J Real-Time Image Proc
DOI 10.1007/s11554-012-0308-z

ORIGINAL RESEARCH PAPER

# Gentle ICM energy minimization for Markov random fields with smoothness-based priors

**Zoran Zivkovic**

Received: 30 March 2012 / Accepted: 27 November 2012


**Abstract** Coordinate descent, also known as iterated conditional mode (ICM) algorithm, is a simple approach for minimizing the energy defined by a Markov random field. Unfortunately, the ICM is very sensitive to the initial values and usually only finds a poor local minimum of the energy. A few modifications of the ICM algorithm are discussed here that ensure a more 'gentle' descent during the first iterations of the algorithm and that lead to substantial performance improvements. It is demonstrated that the modified ICM can be competitive to other optimization algorithms on a set of vision problems such as stereo depth estimation, image segmentation, image denoising and inpainting.

**Keywords** Optimization · Markov random field · Image segmentation · Image denoising · Depth estimation

## 1 Introduction

Solving vision problems through energy minimization is a widely used approach. An energy function usually corresponds to a Markov random field (MRF) probabilistic model where each image pixel becomes a node of the MRF. The energy minimization becomes a maximum a posteriori estimation.

The energy functions for vision problems are often difficult to minimize. Many standard multivariate optimization techniques [8] have failed. Minimization by coordinate descent, also known as iterated conditional mode (ICM) algorithm [3], usually finds a poor local minimum as demonstrated many times [4, 16, 18]. Simulated annealing is highly inefficient [2]. Many heuristics have been tried since then [13]. Recently, promising results have been achieved by applying "message passing" and "graph cut"-based algorithms. The "message passing" algorithms were inspired by the well-known max-product (min-sum) belief propagation (BP) which finds the optimum for a MRF without "loops" [5]. Typical MRFs in computer vision have many loops, since all neighboring pixels are related to each other and BP does not guarantee the optimal solution but performs well in practice [20]. Sequential tree-reweighted message (TRW-S) passing has similar implementation, but it was constructed based on different principles [10]. The TRW-S is considered to be the best-performing message-passing algorithm [18]. The "graph-cut"-based algorithms are performing MRF minimization by defining it as a set of underlying binary labeling problems and repeatedly computing the global minimum of each of them. The two core strategies describing how to transform a MRF into labeling problems are "expansion" and swap" [6]. The labeling tasks are efficiently solved as a graph-cut and typically the max-flow algorithm is used for this part. Unfortunately, not all MRFs can be decomposed into sets of binary labeling problems and approximations are needed. An overview of the modern optimization algorithms and a comparison on a number of vision problems is given in [18].

Unfortunately, the modern techniques are linked with huge memory and computation costs. Message-passing algorithms can be much faster if the MRF energy function has specific form [14]. Various approximations were introduced to further reduce the computation costs for specific applications, e.g., inpainting in [12]. Approximations in combination with efficient hardware implementations are

Z. Zivkovic (✉)
NXP Semiconductors, Eindhoven, The Netherlands
e-mail: zoran.z.zivkovic@googlemail.com;
zoran.zivkovic@nxp.com

Published online: 13 December 2012

also considered [15, 19]. The graph-cut-based algorithms are recently improved by finding ways to reuse the results of previous graph-cut iterations [1, 11]. The first iteration remains the same, but the computation time for each subsequent iteration is less depending on the problem. These methods are interesting for video processing where there is no much change between two video frames.

In this paper, the ICM algorithm is reconsidered. The results are inspired by the highest confidence first (HCF) heuristic for performing ICM updates [7]. The HCF defines a confidence measure for each MRF node. ICM updates are performed by giving preference to the nodes with high confidence. However, the HCF still performs much worse than the modern algorithms [18]. Furthermore, HCF is computationally expensive, since it requires constant sorting of the nodes according to their confidence measure. Based on the same principles, more effective and efficient modifications of the ICM algorithm are introduced to limit the influence of the uncertain initial values during the first few iterations by modifying the MRF smoothness costs. Other heuristics for modifying the smoothness costs of a MRF were used before but with other purpose. For example, in [9] the smoothness costs are increased to speed up user-assisted alpha matting around the user-selected pixels.

The approach presented here significantly outperforms the standard ICM and HCF. The new modified ICM finds close-to-optimal solutions for a number of vision problems. If the real-time performance is the main concern, the new modified ICM might be a reasonable low-cost solution both in terms of memory and computation requirements.

The paper describes ICM and the proposed modifications first. A set of vision problems is considered and results compared to other algorithms as in [18]. The last section contains the conclusions.

## 2 Energy minimization using ICM

Let $x_c$ denote the variable that needs to be estimated for the $c$ th node of a MRF. In computer vision problems, typically each image pixel becomes a node of the MRF. Usually, discrete problems are considered where $x_c \in \{1, \ldots, L\}$. For example, in stereo depth estimation, $x_c$ value corresponds to the pixel depth value [2], and in image segmentation it is the pixel label [16]. An energy function typically has two parts: data terms to penalize solutions that are inconsistent with the observed data, and smoothness terms to enforce spatial coherence. The general form for a problem where we estimate variables for $N$ nodes and smoothness terms containing two nodes, is:

$$E(x_1, \ldots, x_c, \ldots, x_N) = \sum_{c=1}^{N} [D_c(x_c) + \sum_{j \in \mathcal{N}(c)} S_{cj}(x_c, x_j)] \tag{1}$$

where $D_c(x_c)$ is the data term for the $c$ th node, and $S_{cj}(x_c, x_j)$ are the pair-wise smoothness constraints for all nodes $j$ from some neighborhood $\mathcal{N}(c)$ of the $c$-th node. The energy (1) is a function of $N$ variables and in general it is difficult to find the minimum. An introduction to MRF representations in computer vision can be found in [13].

In minimization by coordinate descent, also known as ICM algorithm, each variable $x_c$ is considered separately:

$$x_c = \min_{x_c} E(x_1, \ldots, x_c, \ldots, x_N) = \min_{x_c} E_c(x_c), \tag{2}$$

where

$$E_c(x_c) = D_c(x_c) + \sum_{j \in \mathcal{N}(c)} S_{cj}(x_c, x_j) \tag{3}$$

denotes the local part of the energy function depending only on $x_c$ while the other variables are fixed. This is repeated for all the variables $c = 1, \ldots, N$ in a number of iterations until convergence. ICM converges fast to a local minimum [13]. If the number of iterations is $K$, ICM will run in $O(\mathrm{KNL})$ time. Since the label updates (3) are local, there is, in principle, no need for additional memory.

