LogCut - Efficient Graph Cut Optimization for Markov Random Fields

Victor Lempitsky
Moscow State University
victorlempitsky@gmail.com

Carsten Rother
Microsoft Research Cambridge
carrot@microsoft.com

Andrew Blake
Microsoft Research Cambridge
ablake@microsoft.com

Abstract

Markov Random Fields (MRFs) are ubiquitous in low-level computer vision. In this paper, we propose a new approach to the optimization of multi-labeled MRFs. Similarly to α-expansion it is based on iterative application of binary graph cut. However, the number of binary graph cuts required to compute a labelling grows only logarithmically with the size of label space, instead of linearly. We demonstrate that for applications such as optical flow, image restoration, and high resolution stereo, this gives an order of magnitude speed-up, for comparable energies. Iterations are performed by "fusion" of solutions, done with QPBO which is related to graph cut but can deal with non-submodularity. At convergence, the method achieves optima on a par with the best competitors, and sometimes even exceeds them.

1. Introduction

Markov Random Fields (MRFs) are ubiquitous in low-level computer vision, finding applications in image restoration, stereo matching, texture analysis, segmentation and elsewhere. Here we are concerned with energy minimization in multi-label problems, such as stereo where the labels represent disparities, or image restoration where the labels are discrete intensity levels. A powerful technique for optimization with binary labels is graph cut [5] and this can be extended approximately to multi-label problems, using so-called α-expansion [3]. Unfortunately, α-expansion has intrinsically linear time-complexity, since it visits labels exhaustively. On the other hand, in numerical optimization it is common to exploit the regularity of simple energy functions to avoid exhaustive search of label space — *e.g.* binary subdivision in one-dimension. This suggests that MRF energies might be optimized in better than linear time, once the label space has some structure.

The *LogCut* algorithm proposed here deals with the label space by binary subdivision rather than label by label. Instead of applying graph cut once for each possible value of the labels, it is applied at successive bit-levels, starting with the most significant. Thus the time-complexity of LogCut, for a single sweep through the labels, is logarithmic in the size of the label set rather than linear, as figure 1 shows. Note that, in addition to computation time for graph cut as shown, it is necessary to prepare by computing the values of unary potentials in the MRF. For certain models (see later) this turns out also to incur only logarithmic complexity. Generally, there may be a linear-time overhead but, in practice, it is typically insignificant compared with the computation time for graph cut.

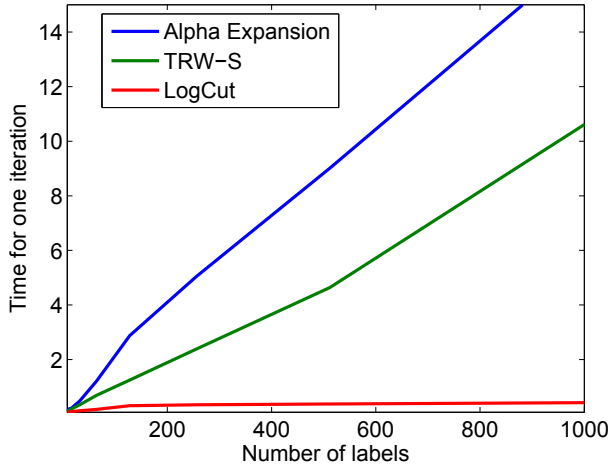


Figure 1. **Each iteration of the LogCut algorithm has logarithmic complexity.** A single iteration (sweep) of α-expansion has time-complexity that is linear in the size of the label space, since each label value must be visited once. Similarly another competitive algorithm, TRW-S, has linear complexity. However for LogCut, because of its divide and conquer approach in label space, execution time grows much more slowly (nearly logarithmically) with the size of label space. Results here are averaged over a test data set for the application of image restoration.

Importantly, such a binary subdivision process makes hard decisions about more significant bits before processing less significant bits. To compensate for this, we propose to reiterate the LogCut algorithm by using different hierarchical partitions of the label space, corresponding to different bit codings of the labels, and leading to different solutions. These solutions are combined in an optimal way via additional graph cut operations, which we call *fusion moves*.

To summarize, the paper develops the LogCut algorithm in terms of three new ideas: binary subdivision of label-space, learned bitwise approximation of pairwise potentials and fusion of successive sweeps. The effectiveness of LogCut, especially in reducing computation time and also in achieving good approximate optimization of MRF energy, is validated against three important vision problems with structured label spaces: image restoration, stereo matching and optic flow computation.

2. Energy Minimization

The form of optimization problem considered here is the minimization of an energy E, defined over a set of integer labels $\mathbf{x} = (x_1, \ldots, x_N)$ at nodes $\mathcal{V} = \{1, \ldots, N\}$. The nodes form a graph with a neighbourhood structure defined by the set of arcs $\mathcal{E}$. Labels take integer values $x_m \in \{1, \ldots, K\}$. The energy E is assumed decomposable into a sum of functions on nodes and edges:

$$E(\mathbf{x}) = \sum_{m \in \mathcal{V}} f_m(x_m) + \sum_{(m,n) \in \mathcal{E}} \gamma_{m,n} g_{m,n}(x_m, x_n). \quad (1)$$

Here the *regularization parameter* $\gamma_{m,n}$, adjusts the balance between unary and pairwise terms. In some models the spatial variation of $\gamma_{m,n}$ is data-dependent; alternatively it may be spatially constant so $\gamma_{m,n} = \gamma$. In this paper we consider MRFs with a simple four-connected neighbourhood system, which is however not a restriction imposed by our optimization method. Such problems are central to Markov Random Field (MRF) approaches to key vision problems such as image restoration [7], stereo matching [3], surface reconstruction [1] and image segmentation [2]. It is also assumed that while unary potentials are data-dependent, the pairwise terms may have data-dependence in the $\gamma_{m,n}$ but not in the $g_{m,n}(x_m, x_n)$ terms.

