}^0$ or from $\mathbf{x}^1$. More formally, a combination $\mathbf{x}^c$ is defined by an auxiliary binary vector $\mathbf{y} \in \{0,1\}^{\mathcal{V}}$, such that:

$$\mathbf{x}^c(\mathbf{y}) = \mathbf{x}^0 \bullet (1-\mathbf{y}) + \mathbf{x}^1 \bullet \mathbf{y}\ , \tag{2}$$

where $\bullet$ denotes the Hadamard (node- or element-wise) product, i.e. $x^c_p(\mathbf{y}) = x^0_p$ if $y_p = 0$ and $x^c_p(\mathbf{y}) = x^1_p$ if $y_p = 1$.

Then the energy (1) of any such combination if defined as the energy of the auxiliary vector:

$$E^f(\mathbf{y}) = E(\mathbf{x}^c(\mathbf{y})) = \sum_{p\in\mathcal{V}} U^f_p(y_p) + \sum_{p,q\in\mathcal{N}} V^f_{pq}(y_p, y_q)\,, \tag{3}$$

where new auxiliary unary and pairwise potentials are defined as:

$$U^f_p(i) = U_p(x^i_p)\,, \quad V^f_{pq}(i,j) = V_{pq}(x^i_p, x^j_q)\,. \tag{4}$$

Minimizing equation (3) as a function of the binary labeling $\mathbf{y}$ using non-submodular graph cuts yields the labeling $\hat{\mathbf{y}}$. If all the nodes of $\mathbf{y}$ are labeled, then this labeling corresponds to the global optimum of the problem (3) and, therefore, the resulting labeling $\mathbf{x}^c(\hat{\mathbf{y}})$ corresponds to the globally optimal combination of $\mathbf{x}^0$ and $\mathbf{x}^1$ in the sense of the original problem (1). We call such a combination a *fusion* of $\mathbf{x}^0$ and $\mathbf{x}^1$, and the process of its computation the *fusion move*.

In some cases, solving the problem (3) using minimum cut may only yield a partial optimal labeling $\hat{\mathbf{y}}$ that contains some unlabeled nodes. In practice, however, we have found that for a variety of fusion problems only very few of the nodes were unlabeled when computing the fusion move (e.g. just a few pixels in the image, or 0.1% of nodes — the exact numbers are reported together with each experiment). Before moving on to the relationship of the fusion move to previous approaches, we will discuss an informal intuition why this number was so insignificant and after that propose a very simple, yet theoretically sound way of dealing with unlabeled nodes.

The number of unlabeled nodes in the partially global optimal labeling for a pairwise MRF with binary labels is closely related to the number of non-submodular pairwise terms, which are violating the constraint $V_{pq}(0,0) + V_{pq}(1,1) \leq V_{pq}(0,1) + V_{pq}(1,0)$ [20]. In the fusion move case, this constraint means that taking the labels of two adjacent nodes from the same proposal should on average have smaller pairwise cost than taking them from different proposals. But this is exactly what typically happens in our optimization schemes, because both proposals are obtained through different uncommunicating processes. Thus for each pair of nodes, the pairwise cost within each proposal labeling is small (since the proposal labelings are somehow "optimized"), while taking their labels from different proposals may generate a "seam" and, hence, incur a high pairwise cost. Therefore, the number of strongly non-submodular terms tends to be small and thus almost all nodes become labeled. A similar observation with respect to the number of non-submodular terms during image stitching was made in [14].

It is desirable to have sound theoretical guarantee about fusion move performance in case the number of unlabeled nodes is not negligable. Such guarantees can be obtained using a very simple strategy for labeling the unlabeled nodes (as presented in our previous work [18]). Let us assume without loss of generality that $E(\mathbf{x}^0) \leq E(\mathbf{x}^1)$. Then we label all unlabeled nodes in $\hat{\mathbf{y}}$ with 0. We denoted this auxiliary labeling as $\tilde{\mathbf{y}}$. Thus, $\tilde{y}_p = \hat{y}_p$ if $\hat{y}_p \neq$ '?', and $\tilde{y}_p = 0$ otherwise. The final output of the fusion move is the labeling $\mathbf{x}^c(\tilde{\mathbf{y}})$. In other words, all nodes p for which $\hat{y}_p =$ '?' receive their labels from $\mathbf{x}^0$ in the fused solution. According to the persistence property of the QPBO-mincut algorithm [4], [14], the energy of the auxiliary labeling $E^f(\tilde{\mathbf{y}})$ is not greater than the energy of the auxiliary labeling $E^f(\mathbf{0})$, which implies:

$$E(\mathbf{x}^c(\tilde{\mathbf{y}})) \leq E(\mathbf{x}^0) \leq E(\mathbf{x}^1)\,, \tag{5}$$

i.e. *the fusion move is guaranteed not to increase the energy (1) compared to the smallest of the energies of* $\mathbf{x}^0$ *and* $\mathbf{x}^1$. While this simple strategy for labeling the unlabeled nodes in the auxiliary labeling was sufficient for obtaining nearly global optimal fusions of proposals in all our experiments, fusion moves in harder optimization problems may require more sophisticated search strategies for the unlabeled nodes [17]–[19].

The computational cost of the minimum cut procedure consists of constructing and computing the maximal flow in a network graph that contains at most $2|\mathcal{V}|$ non-terminal vertices and at most $2|\mathcal{N}|+2|\mathcal{V}|$ edges. Note that if the majority of edges correspond to submodular terms the cost of running maxflow is very close to the runtime of maxflow for a fully submodular graph construction involving $|\mathcal{V}|$ non-terminal vertices and $|\mathcal{N}| + |\mathcal{V}|$ edges, see details in [14]. Finally, if for some nodes the proposed labels coincide ($x^0_p = x^1_p$), the corresponding vertices and edges connecting them need not be included in the graph thus further reducing the computational cost.

In the following, we denote the fusion move with the symbol '$\odot$', e.g.:

$$\mathbf{x}^c(\tilde{\mathbf{y}}) = \mathbf{x}^1 \odot \mathbf{x}^2. \tag{6}$$

2.3 Relation to previous multi-label approaches

The fusion move is closely related to several previous and contemporary approaches that have been suggested

for the minimization of energies associated with multi-label MRFs. The first approaches of this kind were suggested in [10], [11], [21]. There multi-label MRF optimization was reduced to a series of binary-label MRF optimizations (moves) that each were solved using minimum cut. The following types of moves were proposed in [10], [11], [21], and each of them can be regarded as a particular case of the fusion move discussed in this paper:

Expansion move [10], [11], [21]. Given a current labeling $\mathbf{x}^0$ and a label $\alpha \in \mathbf{L}$, each node p can either retain its original label x_p^0 or take the label α during the move. A repeated application of expansion moves where α iterates over the set of labels $\mathbf{L}$ is the alpha-expansion algorithm, which is perhaps the most popular graph cut-based optimization algorithm for multi-label MRFs in vision. Each expansion move can be regarded as a fusion move, where the proposals are $\mathbf{x}^0$ and the constant labeling $\mathbf{x}^1$, such that for each node p, $x_p^1 = \alpha$.

Swap move [10], [11], [21]. Given a current labeling $\mathbf{x}^0$ and a pair of labels $\alpha \in \mathbf{L}$ and $\beta \in \mathbf{L}$, each node p with $x_p^0 \in \{\alpha, \beta\}$ can either retain its original label or change it from α to β or vice versa. Nodes with labels which are different to α and β remain unchanged. The swap move can be regarded as a fusion move, where the proposals are the current labeling $\mathbf{x}^0$ and the labeling $\mathbf{x}^1$, such that $x_p^1 = \beta$ if $x_p^0 = \alpha$, $x_p^1 = \alpha$ if $x_p^0 = \beta$, and $x_p^1 = x_p^0$ otherwise.

Jump move [11]. The jump move is defined for an ordered discrete label space $\mathbf{L} = \{0, 1, \dots N\}$. Given a current labeling $\mathbf{x}^0$ and a number $k \in \{-N, -N+1, \dots, N-1, N\}$, each node p during the move can either retain its original label x_p^0 or change it from x_p^0 to $x_p^0 + k$ provided that the latter falls into the range of valid labels $\mathbf{L}$. The jump move can be regarded as a fusion move, where the proposals are the current labeling $\mathbf{x}^0$ and the labeling $\mathbf{x}_1$, such that $x_p^1 = x_p^0 + k$ if $x_p^0 + k \in \mathbf{L}$, and $x_p^1 = x_p^0$ otherwise.