## 3 Highest confidence first updates

Iterated conditional mode energy minimization is highly sensitive to the initial values of the nodes and also to the order in which the nodes are updated. As an illustration, a simple 3-node MRF is constructed based on a stereo matching example from [18]. The variables $x_c$ are discrete disparities between the two images. The data terms and initial local energy terms are shown in Fig. 1. Minima of the data costs are used as initial values for the nodes. The ICM makes a decision at each node based only on the current local energy (3). For example, if node $a$ is updated first, the wrong initial value for the node $b$ has strong influence via the smoothness term, see $E_a(x_a)$ in Fig. 1. As a result, initially, the good value of $a$ is changed and ICM ends up in a poor local minimum. It can be shown that ICM will find the optimum, only if the node $b$ is updated first.

The Highest confidence first (HCF) algorithm by Chou and Brown [7, 13] starts by introducing a local energy confidence (stability) measure $C_c$ for each node. Let $x_c^*$ be the value where the minimum of the local energy (3) is achieved and let $x_c$ be the current value that the node $c$ has. The difference between the local energy for these two values

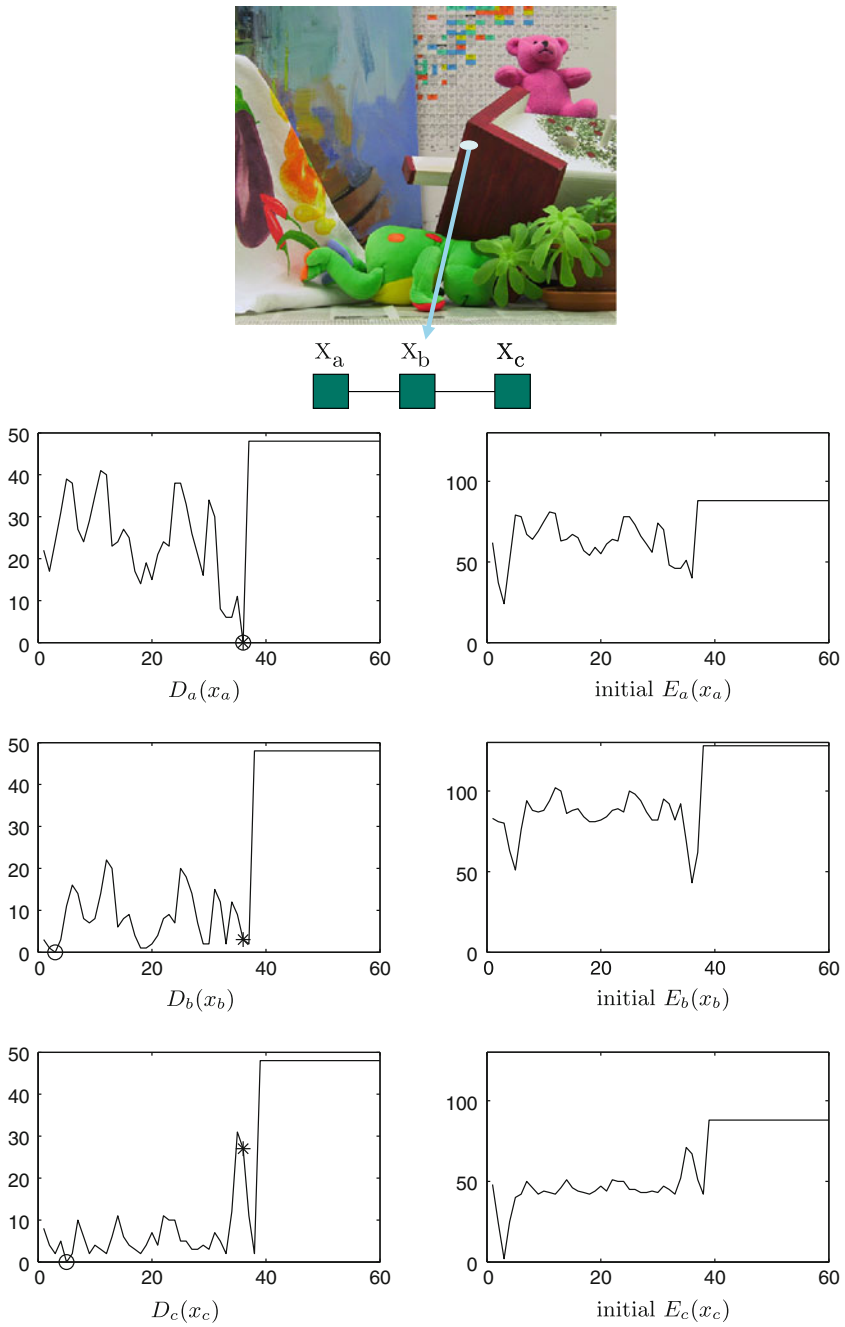



**Fig. 1** Simple 3-node MRF example. The data and smoothness terms are from a real stereo matching problem [18]. The *arrow* indicates the 3 pixels from left image in stereo matching pair that were used. The data terms of the nodes are plotted. The initial values are marked by (*open circle*). The optimal solution is marked by (*asterisk*). Next to the data terms, the initial values of the local energies $E_c$ used by ICM are plotted (3)

$$C_c = E_c(x_c) - E_c(x_c^*) \tag{4}$$

shows how much improvement is possible. The nodes with higher $C_c$ are updated first leading to a greedy ICM that prefers larger steps. In practice, this does not lead to large improvements for vision problems. For the example from Fig. 1, this would update node $c$ first, leading again to a poor solution.

Highest confidence first [7] contains the second, and in our view, more important modification of ICM algorithm. A label is added to each node to mark what will be called "uncommitted" nodes of the Markov field. Let this label be 0 and initially, all nodes are uncommitted, i.e., $x_i = 0$. The smoothness terms are replaced by a modified smoothness terms and the modified local energy function becomes:

$$E'_c(x_c) = D_c(x_c) + \sum_{j \in \mathcal{N}(c)} S'_{cj}(x_c, x_j). \quad (5)$$

where

$$S'_{cj}(x_c, x_j) = w_{cj} S_{cj}(x_c, x_j). \quad (6)$$

and $w_{cj} \in \{0, 1\}$. The weight is set to $w_{cj} = 0$ if the neighboring node is uncommitted $x_j = 0$ and $w_{cj} = 1$ otherwise. In this way, the uncommitted nodes do not have influence on the neighbors. For the uncommitted nodes, instead of the current value in (4), $x_c^{**}$ is used representing the minimum value under the constraint that it is at least at a certain distance $D$ for the minimum $|x_c^{**} - x_c^*| > D$. The confidence measure for uncommitted nodes is:

$$C_c^0 = E'_c(x_c^{**}) - E'_c(x_c^*) \quad (7)$$

This value can be used also as an indication of how pronounced is the local minimum $x_c^*$. In the original HCF [7], the parameter $D$ is 0, which means that the $E'_c(x_c^{**})$ $n$ is the second smallest local energy value.