We make the following choices of energy functions for experiments. There is no restriction on the unary f_m. For the pairwise term we use a truncated power law $g = \mathrm{trunc}(\lambda)$, that is: $g(x, x') = \min((x - x')^p/\lambda, 1)$, taking the quadratic case $p = 2$. In the particular case $\lambda = 1$, we obtain the well known *Potts* model. There are now two unfixed parameters of the model, γ, λ, against which we will investigate the performance of our algorithm.

Binary MRF optimization For a binary label space, $x_m \in \{0, 1\}$, the optimization problem above is solvable exactly by graph cut [13] provided the energy E is *submodular*. Submodularity in this case boils down to the condition that all the pairwise terms are diagonally subdominant, that is $g(0,1) + g(1,0) \geq g(0,0) + g(1,1)$. If the condition is not met, alternative algorithms are available. For example *truncated* graph cut [17] modifies the energy where needed to ensure that submodularity holds. Alternatively Quadratic Pseudo-Boolean Optimization (QPBO) [8] gives a solution which is exact (for boolean optimization), but is not guaranteed to give complete solutions — that is, the labelling may be unknown at certain points on the graph. As we will see for our applications it usually turns out to be known almost everywhere.

Multi-label MRF optimization In the more general setting of integer labels $x_m \in \{1, \ldots, K\}$, graph cut does not apply directly, and exact methods are not available (apart from very special cases *e.g.* when pairwise terms are convex [9]). Effective approximate methods are known however, and they include α-expansion [3], Belief propagation (BP) [16] and Tree-Reweighted message passing (TRW-S) [10]. Message passing techniques (BP and TRW-S) apply to any energy function of the form of (1) but are restricted to local moves at each of its iterations. TRW-S optimizes a lower bound on the energy, and for binary label case it is guaranteed to converge to the global optimum under certain conditions (see [12]). The α-expansion algorithm [3] involves global moves based on binary graph cut and typically converges to low energy solutions in a few sweeps over label space.

2.1. The α-expansion algorithm

Since the ideas of iterative α-expansion play a central role in this paper we review it in more detail. An α-expansion step takes a suboptimal solution $\mathbf{x} = (x_1, \ldots, x_N)$ to the optimization problem and replaces it with an improved solution $\mathbf{x}^*$, by either keeping the original label or replacing it with a fixed label α, at each pixel location. The optimization is $\min_{\mathbf{y}} E(\mathbf{x})$ where $x_m = (1 - y_m)x_m + y_m\alpha$. At each step, the value of E decreases (or stays the same). By visiting all label values α in some sequence, and repeating the sequence, E is approximately optimized. In order for the α-expansion step to be solvable by graph cut, the binary optimization with respect to $\mathbf{y}$ must meet the submodularity requirement or alternative methods (see above) must be used. For instance, the Potts model meets the condition, but the truncated quadratic does not.

2.2. Generalized α-expansion: The fusion move

One new idea of this work is a generalization of α-expansion to combine two solutions to the optimization problem. Suppose $\mathbf{x}'$ and $\mathbf{x}''$ are each trial solutions to the optimization problem. Then the *fusion move* is a binary optimization that fuses two solutions, giving the combined solution $\mathbf{x}$ that minimizes $E(\mathbf{x})$ as before, but where now the auxiliary binary variables $\mathbf{y}$ are:

$$x_m = (1 - y_m)x'_m + y_m x''_m, \quad (2)$$

switching between two solutions $\mathbf{x}'$ and $\mathbf{x}''$. Note that an α-expansion move, an $\alpha - \beta$ swap move and a jump move, are special cases of a fusion move, *e.g.* in an α-expansion move

Figure 2. **Parallelized α expansion based on fusion move**. Stereo matching for the Tsukuba data set. Solutions 1 and 2 are computed by two α-expansions, done in parallel, on the lower and upper halves of the label space. They are combined by fusion into a solution with an energy much below either solution 1 or 2, and on a par with α-expansion. (note that QPBO had no unlabelled nodes in in this example.)