More recently, the use of graph cut moves was investigated in the context of texture synthesis and image mosaicing [22], [23] as well as object detection and segmentation [24]. The types of moves proposed there are similar to the expansion move and can also be regarded as a particular instance of the fusion move. Loosely speaking, these approaches create one fixed proposal solution which has a larger extent than the given image. Then they introduce an auxiliary label α which corresponds to the $2d$-shift of the proposal solution with respect to the original image. Iterative alpha-expansion is then performed to optimize the energy.

We should emphasize here that all these works [10], [11], [21]–[24] have considered the submodularity of the binary-label problem as being necessary for a successful graph cut minimization. For instance, in [23] various so-called truncation schemas were discussed which ensure submodularity. As mentioned earlier, this restriction can be lifted with the application of non-submodular graph cut algorithms [4], [14]. Therefore, all the above-mentioned moves can be applied to more general energies than it was originally suggested by the authors.

Independently of our work on fusion moves, another group of researchers [19], [25] have simultaneously investigated the use of fusion-like graph cut moves for image-based rendering and stereo. The proposals considered there are fronto-parallel planes, or piecewise-planar proposals and smoothed versions of the current proposal in [19]. This work [19] also goes beyond pairwise MRFs and considers MRFs with triple-cliques. Each triple-clique auxiliary MRF with binary labels is then reduced to a pairwise MRF using the construction developed in [5], [26].

As we have discussed, particular instantiations of the fusion move idea have been around at least since [21]. The goal of this paper is therefore to demonstrate the generality of the fusion move idea as well as to suggest several new applications for vision and non-vision problems. An important point we advocate here as well as in our previous work [15], [16], which distinguishes this from other preceding work, is that the proposals employed during the iterative application of fusion moves do not necessarily have to be primitive, constant proposals or permutations of the current solution, but may rather come from powerful and, perhaps problem-specific, proposal generation processes, and include even continuous proposals as in [16]. Depending on the way the fusion move is applied, such flexibility allows to find solutions faster and/or with lower energy.

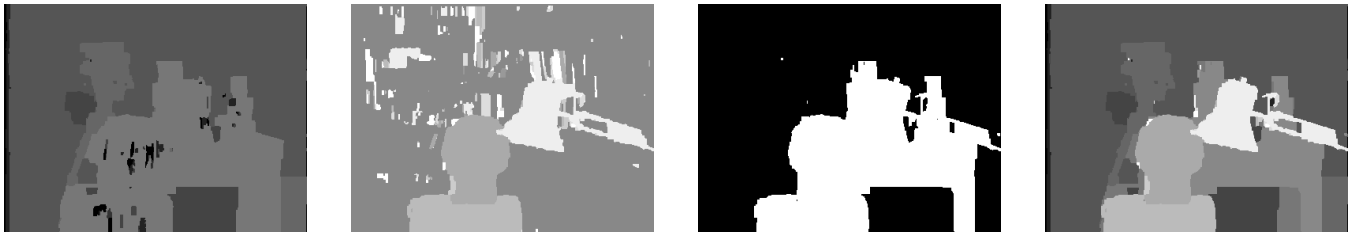
3 Applications and Algorithms

After having introduced the fusion move as a generalization of previous graph cut approaches, we present several new optimization schemes for minimizing energies associated with discrete and continuous-labeled MRF based on the fusion move. Its flexibility and the usability are illustrated by the variety of contexts where it can be applied.

3.1 Parallelizing alpha-expansion using fusion moves

We begin with a simple application of fusion moves to discrete-label MRFs. In particular, we consider the problem of parallelizing MRF optimization for multiple CPU cores. More specifically, our optimization scheme shows how to parallelize alpha-expansion. While relatively little attention is being paid to this problem, the advances of multi-core CPU architectures make this a current issue.

To begin, let us consider the case of two threads running on two CPU cores. The label set $\mathbf{L}$ is then split into two equal size subsets $\mathbf{L}^1$ and $\mathbf{L}^2$ ($\mathbf{L} = \mathbf{L}^1 \sqcup \mathbf{L}^2$, $\big||\mathbf{L}^1| - |\mathbf{L}^2|\big| \leq 1$). In our experiments, the exact way of splitting the label set into two halves did not affect computational performance much; an even-odd split was found to be slightly more efficient than a lower-upper halves split for ordered label sets.



(a) Current solution – 1st core, $E = 325 \cdot 10^4$ (b) Current solution – 2nd core, $E = 472 \cdot 10^4$ (c) Auxiliary binary variables *(all nodes labeled)* (d) Fused solution, $E = 183 \cdot 10^4$

Fig. 1. **Using fusion moves to parallelize alpha-expansion on two CPU cores.** The first core sweeps through label-disparities 0 to 7, while the second core sweeps through label-disparities 8 to 15. Images (a) and (b) show the current solutions on the first and the second core after one sweep. These solutions are fused and yield a disparity map potentially containing all disparities and having much lower energy as shown in (d). The auxiliary binary variables for this fusion move are shown in (c) ($0 =$ black, $1 =$ white; note that in this case there were no unlabeled nodes). Each of the cores starts the next sweep with the fused solution as the current one.

As a reminder, during alpha-expansion the optimization proceeds by sweeping through the label-space. During each sweep, the algorithm visits each label $\alpha \in \mathbf{L}$ once and performs an expansion move for this label from the *current solution* $\mathbf{x}_{cur}$: $\alpha \in \mathbf{L},\ \mathbf{x}_{cur} = \mathbf{x}_{cur} \odot \alpha$. The final labeling will typically slightly differ depending on the order in which the labels are visited. In all experiments in this paper we used a randomized label order, as this typically results in lower-energy labelings.

Similar to alpha-expansion, the *parallelized alpha-expansion* also proceeds by sweeps. Unlike alpha-expansion though, each of the threads maintains its own current solution $\mathbf{x}^i_{cur}$ throughout the process. During a sweep, each of the threads then visits its own set of labels ($\mathbf{L}^1$ or $\mathbf{L}^2$) and performs expansion moves for each of the visited labels starting from its own current solution.

Both threads perform their own sweep in parallel, running on two separate CPU cores. When both threads have completed their sweeps, the fusion move comes into action, as the current solution for the first thread is fused with the current solution for the second thread: $\mathbf{x}^{12} = \mathbf{x}^1_{cur} \odot \mathbf{x}^2_{cur}$. The current solutions from both threads are then updated with the fused version. After that a new sweep may be performed. The pseudocode for the algorithm is given in Fig. 2. Note that the fusion may incur non-submodular terms and hence cannot be handled by a generic graph cut optimization.

An example of the fusion move in context of this application is shown in Fig. 1 using a standard MRF for narrow-baseline stereo [21], where the labels correspond to disparities, unary potentials encode matching costs, and pairwise potentials (here, 4-connected Potts) ensure smoothness of the labeling. The fusion move after the first sweep is shown; for the sake of demonstration clarity, the labels (disparities) were split into upper and lower halves.

Parallelizing to multiple threads is similar to the two thread case. The label space $\mathbf{L}$ is subdivided into T equal parts $\mathbf{L}_1, \mathbf{L}_2 \ldots \mathbf{L}_T$; during each sweep each of the T threads sweeps through its own set of labels. After each sweep the current solutions are fused into a single one.

Algorithm 1 Parallelized alpha-expansion

Require: MRF optimization problem with label space $\mathbf{L}$
1: Split $\mathbf{L}$ into $\mathbf{L}^1$ and $\mathbf{L}^2$
2: Initialize $\mathbf{x}^1_{cur}$ and $\mathbf{x}^2_{cur}$ to any labelings
3: **for** several sweeps **do**
4: **for** $i = 1, 2$ **in parallel do**
5: **for** $\alpha \in \mathbf{L}^i$ **do**
6: $\mathbf{x}^i_{cur} \leftarrow \mathbf{x}^i_{cur} \odot \alpha$
7: **end for**
8: **end for**
9: wait for all threads
10: $\mathbf{x}^1_{cur} \leftarrow \mathbf{x}^1_{cur} \odot \mathbf{x}^2_{cur}$
11: $\mathbf{x}^2_{cur} \leftarrow \mathbf{x}^1_{cur}$
12: **end for**
13: **return** $\mathbf{x}^1_{cur}$

Fig. 2. The pseudocode for the parallelized alpha-expansion algorithm for the case of 2 CPUs.