For example from Fig. 1, the data costs presented on the left side are equal to the initial modified local energy $E'_c$. It can be observed that the minimum of $D_a(x_a)$ is more strongly pronounced than for the nodes $b$ and $c$. It is reasonable to assume that the better-pronounced local minima are more likely to be close to the optimal solution and could be trusted more than the data terms where the preference for certain values is not so strong. The effect of the described second modification proposed in the HCF algorithm is that preference is given to the values corresponding to "strongly pronounced" local minima during the initial ICM updates. The node $a$ will be updated first but there will be no influence from the less confident value of node $b$. As a result, the HCF will find the optimal solution for this example problem.

The HCF algorithm sorts all the nodes according to their confidence measure $C_c$. The highest confidence node is updated and then resorted. If the node was uncommitted, it becomes committed. Once all nodes are committed, the HCF is just a greedy ICM. It has been shown that this way of performing ICM updates gives better results for many vision problems [7, 13]. However, because of the constant resorting, the algorithm is computationally expensive.

## 4 Gentle ICM

In this section, an effective implementation of a heuristic similar to the HCF is described. We show how to give preference to the values corresponding to "strongly" pronounced local minima during the initial ICM updates without the expensive sorting operation of the HCF algorithm. The nodes are updated in a predefined fixed order, for example, scanning order as in Fig. 2. At each iteration, all nodes are reset to be uncommitted, but instead of the binary weights (6), real-valued weights are used $w_{cj}^k \in [0, 1]$. At the $k$ th ICM iteration, the smoothness terms are replaced by modified smoothness terms and the local energy function becomes:

$$E_c^k(x_c) = D_c(x_c) + \sum_{j \in \mathcal{N}(c)} S_{cj}^k(x_c, x_j). \quad (8)$$

where

$$S_{cj}^k(x_c, x_j) = w_{cj}^k S_{cj}(x_c, x_j). \quad (9)$$

The ICM does not save the previous values and updates of $x_c$ are performed in-place. As a result, depending of the update order, some $x_j$ in the equations above might be already updated and some not during the current iteration. The $w_{cj}^k$ starts with some small value and it is increased at each iteration until it becomes equal to 1 and the modified ICM becomes regular ICM.

Increasing $w_{cj}^k$ might look similar to "cooling" in the simulated annealing optimization. However, the effect is different. To better understand the influence of the weights, let $s_{max}$ be the maximum value that the smoothness term $S_{cj}(x_c, x_j)$ can have and let 0 be the minimum, usually for $x_j = x_c$. The value of the neighboring node $x_j$ influences the current decision for $x_c$ through the smoothness term. To change $x_c$, the energy decrease due to the smoothness term should be higher than the increase of the data term value $dDc = D_c(x_j) - D_c(x_c)$. Therefore, the $x_c$ will not be changed for $dDc > w_{cj}^k * s_{max}$. A strong local minimum of the data terms will usually have large $dDc$ and stay preserved. The ICM updates will be focused on the areas with the data terms where the preference for certain values is not so strong. Increasing $w_{cj}^k$ will gradually allow updating the nodes with stronger local minima also.

Different ways of increasing $w_{cj}^k$ are possible. In the implementation used in the experiments here the weights are calculated as:

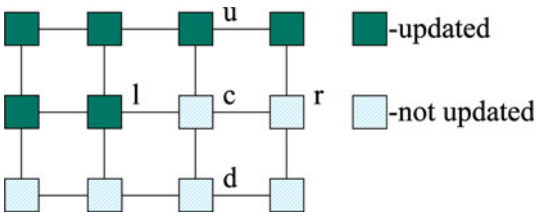



**Fig. 2** MRF on a 4-connected grid and a typical situation while updating nodes in a certain order using ICM

$$\text{if} \quad k \in [1, k_{\text{GICM}}), w_{cj}^k = W_0 \frac{k_{\text{GICM}}}{k_{\text{GICM}} - k + 1} \tag{10}$$

$$\text{if} \quad k \geq k_{\text{GICM}}, w_{cj}^k = 1, \tag{11}$$

and where $W_0 = W$ is some small initial value smaller than $1/k_{\text{GICM}}$. After $k_{\text{GICM}}$ iterations, weights become 1 and updates become regular ICM updates.

The smoothness terms are related to the prior distribution describing the relation between the neighboring nodes [5]. For the Gaussian distribution $S_{cj}(x_c, x_j) \propto \|x_c - x_j\|/\sigma^2$ the Eq. (10) corresponds to linearly decreasing the covariance from $\sigma^2/W$ to its original value $\sigma^2$.

## 5 Gentle ICM with local analysis

The pseudo code of the implementation used in all experiments is given in Fig. 3. The presented basic implementation above, denoted from now on as GICM, requires just simple modification of the regular ICM code.

The assumption from Sect. 2 was that the "strongly" pronounced local minima should get preference during the ICM updates. The strong minima are implicitly preferred by GICM as explained in the previous section and there is no need to define any confidence measure.

Using some confidence measure to give further preference to the stronger minima might be beneficial. For example, in Fig. 1, only the node $a$ has a strongly pronounced minimum. A simple heuristic is presented here to demonstrate that such type of extensions can further improve the results. The confidence measure (4) is used to further reduce the influence of the nodes with low confidence. This version of the algorithm will be denoted as GICM*, see Fig. 3.

for $k = 0, ..., K-1$ //K iterations.
  for $c = 1, ..., N$ //scan nodes
  {
    **GICM:** $W_0 = W$
    **GICM*:** $W_0 = \min(W, W_j)$, $j \in \mathcal{N}(c)$.
    if $k = 0$, $w_{cj}^k = 0$, //to set initial values
    if $k \in [1, k_{GICM})$ and $x_j$ not updated,
      $w_{cj}^k = W_0 \frac{k_{GICM}}{k_{GICM} - k + 1}$
    if $k \geq k_{GICM}$, $w_{cj}^k = 1$.

    Update: find $x_c = l$ to minimize
    $E_c(x_c)^k = D_c(x_c) + \sum_{j \in \mathcal{N}(c)} w_{cj}^k S_{cj}(x_c, x_j)$.
  }

**Fig. 3** The gentle ICM algorithm pseudo code. The standard ICM is obtained if not modified $E_c$ is used

Let us consider a typical truncated smoothness term:

$$S_{cj}(x_c, x_j) = \min(|x_c - x_j|^p, s_{\max})] \tag{12}$$

where $p = 1$ for L1 norm and $p = 2$ for L2 norm. The truncation point $|x_c - x_j|^p = s_{\max}$ could be used to define the parameter $D$ describing how broad a local minimum can be in (4). In the implementation used in the experiments, the initial weight (10) is modified as $W_0 = \min(W, Wj)$, see Fig. 3, where $W_j = C_c^0/s_{\max}$.