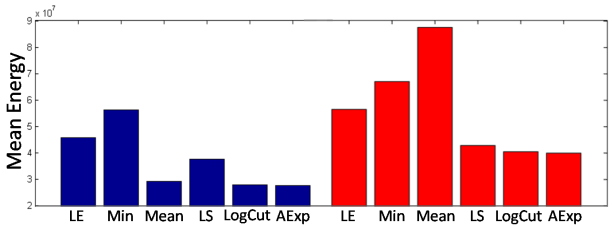


Figure 3. **Comparing various algorithms with logarithmic complexity.** Restoration of images under two different models: truncated quadratic (blue) and near-Potts (red). For each algorithm (see below for definitions of algorithms), the result of just a *single* iteration is shown. LogCut, as proposed in the paper, performs close to α-expansion (AExp), and is the only algorithm to do so, consistently, across both models. (See section 5 for details of the restoration problem and the two models.)

one of the two solutions is the constant image $x''_m = \alpha$. One application of the fusion move is parallelized α-expansion in which each process deals with a subset of labels, and the results are then fused. The idea is illustrated in figure 2.

The fusion problem is generally non-submodular, no matter what the form of the pairwise potentials may be. Therefore, for the binary optimizer in fusion moves, we use QPBO. If QPBO returns unlabeled nodes, there are several available strategies to guarantee that the energy does not go up. The fastest solution is to take the labels of all unlabeled nodes from either solution $\mathbf{x}'$ or $\mathbf{x}''$, whichever has the lower energy. (as a direct consequence of the "persistency" property [11] the resulting solution is guaranteed to have an energy less or equal than the energies of $\mathbf{x}'$ and $\mathbf{x}''$). In practice, we found that for 4-connected vision MRFs the number of unlabelled nodes with fusion move is typically very low (*e.g.* $< 1\%$).

3. Algorithms with Logarithmic Complexity

Although α-expansion is computationally effective, its computational complexity is an issue in that it grows linearly with the size K of the label space, and this is because each label k must be visited once in an iteration. A natural approach to overcome this problem is to deal with label space hierarchically, via divide-and-conquer. If label-space is partitioned successively according to values of binary bits $b = 1, \ldots, B$, with $B = \log K$, starting with the most significant, the labelling of a given pixel can be achieved by executing a tree of binary classification steps. In principle this should achieve logarithmic computational complexity, that is a speed-up of order $K/\log K$, and experimental results (section 5) will bear this out.

In the following we introduce various algorithms based on different approximation schemas. The performance of the different algorithms for one particular test application (image restoration) is shown in fig 3. The best performing algorithm — a trained binary subdivision algorithm, introduced in section 3.4, is the algorithm we term *LogCut* that is proposed in this paper.

3.1. Simple Binary Subdivision

Logcut achieves log-time complexity by representing the integer value at each pixel m as a B-bit word:

$$x_m = \sum_{b=0}^{B-1} x_m^b \, 2^b. \tag{3}$$

This can be seen as a binary optimization problem with $N \log(K)$ variables, effectively an MRF with large cliques each of maximum size $\log(K)$. This would appear to be intractable to optimization by conventional methods. Therefore, retaining the original small-clique MRF structure, the optimization problem is to be solved by iterating over the bit-array $\mathbf{x}^b = (x_1^b, \ldots, x_m^b)$. This starts with the most significant bit $b = B - 1$ and sweeps down to the least significant, $b = 0$. At each level, binary optimization could be solved by graph cut, in principle, though submodularity is an issue. Then, instead of needing a number of iterations that grows linearly with N the number of levels, as for α-expansion, only a logarithmic number of steps is required.

However there is a problem. Consider an intermediate step, when optimization with respect to bits $b+1, \ldots, B-1$ has been done, and now it is time to optimize with respect to bit x_m^b at each pixel. While this optimization is occurring, to what value should the remaining less significant bits

$x_m^{b-1}, \ldots x_m^0$ be set? They could simply be set to zero. We call this simple strategy, when applied repeatedly to successive bitplanes, Log Expansion (LE). In practice (figure 3), Log Expansion is far less effective than α-expansion.

3.2. Lower bound approximations

Rather than simply setting less significant bits to zero, an alternative idea is to use a lower bound. This means when we optimize over a bit-level b (assuming levels $b+1, \ldots, B-1$ were already considered), an approximate energy E^b is used, defined over the new bit variables, as follows:

$$E^b = \sum_{m \in \mathcal{V}} f_m^b(x_m^{b:B-1}) + \gamma \sum_{(m,n) \in \mathcal{E}} g_{m,n}^b(x_m^{b:B-1}, x_n^{b:B-1}), \tag{4}$$

where $x_m^{b1:b2}$ denotes $(x_m^{b1}, \ldots, x_m^{b2})$, and

$$f_m^b(x_m^{b:B-1}) = \min_{x_m^{0:b-1}} f_m(x_m) \text{ subject to } x_m^{b:B-1} \text{ fixed}. \tag{5}$$

Note that, in the case that the unaries are a known function, for example a quadratic of the form $(x-d)^2$ as in image restoration (see later), the minimization (5) can be done in constant time, rather than time proportional to 2^b. This reduces the formal complexity of LogCut from linear to logarithmic in label-set size. In practical settings, for general unaries, the (linear) time to evaluate (5) is in any case dominated by the logarithmic time for graph cut. Similarly, for the pairwise terms,

$$g_{m,n}^b(x_m^{b:B-1}, x_n^{b:B-1}) = \min_{\{x_m^{0:b-1}, x_n^{0:b-1}\}} g_{m,n}^b(x_m, x_n) \tag{6}$$

subject to $x_m^{b:B-1}$, $x_n^{b:B-1}$ being fixed. It is straightforward to show that this is indeed a lower bound, that is:

$$\min_{\mathbf{x}^{b:B-1}} E^b(\mathbf{x}^{b:B-1}) \leq \min_{\mathbf{x}} E(\mathbf{x}). \tag{7}$$