These fusion moves are accomplished in a hierarchical manner in parallel; i.e. for four cores, the two fusions $\mathbf{x}^{12}_{cur} = \mathbf{x}^1_{cur} \odot \mathbf{x}^2_{cur}$ and $\mathbf{x}^{34}_{cur} = \mathbf{x}^3_{cur} \odot \mathbf{x}^4_{cur}$ are computed in parallel first, and the fusion $\mathbf{x}^{1234}_{cur} = \mathbf{x}^{12}_{cur} \odot \mathbf{x}^{34}_{cur}$ is done next.

In terms of memory requirements, parallelized alpha-expansion requires more memory than the conventional alpha-expansion, as each thread requires $O(|\mathcal{V}| + |\mathcal{N}|)$ additional memory to store its current solution and to maintain the graph for expansions. Still, given an MRF with a large number of labels, both alpha-expansion and parallelized alpha-expansion algorithms are quite memory efficient, as their memory requirements do not grow with the size of the label space (unlike e.g. message-passing methods).

Tab. 1 gives results for the parallelized alpha-expansion for up to five threads and cores for a large-scale stereo MRF problem based on truncated-linear pairwise potentials [10] (the "books" dataset from [27] was used). As can be seen, the parallelization allows a substantial reduction of the run-time. While the accel-

TABLE 1
Time vs. Number of threads for a sample large stereo MRF problem (1390x1110 nodes, 160 disparity labels). The experiment was run on a multi-core computer, so that each thread could be allocated a separate CPU core. 4 sweeps through the label space were performed. All runs yielded labelings with a similar energy (19600 ± 20), while the parallelized alpha-expansion achieved this faster than alpha-expansion (1.6 times faster for 2 threads, 2.49 times faster for 5 threads).

Number of threads	1	2	3	4	5
Time (in seconds)	450	263	219	201	187
Energy	19601	19586	19593	19614	19618

eration is, of course, not linear (due to shared memory access as well as the presence of fusion move steps), it is still quite considerable. Quite importantly, the energies of the labelings obtained with parallelized runs are roughly equal to the energy obtained with standard single-thread alpha-expansion (in fact, even slightly lower in case of $T = 2, 3$). The number of unlabeled nodes within the fusion moves was typically zero and never exceeded 0.05% during any of the runs.

3.2 The LogCut algorithm

As we have just seen, parallelized alpha-expansion allows us to accelerate MRF optimization provided multiple CPU cores are available. Now we will discuss how fusion moves can be used for a typical computer vision MRF to speed up MRF optimization on a single processor, beyond the current state of the art. The proposed approach is based on the observation that alpha-expansion performs a rather excessive number of graph cut computations during each sweep. Indeed, each sweep of alpha-expansion involves $|\mathbf{L}|$ graph cuts, while $\log|\mathbf{L}|$ binary graph cuts are sufficient, in principle, to get a full MRF labeling using a divide-and-conquer strategy. Such a divide-and-conquer graph cut algorithm, termed *LogCut*, was proposed in our previous work [15].

The LogCut algorithm assumes the availability of similar MRF problems at training stage, and uses training to improve the efficiency of the divide-and-conquer scheme. In this paper, we will discuss this labeling algorithm briefly and then focus on the use of the fusion move. We refer the reader to [15] for remaining details.

LogCut assumes an ordered set of labels $\mathbf{L} = \{0, 1, ...K\}$[1]. Then the label x_p of each node p can be represented as a B-bit word $[x_p^0, x_p^1, ..., x_p^{B-1}]$, where $B = \lceil \log K \rceil$:

$$x_p = \sum_{b=0}^{B-1} x_p^b \; 2^b. \tag{7}$$

The minimization problem for the energy from (1) can then be seen as a binary-label optimization problem with $|\mathcal{V}| \cdot B$ variables x_p^b (which effectively is an MRF with large cliques each of maximum size B). Because such an optimization problem is intractable for conventional methods, we aim at retaining the original pairwise clique structure. We do so by iterating over the bit-array $\mathbf{x}^b = (x_1^b, \ldots, x_p^b)$. In short, LogCut starts with finding the most significant bit $b = B - 1$, and subsequently sweeping through the bit-values at each node down to the least significant bit $b = 0$. At each level, the decision about the value of a particular bit b is made jointly for all nodes via graph cut optimization. Then, instead of requiring the number of iterations to grow linearly with K, as for alpha-expansion, only a logarithmic number of graph cut computations is required to obtain a labeling.

More specifically, LogCut replaces the original optimization problem (1) with a series of binary-labeled MRF optimization problems, solved subsequently:

$$E^b(\mathbf{x}^b) = \sum_{p \in \mathcal{V}} U_p^b(x_p^b) + \; \gamma \sum_{p,q \in \mathcal{N}} V_{pq}^b(x_p^b, x_q^b). \tag{8}$$

The new binary-labeledpotentials U_p^b and V_{pq}^b have to be derived from the original multi-labeled energy (1). In the previous examples this transformation has been straight forward as shown in (3-4). Unfortunately this is nontrivial here since when operating on a particular bit b all the less significant bits $(0, ..., b-1)$ are unknown. Hence, the potentials of the original MRF do not uniquely define the potentials U_p^b and V_{pq}^b. This issue was addressed in [15].

For unary potentials U_p^b, a simple *"min"* heuristics that choses $U_p^b(x_p^b)$ as the minimum of $U_p(x_p)$ over all possible settings of unknown lower bits proved to work very well. Efficient choice of pairwise potentials V_{pq}^b turned out to be much harder, and in [15], several strategies for pairwise terms were discussed. The most successful one was based on training. It uses the training data to choose the new potentials V_{pq}^b, so that the LogCut procedure finds labelings with the lowest possible original energy (1). Importantly, such training procedure does not require any ground truth labeling, as the training data is only required to assess the energy values of the solutions obtained by LogCut under different choices of pairwise potentials.

In [15], we found such training procedure not to be prone to over-fitting, at least for common low-level vision tasks. Thus, when training and test datasets came from a reasonably similar experimental setup, the energy of the solution computed with one sweep of LogCut was typically only slightly higher than the energy of the solution produced with one sweep of alpha-expansion. But LogCut only required a logarithmic number of graph cuts resulting in much shorter run-times for applications where the number of labels is large, as shown in Tab. 2. Our test examples cover a variety of settings: First, low- and high-resolution stereo matching datasets from [27] (458×356 and 1390×1110 pixels). Next, image restoration consisting of image denoising and inpainting of an

1. As discussed in [15] the algorithm can also be applied to partially ordered label-sets such as 2D flow vectors.

TABLE 2
Speedup of LogCut sweeps compared to alpha-expansion sweeps. The speed up is given as the ratio of computation times (alpha-expansion over LogCut). Energy differences (LogCut minus alpha-expansion, thus lower is better) are shown after one sweep and at convergence of iterated versions of both algorithms (all energies were defined in a standard way, so that all individual energy terms U_p and V_{pq} had minima equal zero). From the results it is clear that the speed advantage of LogCut increases with the number of bits used to encode the label-space.

Problem set	Number of labels	Speed up 1st iter.	Energy diff. 1st iter.	Energy diff. convergence
Low-res. stereo	64	**4.9**	+2.6%	-0.3%
High-res. stereo	256	**9**	+3.6%	-0.2%
Image restoration	256	**12.4**	+0.5%	-2.6%
Optical flow	1024	**20.7**	+2.5%	-2.4%

obscured area using the "penguin" image[2] and a set of images from the Corel dataset. And finally optical flow estimation for a set of frame pairs from the well-known Yosemite sequence. For a more detailed description of the data and the models used, we again refer the reader to [15].

As a single LogCut sweep results in a slightly higher energy than an alpha-expansion sweep, a natural question is whether the LogCut algorithm can be modified to produce labelings with lower energies than one sweep of alpha-expansion, or even whether it can beat multiple sweeps of alpha-expansion. Next, we demonstrate how this can be done using the fusion move idea.

Iterated LogCut. The LogCut optimization process introduced so far makes a series of "hard" decisions: once the decision on the bit value x_p^b is made, this bit value remains fixed. Thus, the precise pattern of the errors made by the LogCut algorithm depends on the way in which the bit-coding is applied. Therefore, we consider an iterated version of the LogCut algorithm, where different bit-codings are used. During the first iteration, the bit-coding is applied to the label $x \in 1, \ldots, K$ as described above, which yields the current solution $\mathbf{x}_{cur}$.

During subsequent iterations the bit-encoding is applied to circularly shifted labels $(x_p + s) \mod K$ rather than to x_p itself. After that, LogCut is run for the new bit-encoding and the resulting solution is shifted back to obtain a new proposal. The proposal is then combined with the current solution using the fusion move (the current solution is initialized to the LogCut result without the shift). An example of such a fusion is shown in Fig. 4. It can be seen how the fusion move allows to obtain a lower energy solution by picking different parts from the different proposals.