## 6 Experiments

The presented modifications of the ICM algorithm, GICM and the extension GICM* are tested. For the weights $W = 1/8$ is used and $k_{\text{GICM}} = 4$ such that after four iterations, the regular ICM updates are performed. The same algorithms are used for the comparison as in [18]. For the graph-cuts, the standard implementations of the "swap-move" (SWAP) and the "expansion-move" (EXP) [6] are used. For the LBP two different implementations are used, BP-S and BP-M from [18]. They differ in the order in which the nodes are updated and in the way the final solution is computed. Tree-reweighted message-passing (TRW-S) [10] is included as probably the most efficient message-passing-based algorithm. In addition, we include the original HCF algorithm [7] and the recent efficient graph-cut algorithm named FastPD [11]. The FastPD is based on the expansion graph-cut. It is using a standard first iteration but the method is reusing previous computations to speed up the new iterations by solving smaller parts of the whole problem. A similar approach was also presented in [1]. Similar principles as in FastPD are used, but additional improvements in computation time are achieved by analyzing and simplifying the initial problem. Unfortunately, this is done only for the Potts smoothness costs. The authors also report performance close to FastPD. Therefore, this method is not included in the evaluation.

Example results are in Figs. 5, 6, 7, 8. For comparison with the modern optimization algorithms, the result of the algorithm that achieved the lowest energy is also included. The final achieved energy levels are in Table 1. The same set of vision energy minimization problems with the same parameters as in [18] is used. The vision problems and the parameters defined in [18] are briefly described here for completeness. The achieved minimum energy as percentage of the energy lower bound that can be computed as described in [10]. The lower bound values for the used problems can be found at "http://vision.middlebury.edu/MRF/".

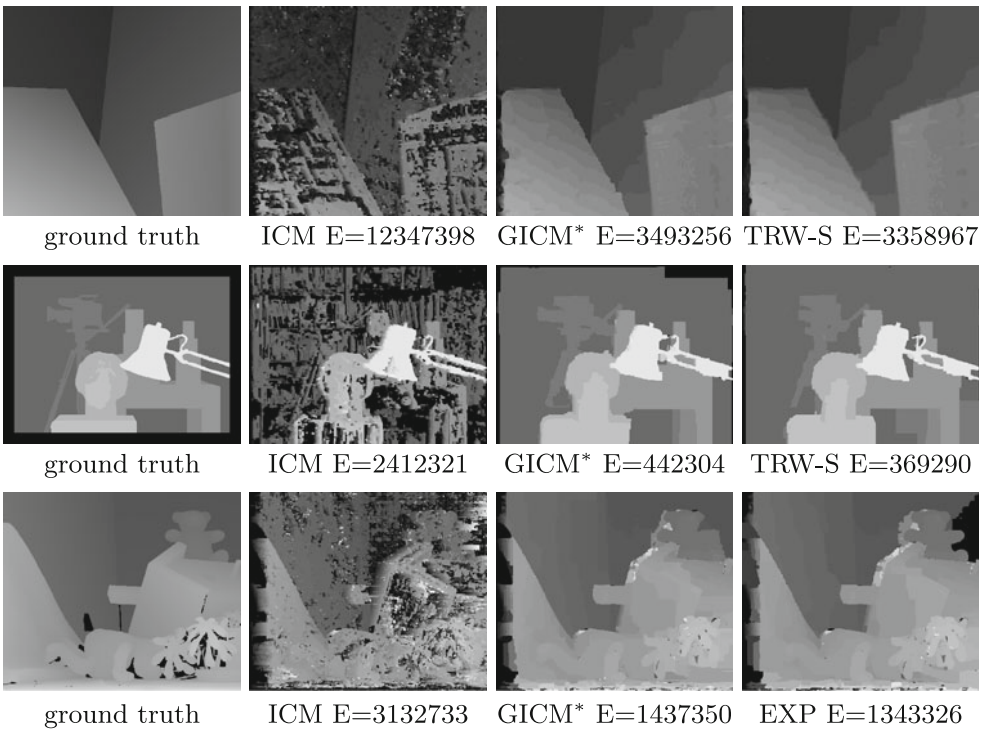



**Fig. 4** Stereo matching examples with attained energy values: "Venus" (*top*), "Tsukuba" (*middle*) and "Teddy" (*bottom*)

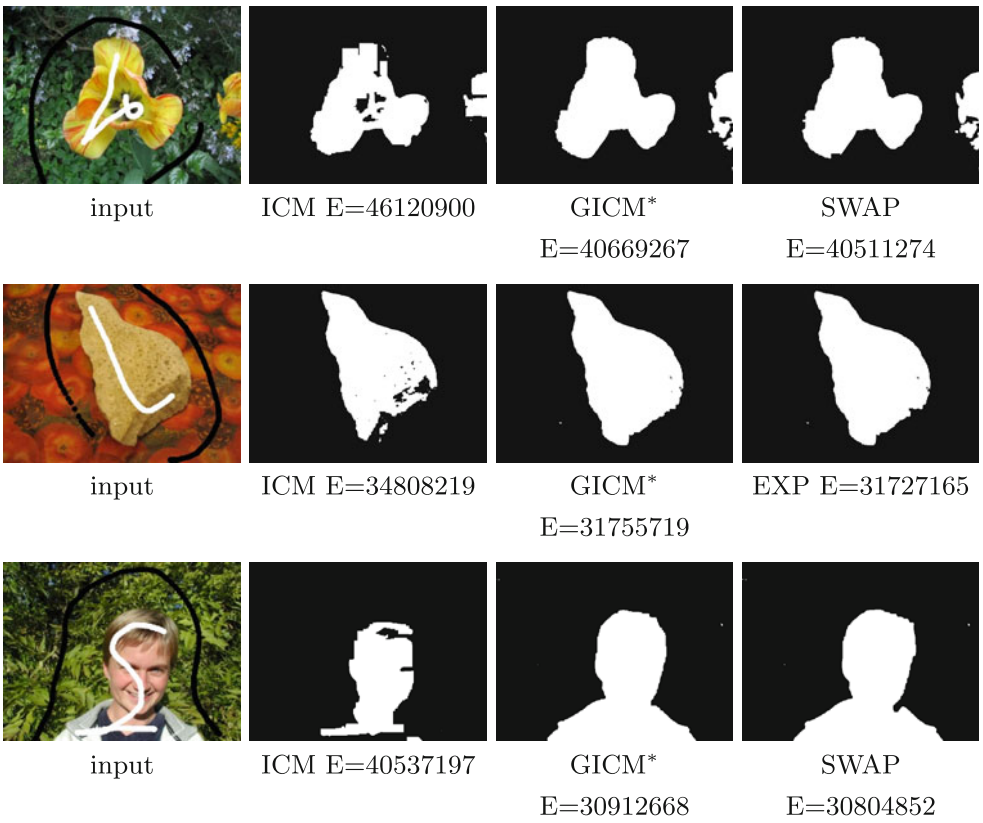



**Fig. 5** Image-segmentation examples with attained energy values: "Flower" (*top*), "Sponge" (*middle*) and "Person" (*bottom*)