It is also clear that as $\gamma \to 0$ (approaching the trivial problem in which pairwise terms are switched off) the bound becomes exact. This is not the case for the earlier strategy of setting lower order bits to zero. However, despite the intuitive appeal of the lower bound strategy, it performs substantially worse than the simple LE algorithm above, as figure 3 shows (columns labelled *min*). The *min* operation used in the definition can be replaced with other operations, and specifically we consider the *mean* approximation constructed in this way. Figure 3 shows that *mean* also performs far worse than α-expansion.

3.3. Least squares approximation

One could argue for various other forms of approximation, but instead we propose to introduce greater flexibility by *learning* the approximation to the pairwise potential. One approach to learning is to construct a least-squared error approximation $g_{m,n}^b$ to $g_{m,n}$ at each bit level. Taking $g_{m,n}$ to be a function of the difference $\Delta x_{m,n} = x_m - x_n$, gives the approximations

$$g_{m,n}^b = \sum_{\Delta \mathbf{x}^{b+1:B-1}} P(\Delta \mathbf{x}^{0:b} \mid \Delta \mathbf{x}^{b+1:B-1}) g_{m,n}(\Delta x_{m,n}). \tag{8}$$

The expectation is computed from training data consisting of a separate set of images for which the optimal solution $\mathbf{x}$ has been estimated by a "reference" method, taken here to be the best algorithm, TRW or α-expansion depending on energy model. Figure 3 shows (see results labelled LS) that this form of learned approximation is an improvement over previous approximations (LE, min, mean) but still not uniformly as good as α-expansion.

3.4. Trained parametric approximation

Availability of training data also suggests another way of choosing approximate pairwise potentials $g_{m,n}^b$. Given a set of pairwise potentials $G = \{g_{m,n}^b\}_{b=1..B}$, the sequential computation of bit levels on a training data-set, using these approximate pairwise potentials, produces a solution $\mathbf{x}(G)$. (Training data, for image reconstruction for instance, typically consists of several training images, to avoid over-learning.) It is natural to seek for the potential set G that results in the smallest training energy $E(\mathbf{x}(G))$. In models used here, where $g_{m,n}$ is a truncated quadratic trunc_λ, the approximated, pairwise potentials can be represented in a similar parametric form:

$$g_{m,n}^b(.) = \mu^b \, \mathrm{trunc}_{\lambda^b}, \;\; 0 \leq \mu^b \leq 1, \;\; 0 \leq \lambda^b \leq \lambda. \tag{9}$$

In that way, the set G of approximated potentials is defined by a vector with $2B$ elements: $G = \{\lambda_1, \mu_1, \lambda_2, \mu_2, \ldots \lambda_B, \mu_B\}$. To optimize the training energy $E(\mathbf{x}(G))$, we iterate over bit levels. While visiting bit level b the vector elements corresponding to all other bit levels are fixed and $E(\mathbf{x}(G))$ is minimized with respect to μ_b and λ_b. (The evaluation of $E(\mathbf{x}(G))$ is done, as previously, by a single sweep of QPBO through the bit levels.) For the 2D search over (μ_b, λ_b) exhaustive search is feasible, given the limited range of both variables, but the simplex method [15] is more efficient. Typically, the optimization process converges after $2-3$ iterations over all bit levels. Initial values have to be chosen for G (for example from optimization using the "min" approximation above). In practice, we have found that the value of $E_{train}(\mathbf{x}(G))$ achieved after optimization is largely independent of those initial values.

Details of the performance of LogCut optimization of test data, using learned pairwise potentials as above, are given in succeeding sections. For now, note (figure 3) that its performance is very close to one iteration of α-expansion. Its computation time however, as we saw in figure 1, is very much shorter.

4. Iterated LogCut Algorithm

Irrespective of the choise of binary potentials, LogCut has to make hard desicions about more important bits before going to bits with lower importance. To make the algorithm robust to the errors made by these hard desicions, different bit codings may be considered, and this is done by the iterated LogCut algorithm. This algorithm starts with the solution $\mathbf{x}_0$ obtained by applying a single LogCut sweep as before. Then in each subsequent iteration, a shift s is introduced and applied to label values. The effect is that the binary coding (3) is now applied to $x + s \ (mod\, K)$ rather than to x, and the result of a single sweep in the shifted label space is denoted $\mathbf{x}_s$. The current solution is then fused with the $\mathbf{x}_s$ to give a new solution with decreased energy.