The *label shift* s may be chosen randomly, but a more efficient strategy is to choose the shift that maximally

2. To ease comparison, the "penguin" image restoration model is taken from [1], [28].

Original image | Solution 1 $E = 1628$ | Solution 2 $E = 1621$ | Fusion $E = 1569$

Fig. 4. **Iterated LogCut.** In this image restoration example, the original image is denoised and the black bar is inpainted. The fusion move can be used to obtain a lower energy labeling by fusing solutions coming from LogCut with different bit-encodings (solution 1 - original encoding, solution 2 - circularly shifted encoding). The fused solution has substantially lower energy and does not exhibit the artifacts present in either of the proposals (marked by arrows).

decorrelates the errors with all previous bit encodings. Effectively, the MRFs (8) group labels into *superlabels*, e.g. the MRF at the highest bit level $B-1$ operates with two superlabels of size 2^{B-1}, the next one operates with four superlabels of the size 2^{B-2}, etc. The bit coding then defines how the labels are grouped into superlabels on each of the bit levels. This observation allows optimizing the choice of the shifts.

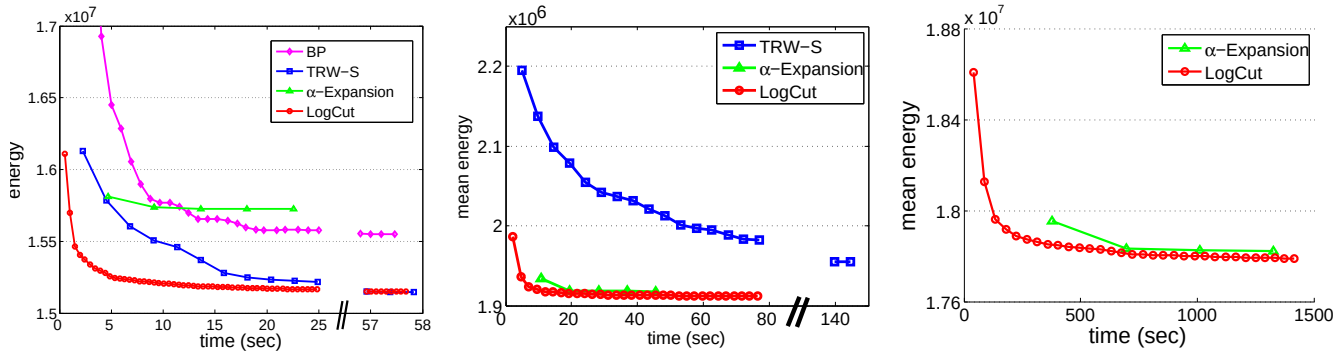
For example, it is known in advance that if two shifts s_1 and s_2 differ by $K/2$ then LogCut will give exactly the same result for both shifts. This is because the grouping of labels into superlabels will remain the same at all bit levels (by simply switching the meaning of labels 0 and 1 for the MRF (8) at the highest bit level). In general, to predict how successful the shifts s_1 and s_2 will be at producing proposals with different errors, we define the *regrouping distance* between shifts at a certain level b as

$$r_b(s_1, s_2) = \frac{1}{2} - \left| \frac{|s_1 - s_2| \mod 2^b}{2^b} - \frac{1}{2} \right|, \qquad (9)$$

which varies between 0 and $\frac{1}{2}$. When $r_b(s_1, s_2) = 0$, s_1 and s_2 differ by a multiple of 2^b, and the subdivision of labels at level b into groups for both shifts are identical. Conversely, the largest possible value $r_b(s_1, s_2) = \frac{1}{2}$ implies that the corresponding groups of labels at level b are offset by half of the group size, giving a maximal regrouping of labels. The total regrouping distance is naturally defined as a sum over all bit levels:

$$r(s_1, s_2) = \sum_{b=1}^{B} r_b(s_1, s_2). \qquad (10)$$

Now, at each iteration of the iterative LogCut procedure, the shift that is the most distant from all previous shifts according to (10) is chosen. This encourages a maximal diversity amongst the solutions to be fused. In our experiments, we found that this distance predicted very well which shift gives the greatest reduction in energy after fusion. In general, fusing results from different



(a) Image restoration – "penguin" (b) Low res. stereo (64 labels) (c) High res. stereo (256 labels)

Fig. 3. **Time vs. energy plots for image restoration and stereo matching with LogCut**. The image restoration example in (a) uses the model from [1], [28], for the sake of comparison. The stereo experiments in (b) and (c) are carried out at two different resolutions, and we report the mean energy over a test set of 5 examples in tcase of high resolution (c) and 8 examples in case of low resolution (b). All experiments demonstrate better computational performance of iterated LogCut as well as its ability to obtain low energy solutions after several iterations. Iterated LogCut is compared with α-expansion [10] as well as respective authors' implementations of efficient belief propagation [28], and TRW-S [29] (where memory feasible). Every tick in each graph corresponds to one iteration of LogCut, one sweep of alpha-expansion, or one round of message-passing of BP or TRW-S.

Algorithm 2 Iterated LogCut

Require: MRF problem E,
trained MRFs E^b, $b = 0, \ldots, B-1$.

1: **for** several iterations **do**
2: pick s to minimize (10)
3: apply bit-coding to $(\mathbf{x} + s) \mod K$
4: **for** $b = B-1$ downto 0 **do**
5: $\mathbf{x}^b \leftarrow \arg\min E^b(\mathbf{x}^b)$
6: **end for**
7: $\mathbf{x}^{cur} \leftarrow \mathbf{x}^{cur} \odot [\mathbf{x}^0, \mathbf{x}^1, \ldots \mathbf{x}^{B-1}]$
8: **end for**
9: **return** $\mathbf{x}^{cur}$

Fig. 5. The pseudocode for the Iterated LogCut algorithm.

bit encodings proves effective in optimizing the energy much further than is possible with just a single iteration of LogCut. The pseudocode for the iterated LogCut algorithm is given in Fig. 5.

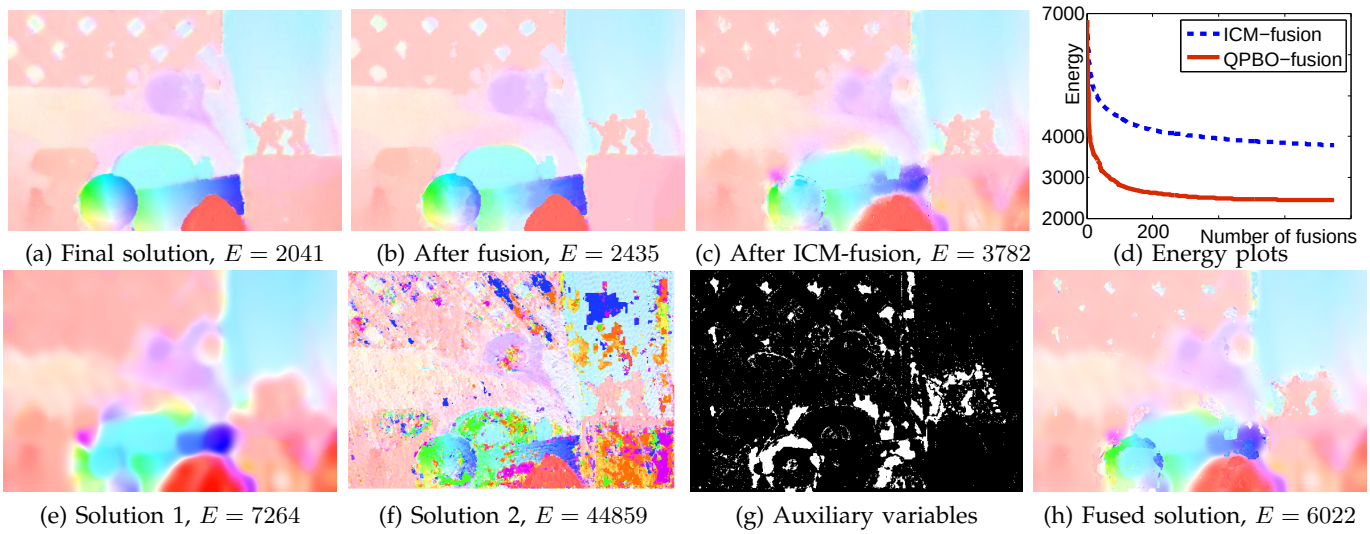
Over several dozen of runs for an image restoration problem (where the number of non-submodular edges during each fusion move is typically the largest), the number of unlabeled nodes never exceeded 0.4% for any optimization instance.