### 6.1 Stereo matching

The data cost is absolute color difference, taking care of the image-sampling effects as suggested by Birchfield and Tomasi [4]. For the "Venus" images, 20 disparity levels are used and $S_{cj}(x_c, x_j) = 50 \min(|x_c - x_j|^2, 7)$. For the "Tsukuba" and "Teddy" intensity gradient is calculated $\nabla_{cj}$ and used to locally adapt the smoothness costs. For the

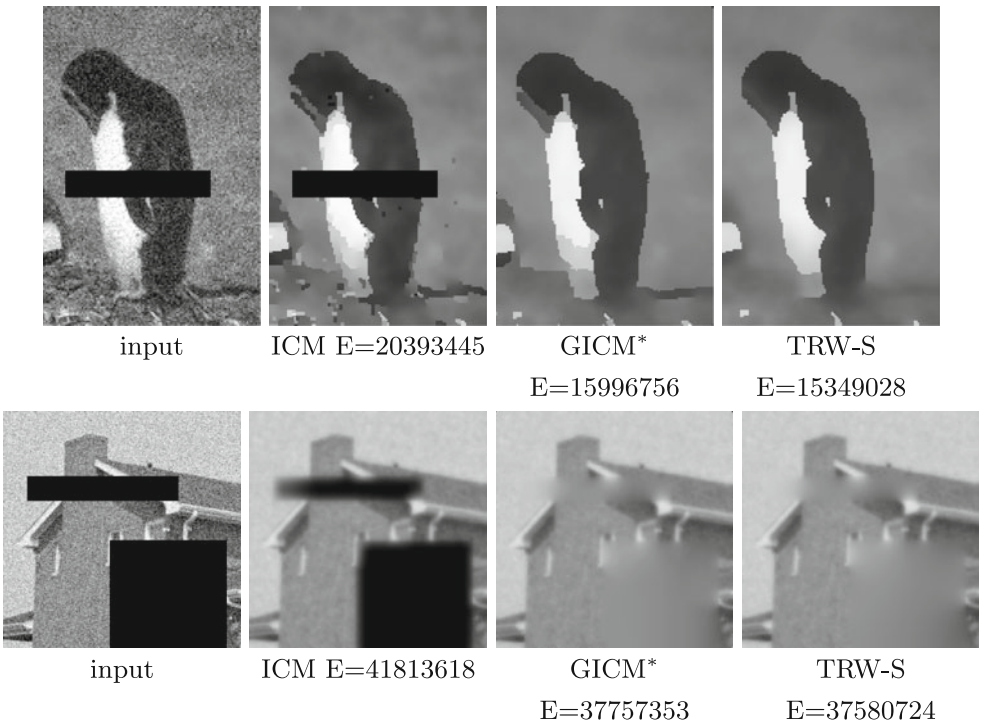



**Fig. 6** Image-denoising and inpainting examples with attained energy values: "Penguin" (*top*) and "House" (*bottom*)


**Fig. 7** Image-stitching examples (photomontage) with attained energy values : "Panorama" (*top*) and "Family" (*bottom*). The differences can be spotted at sides of the "Panorama" example images and at the faces in the *middle* of the image for the "Family" example

"Tsukuba" images, 16 disparity levels are used and $S_{cj}(x_c, x_j) = 20 \quad g_{cj}\min(|x_c - x_j|, 2)$, where $g_{cj} = 2$ if $|\nabla_{cj}| \leq 8$ and $g_{cj} = 1$ otherwise. For "Teddy", 60 levels and $S_{cj}(x_c, x_j) = 20 \; g_{cj}\min(|x_c - x_j|, 2)$, where $g_{cj} = 3$ if $|\nabla_{cj}| \leq 10$ and $g_{cj} = 1$ otherwise. The GICM result and the lowest energy result from [18] Fig. 4 (right) look both very similar to the ground truth depth map Fig. 4 (left).

### 6.2 Binary image segmentation

The variables $x_c$ have values 0 indicating background pixels and 1 indicating foreground. The foreground and background are modeled as two Gaussian mixture models, as in [16]. The data costs are the log-likelihoods of the two models. The user drawing for the initial selection of the

background and the foreground is shown over the original images in Fig. 5. The smoothness is a Potts model that is contrast sensitive: 0 if $x_c = x_j$, otherwise $50(\exp(-\|I(c) - I(j)\|^2/2\sigma^2) + 10)$. Here $I(c)$ and $I(j)$ are the corresponding RGB colors and $\sigma^2$ is the variance of $\|I(c) - I(j)\|$ over the whole image. The differences compared to the optimal graph-cut results can be hardly noticed, see Fig. 5.

### 6.3 Image denoising and inpainting

The variables $x_c$ are the 8-bit pixel intensities. The data cost is the squared difference between the observed and estimated pixel intensity, and zero in the obscured parts. For "Penguin" $S_{cj}(x_c, x_j) = 25 \quad \min(|x_c - x_j|^2, 200)$ and for "House" $S_{cj}(x_c,x_j) = 5\,|x_c - x_j|^2$. The results of the gentle ICM look very similar to the best results from [18], see Fig. 6.

### 6.4 Photomontage

Input is a set of aligned images $I_1, \ldots, I_M$. A new combined stitched image is constructed where $x_c$ indicates which input image is used for the current pixel of the resulting image. In the "Panorama" example, a larger image is reconstructed from a set of partially overlapping images. The data cost is set to $\infty$ if the pixel of the larger image does not exist in the smaller image, otherwise the cost is zero. In the "Family" example, images are of the same size but there are user supplied drawn strokes to indicate which pixels from which image should be included. The data cost is $\infty$ if a pixel is chosen from a different image than the user indicated. Otherwise, the data cost is zero. The smoothness is $S_{cj}(x_c,x_j) = 0$ if the neighboring pixels are taken from the same image $x_c = x_j$. If different images are chosen, $S_{cj}(x_c, x_j) = |I_{x\_c}(c) - I_{x\_j}(c)| + |I_{x\_c}(j) - I_{x\_j}(j)|$. For the "Family" example, the smoothness costs are divided by the sum of absolute gradients in both images $|\nabla_{cj} I_{x\_c}| + |\nabla_{cj} I_{x\_j}|$. Example results are in Fig. 7.