The effectiveness of iterated LogCut depends on the particular choice of label shifts s. It is in fact quite effective to select a shift s randomly but results with the following *Maximum Regrouping* strategy turn out to be a little more consistent. Define the *regrouping distance* between shifts to be:

$$r_b(s_1, s_2) = \frac{1}{2} - \left| \frac{|s_1 - s_2| \mod 2^b}{2^b} - \frac{1}{2} \right|, \qquad (10)$$

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, for each of the two 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 maximal regrouping of labels. Total regrouping distance is naturally defined as a sum over bit levels:

$$r(s_1, s_2) = \sum_{b=1}^{B} r_b(s_1, s_2). \qquad (11)$$

Now, at each iteration of LogCut, the shift that is the most distant from all previous shifts, according to (11), is chosen. This encourages the maximum diversity amongst the solutions to be fused. These fused iterations prove effective in optimizing energy further than is possible with just a single iteration and results are given later in section 5. The entire iterated LogCut algorithm is summarized in figure 4.

Importantly, iterated LogCut spends most of the time on obtaining solutions rather than fusing them. Consequently, the algorithm may be efficiently parallelized: the solutions for different s may be computed in parallel on different processors and fused afterwards.

5. Experiments

In this section we validate experimentally the performance of the iterated LogCut algorithm in three different domains: image restoration, stereo matching and optic flow computation. These domains exhibit significant gains in computational efficiency for LogCut compared with α-expansion, efficient BP, and TRW-S. The gains become more marked as the size of label space increases, and so the highest gains occur for optic flow, where the vector nature of the labels requires a large (*e.g.* 10-bit) label space.

Training
Define E^b for each bit-level b (4)
with f^b (5) and g^b trained (sec. 3.4)

Optimization
Initialization
Optimize E using one trained LogCut sweep $\rightarrow$ $\mathbf{x}_0$

Iteration
Run until convergence (*i.e.* until $\mathbf{x}_t = \mathbf{x}_{t-1}$):
  Pick a shift s (sec. 4)
  Obtain labeling $\mathbf{x}$ using shift s and
    trained LogCut sweep (sec. 3.1)
  $\mathbf{x}_t \rightarrow$ fuse($\mathbf{x}_{t-1}, \mathbf{x}$) (sec. 2.2) (using QPBO)
x_t at convergence is the final solution

Figure 4. **A summary of the LogCut algorithm**

Applications and Data Sets. Fig 5 shows typical test and training images used with LogCut. For image restoration we have used 10 training and 10 test images from the Corel database (approx. 240×160 gray scale images) which gives a label space of 8 bits. As in previous work [19, 6] we added Gaussian noise to all images and obscured portions of the image, creating areas where the unary is fixed to 0, and have therefore effectively to be inpainted. As above, we used either model 1: $\lambda = 200, \gamma = 2000$ or model 2: $\lambda = 4, \gamma = 5000$. For historical continuity, we also performed the restoration of the single penguin image, using exactly the same setup as in [19, 6] ($\lambda = 200, \gamma = 5000$).

For stereo matching, we have used 9 registered datasets recently introduced in [18]. To obtain good-looking results without intricate occlusion reasoning, we used a trinocular setup, where depth maps were recovered for middle images and the unaries were computed by taking minima of (sampling-insensitive, truncated SAD) matching costs with the left and the right images. For pairwise terms, we used $\lambda = 4$, which deals better with slanted surfaces than a pure Potts model. As is commonly done, we used edge-adaptive regularisation $\gamma_{m,n}$ in which if the color difference along an edge is large enough, $\gamma_{m,n} = 50$, otherwise $\gamma_{m,n} = 150$. The experiments were carried either at full resolution (image size = 1390×1110, 8-bit label space) or at downsampled resolution (image size = 458×356, 6-bit label space). In the former case, the sheer dataset size means that TRW-S is infeasible, and permits the use of only one training dataset (4 datasets were used for training at small resolution). Respectively, either 8 or 5 datasets were left for testing.

For the optical flow evaluation, we used a standard benchmarking Yosemite sequence (version with clouds). The motion vectors were discretized into 32×32 lattice

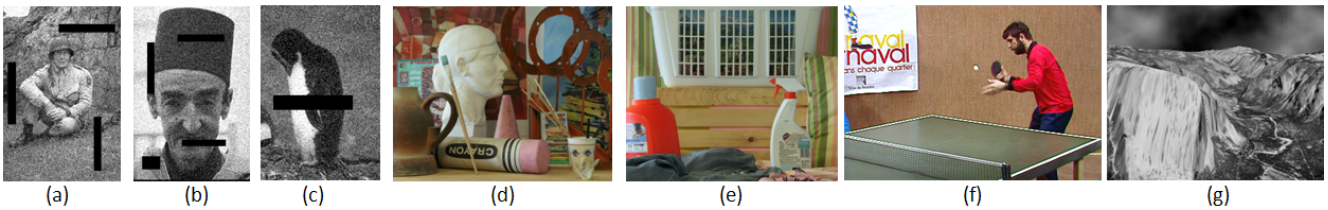


Figure 5. **Data sets.** Examples of training (a,d,f) and test (b,c,e,g) images for image restoration (a-c), high (and low) resolution stereo (d,e) and optical flow (f,g). Black areas in (a-c) mean obscured image portions.