An important property of the iterated LogCut is that it spends most of the time on obtaining proposals (lines 3–6) rather than fusing them. Consequently, the algorithm may be efficiently parallelized: the solutions for different shifts s may be computed in parallel on different cores and fused afterwards. Nevertheless, even being run in a single thread mode, the iterated LogCut algorithm performed favourably compared to other state-of-the-art MRF optimization algorithms (see Fig. 3). As can be seen, the use of fusion moves within the iterated LogCut algorithm allows to obtain labelings with much lower energy compared to a single iteration of LogCut. As a result, the algorithm is able to compute initial high-quality solutions faster than other algorithms and in the end produces labelings with energies that are on a par with best-performing competitor (TRW-S [29] or alpha-expansion [10], depending on the application). As can be seen in Fig. 3(a) and Tab. 2, the fusion of multiple problem-specific LogCut proposals may even yield solutions with considerably smaller energy compared to the fusion of multiple constant labelings (alpha-expansion), since they are different local optimizers. The increase in performance (lower runtime), however, comes at the price of having to perform an offline training step to obtain the new unary and pairwise terms of the auxiliary random field (8).

3.3 The FusionFlow algorithm

The previous two application examples dealt with MRFs with discrete labels and demonstrated how fusion moves can be used to accelerate optimization in that case. Here, we will consider MRFs with continuous labels. In particular, we will perform optical flow estimation as described in our previous work [16]. Given a pair of frames I^0 and I^1, the goal is to find a 2D *flow vector* $x_p \in \mathbf{R}^2$ for each pixel p in the first image I^0, so that pixel locations p in I^0 and $p + x_p$ in I^1 show corresponding features.

Apart from the different label space, the optical flow problem is actually quite similar to stereo-matching, where graph cut optimization has been applied with great success. We formulate the optical flow estimation problem as a spatially-discrete, continuous-valued MRF. The corresponding energy takes the form (1), but with the crucial difference to the previous cases that the values (labels) at each pixel are continuous and moreover two-dimensional (i.e., $\mathbf{L} = \mathbf{R}^2$). As in stereo



(a) Final solution, $E = 2041$ (b) After fusion, $E = 2435$ (c) After ICM-fusion, $E = 3782$ (d) Energy plots

(e) Solution 1, $E = 7264$ (f) Solution 2, $E = 44859$ (g) Auxiliary variables (h) Fused solution, $E = 6022$

Fig. 6. **Optical flow estimation using the fusion move.** Flow estimation for the *Army* sequence from [30] (see Fig. 7). The top row shows the solution after the final continuous optimization (a), the solution after the fusion stage (b), and the solution after the suboptimal ICM-based fusion stage (c). The energy as a function of the number of fusions for the fusion using QPBO and for the ICM-based fusion are plotted in (d). The bottom row shows the first iteration of the algorithm: A randomly chosen initial solution (e) (computed with Horn-Schunck) is fused with another randomly chosen proposal solution (f) (computed with Lucas-Kanade). The fusion move allows to compute the optimal combination (h) resulting in a much lower energy, which is passed on to the next iteration. The auxiliary variables are shown on (g). In this example 99.998% of the nodes were labeled.

the unary potentials are used to encode the observation model while the pairwise potentials are used to impose smoothness of the flow field. More specifically, the data term $U_p(x_p)$ encodes the match between the RGB value at pixel p of the first image and pixel $p+x_p$ of the second image. To be more robust to illumination changes and shadows, we first high-pass filter[3] the input images I^0 and I^1 by subtracting Gaussian-smoothed versions $G*I^i$. Hence we aim to match I^1-G*I^1 with I^0-G*I^0. Color and brightness changes are penalized using the Geman-McClure robust penalty

$$\rho_U(d) = \frac{d^2}{d^2+\mu^2}$$

to account for occlusions and specularities (d here is the Euclidean RGB distance between the corresponding pixels; we manually set the constant $\mu = 16$). The pairwise term $V_{pq}(x_p, x_q)$ penalizes the difference between horizontal and vertical components of the flow between neighboring pixels p and q using the non-convex robust penalty

$$\rho_V(d) = \log\left(1 + \frac{1}{2\nu^2}d^2\right),$$

which is derived as the log of the Student-t distribution and motivated by the studies of the natural statistics of optical flow [31]. Here, d is the difference between either horizontal or vertical components of the flow vectors at the two adjacent pixels; the parameter ν was set to 0.2.

3. Note that while high-pass filtering improves performance in areas of large-scale brightness changes, it may deteriorate performance in other (e.g., noisy) areas. However, in practice we found that the advantages outweigh the problems and we obtained substantially improved results overall. Nonetheless, it should be noted that the FusionFlow approach is independent of this preprocessing step.

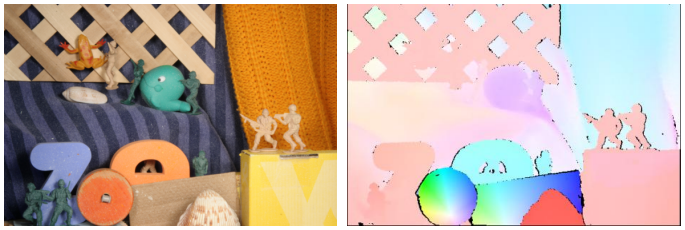


Fig. 7. The *Army* sequence from [30]: one of two input images and the ground truth flow (*black* = unknown, *hue* = motion direction, *saturation* = motion magnitude).

The proposed MRF energy is more difficult to optimize than the energies used in recent popular optical flow methods based on continuous optimization such as [32], since both data and spatial terms in our formulation are robust, thus non-convex. Also, the data term works with the high frequency content of images, which only adds to its non-linearity. Therefore, traditional continuous optimization schemes based on coarse-to-fine estimation and gradient descent often end up in poor local minima.

On the other hand, the proposed energy is also harder to optimize than many energies used in stereo matching, since the value at each pixel spans a potentially unbounded 2D domain rather than a bounded 1D domain, making it infeasible for purely discrete techniques to sample it densely enough. Our FusionFlow approach addresses this using a new, powerful *discrete-continuous* optimization scheme based on fusion moves that combines the merits of discrete and continuous-valued optimization approaches.

The minimization proceeds in two stages. During the first stage, a number of proposals are generated and

combined using fusion moves[4]. It is important to note that the proposal solutions need not to be of high quality across the whole image in order to be "useful". Instead, each solution may only contribute to a particular region of the final solution as long as it contains a reasonable flow estimate for that region, no matter how poor it is in other parts. This suggests the use of different flow computation methods with different strengths and weaknesses for computing the proposals. Therefore, we decided to use proposals computed with the two classic flow estimation algorithms, namely the Lucas-Kanade [34] (LK) and the Horn-Schunck [35] (HS) methods. Indeed, Lucas-Kanade solutions often yield good results in textured regions, but are virtually useless in textureless regions, while the Horn-Schunck method often gives good estimates in regions with smooth motion even when lacking image texture, but severely oversmoothes motion discontinuities.

To obtain a rich and varying set of proposal solutions, we use the LK and HS methods with various parameter settings. For HS we vary the strength of the regularization ($\lambda \in \{1, 3, 100\}$). Since both methods should be applied within a coarse-to-fine warping framework to overcome the limitations of the linearized data term (of the proposals, not of our energy), we also vary the number of levels in the coarse-to-fine hierarchy ($l \in \{1, \ldots, 5\}$). Finally, for all LK solutions and a few HS solutions we produce shifted copies (a flow field is shifted by both $\pm 2^{l-1}$ and $\pm 2^l$ pixels along each image axis). For the LK method, this corresponds to the use of a family of non-centralized windows and, hence, gives better chances of providing correct flow values near flow discontinuities, and as we found reduces the energy of the solution. These variations result in about 200 proposals (most of them, however, are shifted copies and do not take much time to compute).

The fusion of the proposals is accomplished in a sequential manner, where the initial labeling corresponds to one of the proposals, randomly chosen. After that, the remaining LK and HS proposals are visited in random order, and each of them is fused with the current solution. An example of such a fusion (during the first iteration of the process) for a sample problem is shown in Fig. 6 in the bottom row.