### 6.5 Energy-minimization performance

The achieved energy values in Table 1 demonstrate that the "gentle" ICM often provides results similar to the modern algorithms [18]. The trivial modification GICM gives already a huge boost in performance when compared to the standard ICM implementation [5, 18]. The additional reliability checks GICM* lead to further improvement. The "shape" of the local energy is analyzed in a simple way. More elaborate analysis tuned to a particular problem is expected to give even better results, but this is beyond the scope of this paper.

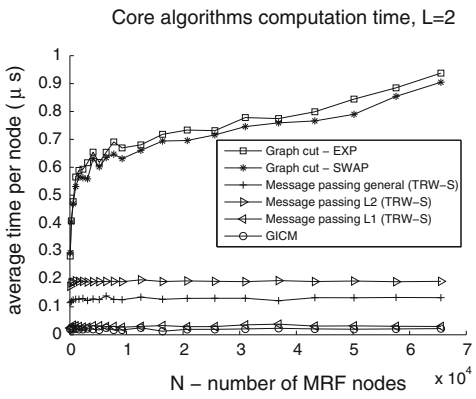
**Fig. 8** Computation time experiments for various numbers of labels $L$. The measured time and energy at each iteration are shown. *Left*: average computation time for various number of nodes of the core algorithms. The averages are computed from 100 iterations and 10 different MRF problems generated by randomly generating the data costs for each selected number of nodes $N$ and labels $L$. For $L = 60$, the presented results for graph-cuts are for $L = 64$, because the algorithm implementation allows only power 2 numbers. *Right*: computation time and energy with iteration for real problems with corresponding number of labels $L$ ▶

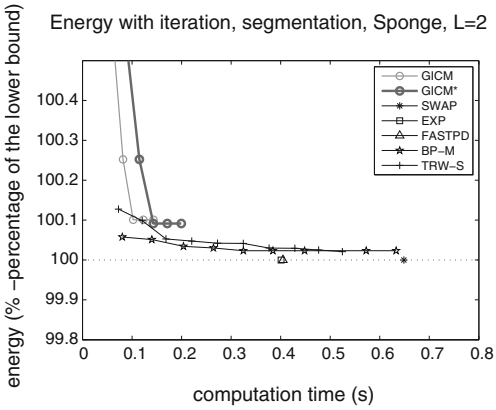
### 6.6 Computation time

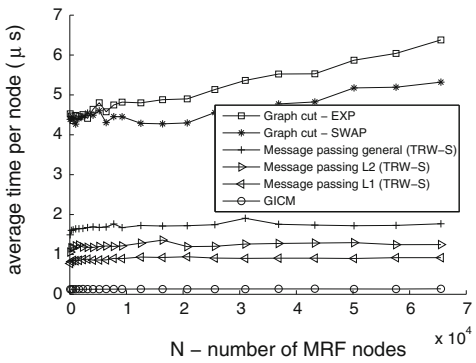
A set of experiments was performed to further analyze the computation time performance of the GICM and compare it to the modern graph-cut and message-passing methods. The computation costs depend on the number of MRF nodes $N$ (i.e., image size), the number of discrete labels $L$ (e.g., $L = 2$ for the segmentation problems and $L = 256$ for the denoising problem) and the number of iterations $K$. The number of iterations for each of the tested algorithms depends heavily on the problem and application. On the other hand the computation per iteration is determined by the optimization method structure. Therefore, these two aspects are analyzed separately to clearly illustrate both of them.

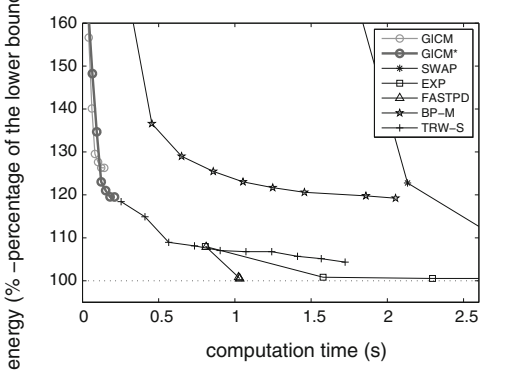
A set of problems is chosen with different numbers of labels $L$ starting from the simplest binary segmentation problem $L = 2$ and ending with $L = 256$. For each selected number of labels, the computation time per iteration is analyzed on synthetic problems to evaluate the theoretical performance and how the computation time will scale with the number of nodes $N$. Only the core message-passing and graph-cut algorithms common for all methods are considered. A corresponding real problem is then used to illustrate the computation and energy-reduction with each iteration. For the real problems, we include larger set of various practical implementations such as BP and FastPD [11]. The FastPD uses standard expansion graph-cut initial iteration, but reusing the results of the previous graph-cut makes subsequent graph-cut simpler. A similar approach is presented earlier by [1], but we used here just the FastPD as a more general approach. All the algorithms and the proposed modifications are implemented within the code from the Middlebury dataset [18] that is then used for evaluation. Gcc compiler and -$O_3$ optimization was applied and the time was measured on an Intel Core i5-2520M at 2.5 GHz laptop with 4 GB of memory.

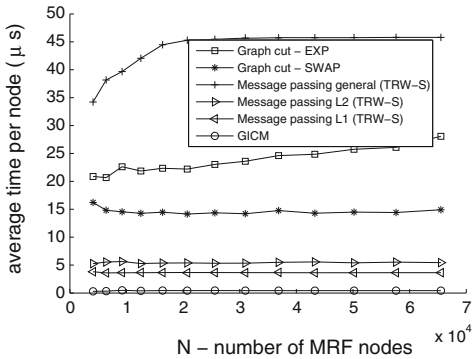
Synthetic MRF are constructed to analyze the performance of the algorithms per iteration. GICM, performs a simple one dimensional minimum search for $L$ labels for each node and as result runs in $O(N\,L\,K)$ time [5]. HCF [7, 13] is the same, but requires additional resorting of the

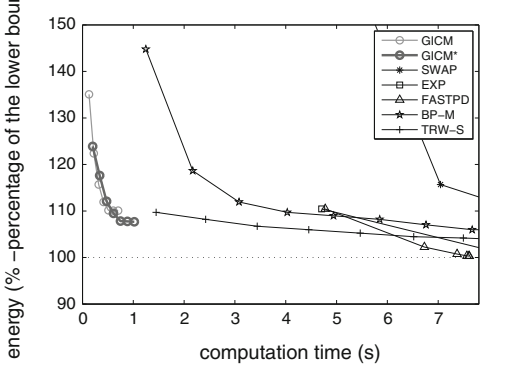

Core algorithms computation time, L=2
average time per node ( μ s)
N – number of MRF nodes
x 10^4
Graph cut – EXP
Graph cut – SWAP
Message passing general (TRW–S)
Message passing L2 (TRW–S)
Message passing L1 (TRW–S)
GICM