Problem	Bits	Speed-up 1st iter	Energy diff 1st iter	Energy diff converg.
Low-res stereo	6	**4.9** (1.6)	+2.6%	-0.3%
High-res stereo	8	**9** (2.2)	+3.6%	-0.2%
Image restor. model 1	8	**12.4** (6.1)	+0.5%	-2.6%
Image restor. model 2	8	**11** (–)	+4.4%	+0.3%
Optical flow	10	**20.7** (10.3)	+2.5%	-2.4%

Figure 6. **LogCut is substantially faster than α-expansion.** The speed up is given as the ratio of computation times for α-expansion vs. LogCut, for one iteration. (In brackets we give time ratio based on running LogCut until its energy matches that after one iteration of α-expansion.) Energy differences (LogCut minus α-expansion) are shown after one iteration and at convergence. It is clear that the speed advantage of LogCut increases with the number of bits used for label-space.

(with $1/4$-pixel step). For the bit coding of labels, we interleave the bits corresponding to the horizontal and the vertical components. For unaries we used truncated SAD difference, while for pairwise terms we used 2D truncated quadratic potential $g(v, v') = \min(((v_x - v'_x)^2 + (v_y - v'_y)^2)/\lambda, 1)$ with $\lambda = 16$ and edge-adapting $\gamma = 150/300$. A single frame pair from a completely different dataset (Berkeley "table tennis") was used for training.

Overall Performance. We compared LogCut to α-expansion [3] and TRW-S [10], which are known to be the best methods for typical vision MRF optimization problems [19]. For image restoration problems, we also considered efficient BP [6] (authors' implementation). All algorithms were either run until convergence or stopped after a long runtime. If the size of the label space and images get large, for high resolution stereo and optical flow, TRW-S was infeasible to run due to working memory limitations.

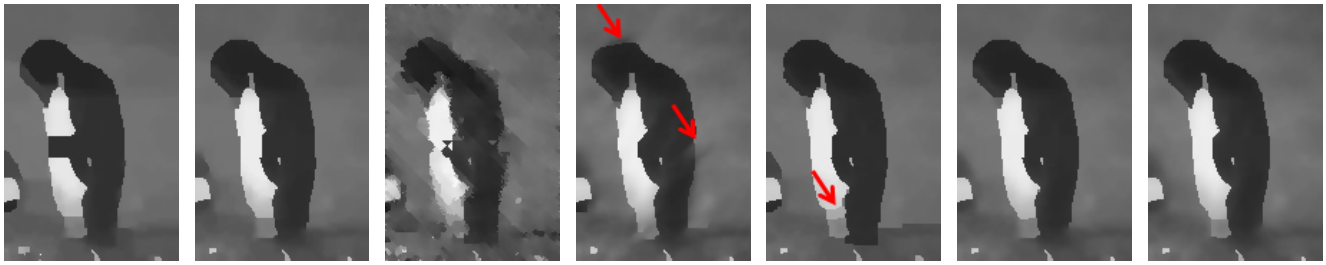
Non-submodular terms during α-expansion were handled by truncated graph cut [17] computed with maxflow algorithm [4] (QPBO was also tried but the marginal improvement in energy did not justify the increased computational expense). We also used random order of visiting different αs during each iteration of α-expansion as this was found to be superior over linear order. Note that our comparison did not consider the two very recent improvements of α-expansion [14, 20] ([14] suggested a way to accelerate α-expansion by the exploitation of a dual of the problem; however, it does not reduce the time required for the first sweep. [20] proposed a generalization of α-expansion based on *range moves*, which is capable of obtaining lower energy values at a price of longer runtime.)

The comparisons are presented in the form of time-energy plots (figure 10) and a comparison table (figure 6). All results are averaged over respective test sets. For a number of problems, we present selected resuls (figures 7, 8, 9). Note that the time required for MRF construction step (allocating initial structures, setting values for initial unary and pairwise terms) was not included in the statistics. As can be seen, LogCuts outperforms the competitors when terminated early in that, for a given degree of approximation of energy, LogCut requires substantially less computation time.

It is well known that α-expansion is especially well suited to optimizing Potts model energies [19]. This is because an expansion move proposes a constant label which attracts zero penalty under the Potts model. Even in the Potts regime, we see that LogCut is competitive (figure 10b). Away from the Potts regime, where α-expansion is weaker, LogCut has a substantial advantage (figure 10a,c,f).

The requirement for offline training in LogCut is indeed an additional burden. In mitigation, however, just a single training image has often proved sufficient (high-res stereo and optical flow experiments). In fact we have examined that for image restoration model 1 and 2, that in the extreme case of taking just one single training image, the highest energy on the test set (after one iteration) was only 1.5% higher than training on all training images. Secondly, training data does need not be especially similar to test data, so one set of trained potentials should suffice for a wide range of test data.

Which binary optimizer? We have chosen to use QPBO for all binary optimization problems, for the following reason. For the image restoration model 1, for all test images, the percentage of non-submodular terms was always below 3% during LogCut sweeps, and below 2% during a fusion step. For model 2, the percentages are lower still. For QPBO this always gave less than 0.003% unlabeled nodes during a LogCut sweep and under 0.4% during fusion. The result is that LogCut with QPBO consistently achieves lower energies than LogCut with truncated graph cut in comparable runtime.