After all LK and HS solutions are visited once, we add new proposals based on the current solution. In particular, the motion vectors of the obtained fused solution are clustered into 64 clusters using the k-means algorithm. The centers of the clusters $c_i \in \mathbf{R}^2$ are used to produce constant proposal flow fields $x_p^i = c_i$. Note that more sophisticated proposals that are dependent on the current solution may be computed (see e.g. [36]) and our constant solutions are just one step in this direction. The constant proposals are then added to the pool of LK and HS proposals and the fusion process continues until each proposal is visited two more times. At this point the procedure typically converges, i.e. the obtained fused solution can no longer be changed by fusion with any of the proposals. The number of unlabeled nodes during each fusion was always negligible (we never observed it exceeding 0.1% of the nodes).

After the fusion of solutions is accomplished, a low-energy solution that is much closer to ground truth and has much smaller energy than any of the proposals is obtained (Fig. 6a). We have also compared QPBO-based fusion against a simpler and faster fusion technique. Thus, we have used the ICM algorithm [37] to solve approximately each fusion problem (3), otherwise leaving the whole fusion framework unchanged. During each fusion step, this fusion procedure tries to change the flow vector at a pixel from the current solution to the flow vector for this pixel in the proposed solution. The pixels are visited subsequently and until convergence; the changes are accepted only if they decrease the energy. Fig. 6c clearly demonstrate that using ICM rather than QPBO for the fusion operation results in the inferior solution. Fig. 6d also demonstrate how the energy is decreased with each fusion, when the optimal (QPBO) and the suboptimal (ICM) fusion algorithms are used. The QPBO-fusion decreases the energy much faster (a steeper curve), bypassing the final ICM-fusion energy in just 13 fusions. One may conclude that it is the power of the QPBO algorithm and not only the diversity of the proposals that is required to compute a good solution as the result of the fusion.

During a subsequent second stage of the optimization, we perform a continuous optimization step that helps "cleaning up" areas where the proposal solutions were not diverse enough, which for example may happen in relatively smooth areas (visually, the difference is typically very small as can be seen from Fig. 6 (b)-(c)). This is done using a standard conjugate gradient method [38]. The pseudocode for the FusionFlow algorithm is given in Fig. 8.

Since the discrete optimization step avoids many of the poor local optima that are problematic for purely continuous optimization methods, the combination of discrete and continuous optimization leads to local minima with a substantially lower energy in most of our experiments [16]. In particular, the evaluation of our method on 8 benchmarking sequences [30] demonstrated that the proposed discrete-continuous optimization based on fusion moves was superior to baseline continuous optimization (conjugate gradients applied in a coarse-to-fine manner). Moreover, even the first stage of our approach (fusion of < 300 proposals) considerably outperformed alpha-expansion optimization with more than 1000 labels (discrete baseline method in Table 3.3) that were uniformly spaced in an admissible range of motion (which was determined from the FusionFlow solution).

4. Interestingly, it was demonstrated very recently in [33] that the fusion of flow fields may also be accomplished using continuous optimization, with the possibility of introducing higher-order smoothness terms.

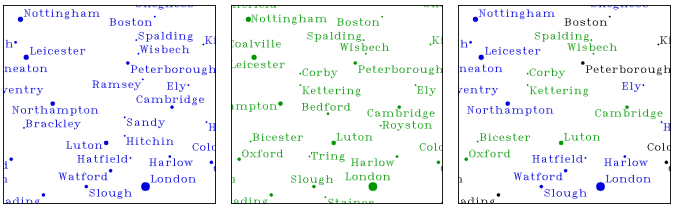
TABLE 3
Comparison of different optimization techniques: Our full discrete-continuous, our discrete (i.e. fusion of proposals without continuous improvement), baseline continuous, and baseline discrete algorithms. Shown are the flow accuracy (average angular error) and the energy achieved. On the 8 test datasets [30], our optimization scheme consistently outperforms the two baseline algorithms.

Optimization	Army		Mequon		Schefflera		Wooden		Grove		Urban		Yosemite		Teddy	
	AAE	$E(\mathbf{f})$	AAE	$E(\mathbf{f})$	AAE	$E(\mathbf{f})$	AAE	$E(\mathbf{f})$	AAE	$E(\mathbf{f})$	AAE	$E(\mathbf{f})$	AAE	$E(\mathbf{f})$	AAE	$E(\mathbf{f})$
Fusion + continuous	**4.43**	**2041**	**2.47**	**3330**	**3.70**	**5778**	**3.68**	**1632**	4.06	**17580**	6.30	**5514**	4.55	**1266**	7.12	**9315**
Fusion only	4.97	2435	4.83	4375	5.14	7483	5.24	2180	**4.00**	21289	**6.27**	6568	**4.03**	1423	**6.68**	10450
Baseline continuous	7.97	4125	52.3	21417	36.1	24853	16.8	7172	64.0	78122	46.1	26517	23.2	4470	63.9	31289
Baseline discrete	5.61	3038	5.19	6209	5.36	8894	4.94	2782	9.03	44450	18.7	17770	5.67	1995	9.13	15283

Algorithm 3 FusionFlow

Require: Optical flow MRF problem E.
Generate a set X of proposal solutions
Initialize $\mathbf{x}^{cur}$ to a random solution from X
for $sweep = 1, 2, 3$ **do**
  for $\mathbf{x} \in X$ **do**
    $\mathbf{x}^{cur} \leftarrow \mathbf{x}^{cur} \odot \mathbf{x}$
  end for
  if $sweep = 1$ **then**
    Cluster the set of 2D vectors $\{x_p^{cur}\}$
    Add constant solutions for cluster centers to X
  end if
end for
for several iterations **do**
  $\mathbf{x}^{cur} \leftarrow \mathbf{x}^{cur} + \delta(\mathbf{x}^{cur})$ \\ Continuous optimization
end for
return $\mathbf{x}^{cur}$

Fig. 8. The pseudocode for the FusionFlow algorithm.



(a) Prop.1, $E = 933$ (b) Prop.2, $E = 2003$ (a) Fused, $E = 855$

Fig. 9. **The fusion move for cartographic label placement** (parts of a larger map are shown). The proposal solutions are obtained using belief propagation (first and second iteration). The fusion move combines these two labelings into a better one. The auxiliary variables are visualized using the following color coding: blue text – first proposal ($y_p = 0$), green text – second proposal ($y_p = 1$), black text – both proposals have the same MRF labels. All nodes were labeled by graph cut during this fusion move.

3.4 Cartographic label placement

As our final example for how fusion moves can be used, we consider a class of MRF energies that arise in the so-called cartographic label placement problem (also known as *map labeling*). These MRFs have a very different structure from those found in computer vision and therefore, not surprisingly, optimization methods popular for vision MRFs often fail for this problem. Yet, below we demonstrate that the fusion move idea still proves useful for this type of problem, and present a general algorithm based on the fusion of belief propagation proposals.

Cartographic label placement is the problem of arranging geographic features along with their text annotations on a map. Here we consider a point-label placement problem in particular, where we are given a rectangular map region, a set of *populated places*, where each populated place is described by a single pair of geographic coordinates, its name, and its population (or other importance measure). Fig. 9 shows an example of a cropped map in where the size of the dot corresponds to size of population. The task is then to create a map of the region that contains the maximal number of populated places along with their annotations, while still being legible and visually pleasant. This is an important and well-studied problem and the interested reader is referred to [39] for a comprehensive bibliography on the subject.

We cast this problem as a multi-label MRF problem as follows. Each populated place is represented using a node p. If a populated place is added to the map, it should be annotated, and the text label can be placed using one of various predefined ways. The set of labels $\mathbf{L}$ is therefore discrete and in our implementation consists of 9 labels: $x_p \in \{0, \leftarrow, \rightarrow, \uparrow, \downarrow, \nearrow, \searrow, \swarrow, \nwarrow\}$. The label 0 means that the populated place is not added to the map (this is required since the list may contain an excessive number of places), while the other labels imply that the place is shown on the map in form of a dot centered at its respective coordinates. The remaining labels correspond to relative locations of the text label with respect to the dot, e.g. $x_p = \uparrow$ means that the annotation is placed above the dot. We then define the unary potential as $U_p(x_p) = \lambda \cdot \text{population}^2$ if $x_p = 0$ and $U_p(x_p) = 0$ otherwise, which penalizes the absence of a place by the square of its population, while making all ways of annotating equally acceptable.