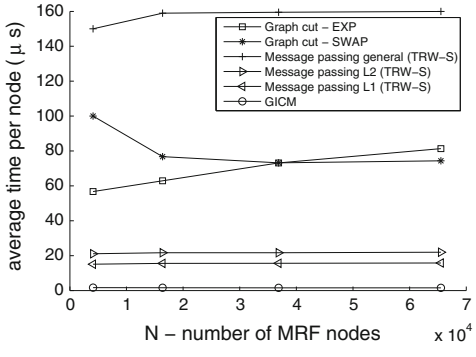

Energy with iteration, segmentation, Sponge, L=2
energy (% –percentage of the lower bound)
computation time (s)
GICM
GICM*
SWAP
EXP
FASTPD
BP–M
TRW–S


Core algorithms computation time, L=16
average time per node ( μ s)
N – number of MRF nodes
x 10^4
Graph cut – EXP
Graph cut – SWAP
Message passing general (TRW–S)
Message passing L2 (TRW–S)
Message passing L1 (TRW–S)
GICM


Energy with iteration, stereo matching, Tsukuba, L=16
energy (% –percentage of the lower bound)
computation time (s)
GICM
GICM*
SWAP
EXP
FASTPD
BP–M
TRW–S


Core algorithms computation time, L=60
average time per node ( μ s)
N – number of MRF nodes
x 10^4
Graph cut – EXP
Graph cut – SWAP
Message passing general (TRW–S)
Message passing L2 (TRW–S)
Message passing L1 (TRW–S)
GICM


Energy with iteration, stereo matching, Teddy, L=60
energy (% –percentage of the lower bound)
computation time (s)
GICM
GICM*
SWAP
EXP
FASTPD
BP–M
TRW–S


Core algorithms computation time, L=256
average time per node ( μ s)
N – number of MRF nodes
x 10^4
Graph cut – EXP
Graph cut – SWAP
Message passing general (TRW–S)
Message passing L2 (TRW–S)
Message passing L1 (TRW–S)
GICM

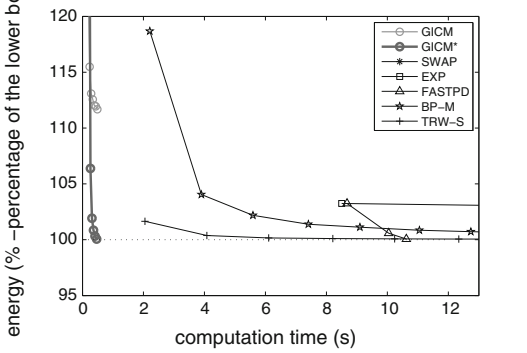

Energy with iteration, denoise and inpaint, House, L=256
energy (% –percentage of the lower bound)
computation time (s)
GICM
GICM*
SWAP
EXP
FASTPD
BP–M
TRW–S

**Table 1** Overview of the achieved minimum energy as percentage of the energy lower bound [10]

| Method: | ICM (%) | HCF (%) | GICM (%) | GICM (%)* | BP-S (%) | BP-M (%) | TRW-S (%) | SWAP (%) | EXP (%) | FASTPD (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| Stereo matching | | | | | | | | | | |
| Tsukuba | 653.36 | 328.7 | 128.73 | 119.8 | 115.1 | 110.1 | **100.02** | 100.51 | 100.32 | 100.3 |
| Venus | 405.11 | 215.4 | 115.51 | 104.06 | 110.02 | 101.67 | **100.02** | 128.70 | 102.83 | 101.2 |
| Teddy | 234.29 | 160.3 | 110.1 | 107.25 | 106.24 | 103.49 | 100.70 | 100.81 | **100.47** | 100.47 |
| Photomontage | | | | | | | | | | |
| Pano | 522.57 | 462.05 | 400.52 | 226.46 | 556.29 | 159.60 | 121.61 | 145.42 | **101.10** | 101.1 |
| Family | 1604.45 | 1235.81 | 792.40 | 343.81 | 1063.82 | 189.36 | **100.06** | 172.84 | 100.48 | 100.4 |
| Binary image segmentation | | | | | | | | | | |
| Flower | 113.85 | 107.3 | 100.79 | 100.39 | 100.07 | 100.07 | **100.00** | **100.00** | **100.00** | **100.00** |
| Sponge | 109.71 | 102.1 | 100.09 | 100.09 | 100.03 | 100.02 | **100.00** | **100.00** | **100.00** | **100.00** |
| Person | 131.59 | 120.2 | 100.52 | 100.35 | 100.03 | 100.03 | **100.00** | **100.00** | **100.00** | **100.00** |
| Image denoising and inpainting | | | | | | | | | | |
| Penguin | 132.90 | 123.7 | 109.15 | 104.22 | 101.30 | 101.30 | **100.03** | 111.29 | 103.74 | 103.8 |
| House | 111.26 | 104.2 | 109.60 | 100.47 | 100.26 | 100.26 | **100.00** | 100.89 | 103.42 | 100.9 |

The best results are indicated in bold. The proposed GICM and the extended version with local weighting GICM* are compared to a set of algorithms

nodes which becomes prohibitive for large images, i.e., large $N$. Due to the poor performance ICM and HCF are excluded from the further analysis. The message-passing algorithms need to combine messages for all labels giving $O(NL^2K)$ computation time in general case [20]. For specific smoothness terms (such as truncated $L1$ and $L2$ norm) it is possible to perform the message-passing in $O(N\,L\,K)$ time as shown by [14]. Since the various BP implementations and TRW-S are internally the same, we only used the TRW-S here to evaluate the general message-passing computation costs. For the graph-cut methods, there are two core approaches: swap and expansion. Both have an internal graph-cut routine. In practice, all graph-cut methods are superlinear in computation time with respect to the number of nodes $N$ [6]. It should be also noted that the swap method needs to perform at least $L^2$ graph-cuts and the expansion needs $L$ to complete one update cycle of all labels. In the evaluation presented here, an iteration refers to just one graph-cut.

The results for various numbers of labels are presented in Fig. 8; on the left side, the performance per node and per iteration of the core algorithms is shown. For the binary problem $L = 2$, we see clearly the superlinear nature of the graph-cut algorithms which can make them expensive for large images. For other cases, with larger numbers of labels $L$, this increase in costs is somewhat masked at start by the large additional overhead of constructing and maintaining all the graphs. It should be noted once more that the swap graph-cut needs to perform $L^2$ graph-cuts to update all labels and is not suitable for large numbers of labels. It can be also observed that the specific implementation for the message-passing with $L2$ costs [14] does not pay off for small numbers of labels because of the additional overhead. For larger $L$, the computation costs of the general message-passing explode and the message-passing for general costs becomes the most complex method, see $L = 256$ in Fig. 8 left. In all cases, the computation time needed for the simple operations in GICM remains much lower than any other method. Only GICM and not the GICM* is presented in these graphs, since the computation time is very similar and it is hard to visually distinguish them on these plots.