LogCut(1 it) 0.5sec. E=161 | LogCut(3 it) 1.5sec. E=154 | Eff.BP(1 it) 2.1sec. E=266 | TRW-S(1 it) 2.3sec. E=161 | α-exp(1 it) 4.7sec. E=158 | LogCut(20 it) 9.6sec. E=152 | TRW-S(40 it) 90sec. E=151

Figure 7. **Image restoration of the penguin image.** Various result images of different optimization techniques ordered in term of runtime. LogCut achieves in the shortest runtime a visually pleasing result (second image). Red arrows point out incorrect reconstructions.

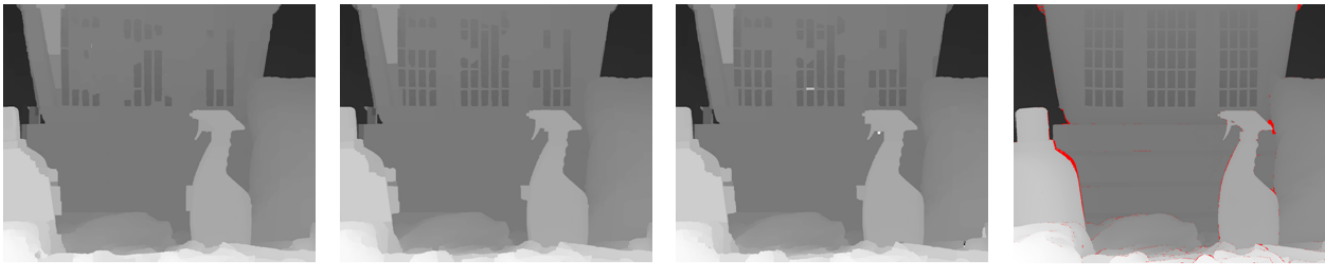
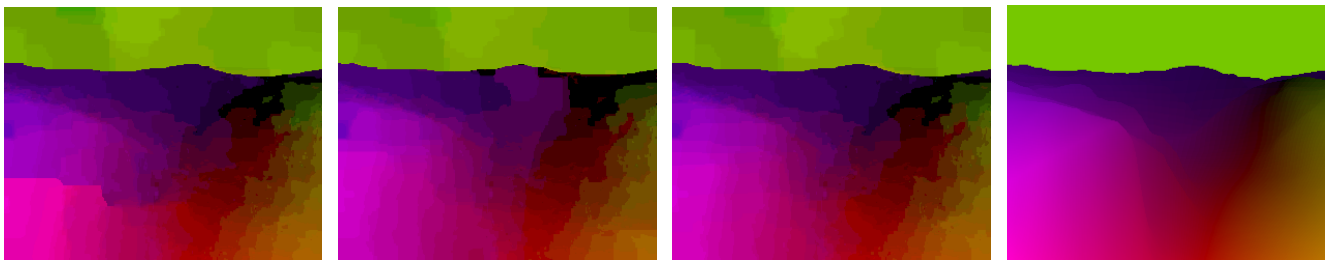


Figure 8. **High Resolution Stereo.** Different results of LogCut and α-expansion. The first result of LogCut recovers the main depth levels correctly. Running both LogCut and α-expansion for a long time improves results slightly (see pattern of the basket - original image in fig 5). The ground truth is seen from a slight different viewpoint where red pixels could not be computed.



α-exp (6 iter.) 390 sec. E = 8773 | LogCut (2 iter.) 8.0 sec. E = 8767 | LogCut (64 iter.) 150 sec. E = 8469 | ground truth *(sky motion is approximate)*

Figure 9. **Optical flow** fields for the last frame pair of Yosemite sequence (hue = direction, value = magnitude). α-expansion solution is very slow to compute and has erroneous sharp transitions. As LogCut proceeds, it outperforms final α-expansion result at 2nd iteration. After convergence it yields a smooth solution with much lower energy.

6. Conclusions and Discussion

The LogCut algorithm has proved to be an efficient algorithm for multi-label energy minimization, reducing time-complexity from linear in the size of label space to logarithmic. In practice this has achieved computation time reductions of up to an order of magnitude on various key problems in computer vision, compared with state of the art algorithms: α-expansion and TRW-S. Further speedup may be easly obtained by a parallelization of different LogCut iterations. In terms of closeness to optimality at convergence, LogCut is on a par with α-expansion and TRW-S or better, depending on the nature of the energy model.

In return for the advantage in computationally efficiency, LogCut requires its additional offline learning step. Firstly, this requires the provision of suitable training data, and secondly hits practical limitations on working memory, especially with larger training sets, and larger label spaces. It is hoped that future work may shed light on the structure of the learned pairwise potentials, in order to simplify or even eliminate the learning procedure.

In the models used in the paper, we have assumed a natural ordering of labels, in that they are integers representing physical quantities. It is an open question whether an effective learning procedure could be developed for unordered labels. Another area of possible extension is to MRF models with larger cliques, which are of interest for filter bank and patch models of images. This would depend on whether suitable parametric models could be found to represent approximate, learned potentials over these larger cliques.