We associate two rectangles $TR_p(x_p)$ and $LR_p(x_p)$ with each node p and annotation label x_p. For $x_p \neq 0$, the *tight rectangle* $TR_p(x_p)$ is defined as a bounding rectangle that includes the dot and the annotation text placed in position x_p, while the *loose rectangle* $LR_p(x_p)$ is defined as the tight rectangle $TR_p(x_p)$ expanded in each of the four directions by a fixed offset Δ. For $x_p = 0$ both rectangles are defined as being empty. The pairwise potentials V_{pq}

are then defined as:

$$V_{pq}(x_p, x_q) = \begin{cases} +\infty\,, & \text{if } TR_p(x_p) \cap TR_q(x_q) \neq \emptyset\,, \\ |LR_p(x_p) \cap LR_q(x_q)|\,, & \text{otherwise.} \end{cases} \tag{11}$$

We, thus, prohibit any overlap of the tight rectangles and penalize the overlap of the loose rectangles proportionally to the area of the overlap. The edge set $\mathcal{N}$ includes all pairs of nodes p and q, such that their loose rectangles may overlap for at least one choice of labels ($\exists x_p, x_q : V_{pq}(x_p, x_q) \neq 0$).

Minimizing the energy of the resulting MRF (1) allows finding the trade-off between the amount of information and the legibility, under the hard constraint of non-overlap of the annotations. This MRF is quite different from the pixel grid-based MRFs that we considered in the previous examples: It has a varying topology, is typically highly-connected, and the pairwise potentials have very different form compared to smoothness-ensuring potentials used in the other examples. Not surprisingly, the relative performance of MRF optimization algorithms proved to be quite different in our experiments from the performance reported on vision MRFs.

We performed a set of experiments with a list of the 910 largest populated places in the UK. We report the results here for a particular choice of the map size, font and constants Δ and λ. However, a similar relative performance of the various algorithms was observed over a large range of the parameter space. The edge set $\mathcal{N}$ in the experiments below contained 52490 edges overall. Thus each node was connected to 115 other nodes on average, although the graph density varied quite considerably being much higher in central England.

We evaluated and compared the following MRF optimization algorithms: alpha-expansion (using non-submodular graph cuts), TRW-S [20], min-sum belief propagation, iterated conditional modes (ICM) [37], and a new algorithm proposed below that is based on belief propagation and the fusion move. We also tried simulated annealing, but empirically it was very difficult to choose an appropriate cooling schedule due to the presence of both infinite and finite values in pairwise potentials. We note, however, that other mathematical formulations of the label placement problem may be much better suited for simulated annealing optimization (see [40]).

The performance of the various algorithms is summarized in Table 4. It can be seen that conventional algorithms that typically perform best on vision MRFs (alpha-expansion and TRW-S) perform much worse than belief propagation (BP), which is able to find solutions with much lower energy. However, BP did not converge for all the label placements MRFs we have tried. Non-convergence of belief propagation is a common problem, and, at a first glance, the best solution may be to just keep the best labeling found at any of the iterations. We observed, however, that the labelings found in subsequent iterations were quite different from each other, and although most of them had significantly higher energy compared to the best found labeling, they still contained parts with good arrangements of text labels.

TABLE 4
Performance of various MRF optimization algorithms on an instance of the cartographic label placement problem. We give two numbers for those algorithms that do not decrease the energy monotonically (min-sum belief propagation and TRW-S): the lowest energy observed during any of the iterations and the energy at the final iteration. The iterated conditional modes (ICM) algorithm was run from 100 random starting-points and with a random order for visiting nodes.

Algorithm	Energy $\cdot 10^3$
ICM (best among 100 runs)	3561
Alpha-expansion (convergence)	1220
TRW-S (1000 iterations)	2773/2832
Belief propagation (100 iterations)	908/59193
BP-fusion (10 iterations)	833
BP-fusion (100 iterations)	829

TABLE 5
Energies of belief propagation and BP-fusion solutions after each of the first 10 iterations.

Iteration #	BP	BP-fusion	Iteration #	BP	BP-fusion
1	933	933	6	60053	838
2	2002	855	7	1324	836
3	1185	851	8	2106	833
4	2315	849	9	59196	833
5	938	847	10	2139	833

The idea put forward here is thus to combine several belief propagation labelings using fusion moves. The algorithm maintains a current solution and fuses it with the belief propagation solution after each iteration (message-passing round). This algorithm, called BP-fusion, is summarized in Fig. 10.

As can be seen from Tables 4 and 5, the suggested algorithm efficiently combines non-convergent belief propagation solutions and is able to obtain a considerably lower energy labeling than any of the belief propagation iterations. An example of the fusion move within the BP-fusion algorithm is shown in Fig. 9. We did not observe unlabeled nodes for fusion moves using our experimental settings, but under extreme variation of the

Algorithm 4 BP-fusion

Require: MRF problem E
1: Initialize current solution $\mathbf{x}^{cur}$
2: Initialize belief propagation messages
3: **for** several iterations **do**
4: Update beliefs with a round of messages
5: $\mathbf{x}^{BP} \leftarrow$ Minimal solution from beliefs
6: $\mathbf{x}^{cur} \leftarrow \mathbf{x}^{cur} \odot \mathbf{x}^{BP}$
7: **end for**
8: **return** $\mathbf{x}^{cur}$

Fig. 10. The pseudocode for the BP-fusion algorithm.

parameters unlabeled nodes may appear.

4 Discussion and Future Work

Summary. In this paper, we have proposed the fusion move as an efficient technique for the minimization of energies associated with pairwise Markov random fields with a certain focus on low-level vision applications. We showed that the fusion move can be regarded as a generalization of a variety of previous graph cut moves that have been suggested in the literature. We presented a number of *hybrid* algorithms in which fusion moves are used to combine proposal solutions. These algorithm examples show that the fusion move is a flexible technique that applies to a variety of settings and often yields algorithmic improvements. To summarize, let us review the source of the proposal solutions and the role played by the fusion move for each of the proposed algorithms:

- **Parallelized alpha-expansion algorithm: proposals from multiple threads / CPU cores.** By splitting the label space between threads running on separate CPU cores we showed that graph-cut based MRF optimization can be efficiently parallelized. Here fusion moves are used to combine and coordinate the optimization processes performed within different threads.
- **Iterated LogCut algorithm: proposals from fast divide-and-conquer algorithms.** The proposals are generated with a very fast, yet suboptimal algorithm that relies on hard decisions (LogCut). By varying the decision splits (label grouping) and combining the resulting labelings using fusion moves, low energy MRF labelings are obtained very quickly.
- **FusionFlow algorithm: proposals from continuous-valued algorithms.** In this case the proposals come from different algorithms that work in a continuous label space and furthermore have complimentary strengths and weaknesses (Lukas-Kanade and Horn-Schunck flow estimators). By combining such proposals using fusion moves, low energy solutions are obtained for MRFs with continuous unbounded label spaces that are not easily amenable for dense discretization.
- **BP-fusion algorithm: proposals from non-convergent message passing.** Here, the proposals come from different iterations of min-sum belief propagation, which on itself does not converge. By combining such proposals with fusion moves, low energy labelings for a hard, highly-connected MRF are obtained within a few iterations.

These algorithm examples not only show that the fusion move extends the applicability of graph cuts to domains where they are either hard to apply (optical flow estimation), or lead to poor solutions (label placement). But moreover, they show that the ability to fuse problem specific proposals helps to obtain solutions with lower energy than competing methods such as alpha expansion. Finally, the application examples showed that fusion moves can facilitate runtime performance increases and allow the parallelization of the popular expansion move techniques into several threads to take advantage of modern multi-core architectures.

TABLE 6
The table summarizes the average degree of a node, number of non-submodular pairwise terms, and number of nodes unlabeled by QPBO-graph cut in a fusion move. The latter was very small throughout all our experiments.

Experiment	av. deg. node	non-submod. terms	unlab. nodes
Parallelized α-exp.	4	$< 0.1\%$	$< 0.05\%$
Iterated LogCut	4	$< 2\%$	$< 0.4\%$
FusionFlow	8	$< 20\%$	$< 0.1\%$
Label placement	115	$< 0.7\%$	0

In summary, the inherent flexibility of fusion moves allowed us to combine problem-specific proposal solutions, which makes the fusion approach suitable for a large variety of optimization problems associated with pairwise MRFs.

QPBO performance within the Fusion Move is summarized in table Tab. 6. It has been observed [18] that the crucial factors for the performance of QPBO are (i) connectivity of the graph, (ii) percentage of non-submodular terms, and (iii) strength of the unary terms. For each application we list the first two factors in the table. We observe that even for higher connected graphs (label placement) and a considerable percentage of non-submodular terms, QPBO performs very well in all of our experiments. Hence there is no need to consider more advanced solvers. On the other hand, the fusion is an NP hard problem and QPBO may not always be sufficient. For instance in [19] it has been demonstrated that for particular MRFs with triple cliques, QPBO-based fusion produces a considerable number of unlabeled nodes. In such a case, additional post-processing heuristics or other solvers may be need to improve the labeling, such as suggested in e.g. [17]–[19].