On the right side of Fig. 8, the real examples are shown that illustrate also the effectiveness of the iterations. While the graph-cut iterations are quite computationally costly, the graph-cut-based methods typically need a few iterations to reach the final value. For the binary problems $L = 2$, they find the optimum in one iteration. The number of needed iterations grows with the numbers of labels. The FastPD makes similar steps as the standard graph-cut, but subsequent iterations require less computation. From the examples presented here and in [1, 11], the computation time reduction is around 50–80 % for each new iteration and this depends on the problem. The message-passing algorithms often make smaller steps and need many iterations to reach the final value. The examples illustrate also the typically superior performance of the TRW-S over BP. The GICM, as ICM, is a greedy method and for vision problems finds a local maximum in just a few iterations. The achieved energy is not as low as for the other methods, but it is computed much faster. The difference in computation time becomes more visible for problems with large numbers of labels. In some examples, TRW-S comes close in computation time to the GICM. However, this is for the $L1$ and $L2$ type costs. The graphs on the left side of Fig. 8 show that TRW-S is very efficient for $L1$ costs, less efficient for $L2$ but slow for a general type of costs.

### 6.7 Discussion

The presented modifications influence the ICM in the initial iterations and help in avoiding the strong influence of the initial values. After a few iterations, the weights that influence the smoothness terms become equal to 1 and the algorithm becomes regular ICM which will decrease the energy in each step until it reaches a local minimum. The "highest confidence first" [7] is a greedy heuristic that relies more on the "strongly" pronounced local minima. The only parameter of the presented base algorithm is the schedule of reducing the weights that influence the smoothness terms which in a way controls how "greedy" the steps are performed. The described fixed schedule is used in all experiments. However, we noticed that the results depend on the schedule and sometimes poor results can be achieved if the weights are increased very slow or very fast. The way the nodes/pixels are processed can also have influence, since the "strongly" pronounced local minima get propagated in different ways. In all experiments, we used the scanning update order, but it was noticed that different update orders lead to different results. The greedy nature of the GICM algorithm can be observed in the presented examples as tendency of the algorithm to propagate some values over weak object borders, see for example, the neck of the boy in Fig. 5 or the teddy bear in Fig. 4. The information at the stronger borders is good enough to prevent the greedy propagation of the labels. This leads to still reasonably good solutions for these examples. The "photomontage" examples also illustrate the greedy nature of the heuristic and the difference in achieved energy with respect to the more complex methods is the largest. The areas with single label in the final solution are large, so the greedy algorithm fails to find the optimal compromise.

The experiments show that including further heuristics that analyze the shape of the local energy can improve results. However, this leads to many options and new parameters that can be introduced and it was not further analyzed here. It is interesting to mention that the greedy approach for alpha matting in [9] is increasing the MRF smoothness costs around the user-selected pixels to speed up the propagation of the user-driven segmentation. This is another way how the approach presented here could be extended, but this is beyond this paper.

## 7 Conclusions

This paper starts with the assumption that ICM performance would be better if the "strongly" pronounced local minima get preference during the ICM updates. While there are many different ways to implement the proposed heuristics [7], the experimental section shows that the implementation proposed here can drastically improve the ICM performance for a number of vision problems. The method retains the properties of the standard ICM after a few iterations. The modern optimization algorithms still achieve lower energy solutions than the proposed solution. However, the presented results indicate that often reasonable solutions can be obtained in short time and with limited memory requirements using the modified ICM presented here. The computational advantages are the most significant for the problems with a large number of labels.

## References


1. Alahari, K., Kohli, P., Torr, P.: Reduce, reuse & recycle: Efficiently solving multi-label MRFs. In: CVPR (2008)
2. Barnard, S.: Stochastic stereo matching over scale. Int. J. Comput. Vis. **3**(1), 17–32 (1989)
3. Besag, J.: On the statistical analysis of dirty pictures. J. R. Stat. Soc. Ser. B **48**(3), 259–302 (1986)
4. Birchfield, S., Tomasi, C.: A pixel dissimilarity measure that is insensitive to image sampling. IEEE Trans Pattern Anal Mach Intell **20**(4):401–406 (1998)
5. Bishop, C.: Pattern recognition and machine learning. Springer, Berlin (2006)
6. Boykov, Y., Veksler, O., Zabih, R.: (2001) Fast approximate energy minimization via graph cuts. IEEE Trans Pattern Anal Mach Intell **23**(11):1222–1239
7. Chou, P., Brown, C.: The theory and practice of Bayesian image labeling. Int J Comput Vision **4**:185–210 (1990)
8. Fletcher, R.: (2000) Practical methods of optimization. Wiley, New York
9. Guan, Y., Chen, W., Liang, X., Ding, Z., Peng, Q.: Easy matting: a stroke based approach for continuous image matting. J Comput Graphics Forum **25**(3):567–576 (2006)
10. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. IEEE Trans Pattern Anal Mach Intell **28**(10):1568–1583 (2006)
11. Komodakis, N., Tziritas, G., Paragios, N.: Performance vs computational efficiency for optimizing single and dynamic MRFs: setting the state of the art with primal dual strategies. Comput Vision Image Underst **112**(1):14–29 (2008)
12. Komodakis, N., Tziritas, G.: Image completion using efficient Belief Propagation via priority scheduling and dynamic pruning. IEEE Trans Image Process **16**(11):2649–2661 (2007)
13. Li, S.: Markov Random Field Modeling in Computer Vision. Springer, Berlin (1995)
14. Felzenszwalb, P.F., Huttenlocher, D.: Efficient belief propagation for early vision. Int J Comput Vision **70**(1):41–54 (2006)
15. Ogawara, K.: Approximate belief propagation by hierarchical averaging of outgoing messages. In: ICPR (2010)
16. Rother C., Kolmogorov V., Blake A.: "'GrabCut"'-interactive foreground extraction using iterated graph cuts. ACM Trans Graphics **23**(3):309–314 (2004)
17. Rother, C., Kolmogorov, V., Lempitsky, V., Szummer, M.: Optimizing binary MRFs via extended roof duality. In: CVPR (2007)

18. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother C.: A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. IEEE Trans Pattern Anal Mach Intell **30**(6):1068–1080 (2008)
19. Yang, Q., Wang, L., Ahuja, N.: A Constant-Space Belief Propagation Algorithm for Stereo Matching. In: CVPR (2010)
20. Yedidia, J., Freeman, W., Weiss, Y.: Generalized Belief Propagation. In: NIPS (2000)