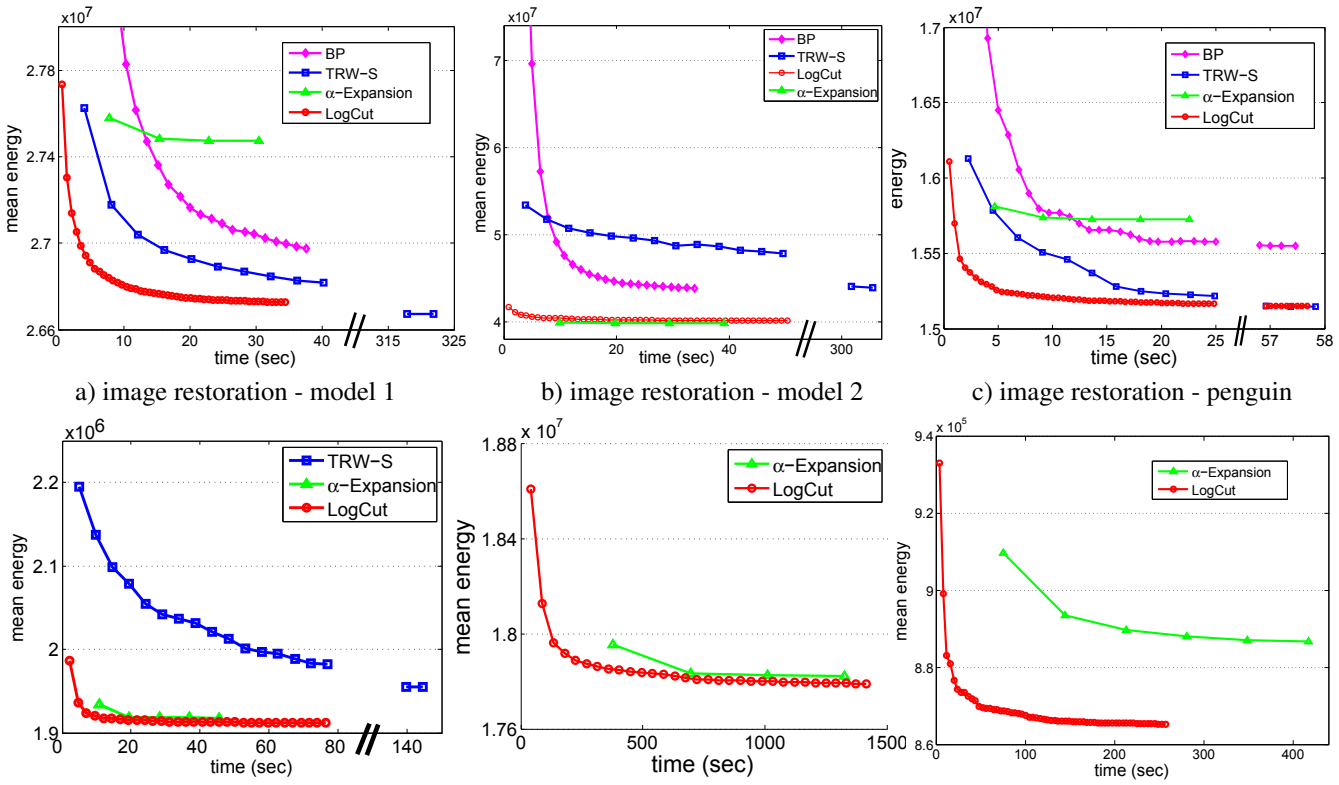


Figure 10. **Averaged time-energy plots** for different types of problems suggest that LogCut outperforms competitors when terminated early, requiring substantially less computation time to achieve equivalent energy. For stereo and optical flow, it even outperforms competitors in energy achieved at convergence. (Each marker on curves represents one iteration of optimization.)

References

[1] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, USA, 1987.

[2] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, pages 105–112, 2001.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *TPAMI*, 23(11), 2001.

[4] Y. Boykov and V.Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *TPAMI*, 26(9), 2004.

[5] D.M.Greig, B.T.Porteous, and A. Seheult. Exact MAP estimation for binary images. *J. Royal Statistical Society*, 51:271–279, 1989.

[6] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. In *CVPR*, 2004.

[7] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *TPAMI*, 6(6):721–741, 1984.

[8] P. Hammer, P. Hansen, and B. Simeone. Roof duality, complementation and persistency in quadratic 0-1 optimization. *Mathematical Programming*, 28:121–155, 1984.

[9] H. Ishikawa. Exact optimization for Markov Random Fields with convex priors. *TPAMI*, 25(10):1333–1336, 2003.

[10] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *TPAMI*, 28(10), 2006.

[11] V. Kolmogorov and C. Rother. Minimizing non-submodular functions with graph cuts — a review. *TPAMI*, 29(7), 2007.

[12] V. Kolmogorov and M. Wainwright. On the optimality of tree-reweighted max-product message passing. In *Conf. Uncertainty in AI*, 2005.

[13] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *ECCV*, pages 65–81, 2002.

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

[15] J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1964.

[16] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, Palo Alto, 1988.

[17] C. Rother, S. Kumar, V. Kolmogorov, and A. Blake. Digital tapestry. In *CVPR*, 2005.

[18] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *CVPR*, 2007.

[19] S. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization algorithms for highly connected graphs. In *ECCV*, 2006.

[20] O. Veksler. Graph cut based optimization for MRFs with truncated convex priors. In *CVPR*, 2007.