Proposal generation. In the end, we would like to discuss the general aspects of applying the fusion move idea to new problems. In this paper, we demonstrate various different ways of creating *problem-specific* proposals. Is there a generic recipe for the choice of proposals? Qualitatively, there are two aspects of the proposal set that is relevant to the success of the fusion approach: *quality* of individual proposals and *diversity* among different proposals. The quality of individual proposals ensures that the convergence to a good solution is faster (cf. the LogCut algorithm where fusing problem-optimized proposals yields faster convergence then fusing constant proposals within alpha-expansion). The quality of individual proposals also serves as a safety net, as the energy of the fused result cannot be higher than the energy of the best solution. The diversity between the proposals determines how much can be added by the fusion move on top of the best solution. Note that this *quality-diversity* issue is a direct analogy with the ensemble classifier creation methods in machine learning such as bagging.

To illustrate these concepts, consider the alpha-

expansion algorithm, that works with different constant proposals. Such solutions are quite diverse, but they are not of high-quality in terms of energy. As a result, alpha-expansion typically produces a low-energy solution which is much better than any of the proposals, but requires an excessive number of proposals (hence being relatively slow). Furthermore, we have shown that extending the diversity of proposals beyond constant labelings allows to obtain even lower energy solutions than those obtained with alpha-expansion. (The iterated LogCut algorithm provides an example of choosing a diverse set of solutions. Here, picking the next proposal so that the distance (10) between its shift s and previous shifts is maximized is aimed at increasing the diversity of the solutions.)

Some generic approaches for generating the diverse set of proposals include taking multiple local minima of the MRF energy obtained either via gradient descent or a greedy algorithms such as iterated conditional modes initialized with different starting points (this avenue is investigated in [41]). Finally, another idea for future research is to generate the proposals by sampling directly from the MRF, e.g. using Monte-Carlo or Gibbs sampling. This strategy, which we tested only briefly, however, faces two problems: firstly, obtaining the diverse set of solutions requires running the sampling process for a very long time, and, secondly, samples from the MRF probability may not be closely related to the MAP solution that is sought.

Another potentially useful idea that is not investigated in the paper, is to generate proposals on-the-fly by taking into account the current solution and its "problematic" parts (similar to boosting, where next weak classifier is picked by observing where the current strong classifier fails to make good prediction). The open questions are, how to define the meaning of a "problematic" part, and how to ensuring that there is enough diversity in the proposals.

REFERENCES

[1] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for Markov random fields with smoothness-based priors," *TPAMI*, vol. 30, no. 6, 2008.

[2] P. L. Hammer, "Some network flow problems solved with pseudo-boolean programming," *Operations Research*, vol. 13, 1965.

[3] D. Greig, B. Porteous, and A. Seheult, "Exact MAP estimation for binary images," *Journal of the Royal Statictical Society, Series B*, vol. 51, no. 2, pp. 271–279, 1989.

[4] E. Boros and P. L. Hammer, "Pseudo-boolean optimization," *Discrete Applied Mathematics*, vol. 123, no. 1-3, pp. 155–225, 2002.

[5] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" *TPAMI*, vol. 24, no. 2, 2004.

[6] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," in *Exploring Artificial Intelligence in the New Millennium*, G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann Pub., 2003, ch. 8, pp. 239–236.

[7] H. Ishikawa, "Exact optimization for Markov random fields with convex priors," *TPAMI*, vol. 25, no. 10, pp. 1333–1336, 2003.

[8] D. Schlesinger and B. Flach, "Transforming an arbitrary minsum problem into a binary one," Tech. Rep. TUD-FI06-01, 2006.

[9] P. Kohli, A. Shekhovtsov, C. Rother, V. Kolmogorov, and P. Torr, "On partial optimality in multilabel MRFs," in *ICML*, 2008.

[10] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *TPAMI*, vol. 23, no. 11, 2001.

[11] O. Veksler, "Efficient graph-based energy minimization methods in computer vision," Ph.D. dissertation, Cornell University, 1999.

[12] N. Komodakis, G. Tziritas, and N. Paragios, "Fast, approximately optimal solutions for single and dynamic MRFs," in *CVPR*, 2007.

[13] O. Veksler, "Graph cut based optimization for MRFs with truncated convex priors," in *CVPR*, 2007.

[14] V. Kolmogorov and C. Rother, "Minimizing non-submodular functions with graph cuts — A review," *TPAMI*, vol. 29, no. 7, pp. 1274–1279, 2006.

[15] V. Lempitsky, C. Rother, and A. Blake, "LogCut - Efficient graph cut optimization for Markov random fields," in *ICCV*, 2007.

[16] V. Lempitsky, S. Roth, and C. Rother, "FusionFlow: Discrete-continuous optimization for optical flow estimation," in *CVPR*, 2008.

[17] E. Boros, P. L. Hammer, and G. Tavares, "Preprocessing of unconstrained quadratic binary optimization," Tech. Rep. RUTCOR RRR 10-2006.

[18] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer, "Optimizing binary MRFs via extended roof duality." in *CVPR*, 2007.

[19] O. J. Woodford, P. H. S. Torr, I. D. Reid, and A. W. Fitzgibbon, "Global stereo reconstruction under second order smoothness priors," in *CVPR*, 2008.

[20] V. Kolmogorov and M. J. Wainwright, "On the optimality of tree-reweighted max-product message-passing," in *UAI*, 2005.

[21] Y. Boykov, O. Veksler, and R. Zabih, "Markov random fields with efficient approximations," in *CVPR*, 1998, pp. 648–655.

[22] V. Kwatra, A. Schödl, I. A. Essa, G. Turk, and A. F. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 277–286, 2003.

[23] C. Rother, S. Kumar, V. Kolmogorov, and A. Blake, "Digital tapestry," in *CVPR (1)*, 2005, pp. 589–596.

[24] J. M. Winn and J. Shotton, "The layout consistent random field for recognizing and segmenting partially occluded objects," in *CVPR (1)*, 2006, pp. 37–44.

[25] O. J. Woodford, I. D. Reid, P. H. S. Torr, and A. W. Fitzgibbon, "On new view synthesis using multiview stereo," in *BMVC*, 2007.

[26] A. Billionnet and M. Minoux, "Maximizing a supermodular pseudo-boolean function: A polynomial algorithm for supermodular cubic functions," *Discrete Applied Mathematics*, vol. 12, no. 1, pp. 1–11, 1985.

[27] D. Scharstein and C. Pal, "Learning conditional random fields for stereo," in *CVPR*, 2007.

[28] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," in *CVPR*, vol. 1, 2004, pp. 261–268.

[29] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *TPAMI*, vol. 28, no. 10, pp. 1568–1583, 2006.

[30] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," in *ICCV*, 2007.

[31] S. Roth and M. J. Black, "On the spatial statistics of optical flow," *IJCV*, vol. 74, no. 1, pp. 33–50, 2007.

[32] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert, "Highly accurate optic flow computation with theoretically justified warping," *IJCV*, vol. 67, no. 2, pp. 141–158, Apr. 2006.

[33] W. Trobin, T. Pock, D. Cremers, and H. Bischof, "Continuous energy minimization via repeated binary fusion," in *ECCV*, 2008.

[34] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI*, 1981, pp. 674–679.

[35] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1–3, pp. 185–203, Aug. 1981.

[36] S. Birchfield, B. Natarjan, and C. Tomasi, "Correspondence as energy-based segmentation." vol. 25, no. 8, pp. 1329–1340, 2007.

[37] J. Besag, "On the statistical analysis of dirty pictures," vol. B-48, no. 3, pp. 259–302, 1986.

[38] C. E. Rasmussen, "`minimize.m`," http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize/, Sep. 2006.

[39] A. Wolff, "The map-labeling bibliography." http://i11www.iti.uni-karlsruhe.de/%7Eawolff/map-labeling/bibliography/.

[40] J. Christensen, J. Marks, and S. Shieber, "An empirical study of algorithms for point-feature label placement." *ACM Transactions on Graphics*, vol. 14, no. 3, pp. 203–232, 1995.

[41] H. Y. Jung, K. M. Lee, and S. U. Lee, "Toward global minimum through combined local minima," in *ECCV*, 2008.