Multi-label image segmentation via max-sum solver*

Banislav Mičušík
Pattern Recognition and Image Processing Group,
Inst. of Computer Aided Automation, Vienna
University of Technology, Austria
micusik@prip.tuwien.ac.at

Tomáš Pajdla
Center for Machine Perception,
Dpt. of Cybernetics, Czech Technical
University, Czech Republic
pajdla@cmp.felk.cvut.cz

Abstract

We formulate single-image multi-label segmentation into regions coherent in texture and color as a MAX-SUM *problem for which efficient linear programming based solvers have recently appeared. By handling more than two labels, we go beyond widespread binary segmentation methods,* e.g., MIN-CUT *or normalized cut based approaches. We show that the* MAX-SUM *solver is a very powerful tool for obtaining the MAP estimate of a Markov random field (MRF). We build the MRF on superpixels to speed up the segmentation while preserving color and texture. We propose new quality functions for setting the MRF, exploiting priors from small representative image seeds, provided either manually or automatically. We show that the proposed automatic segmentation method outperforms previous techniques in terms of the Global Consistency Error evaluated on the Berkeley segmentation database.*

1. Introduction

The image segmentation procedure assigns a label from a given discrete set to each image pixel following some criteria. The segmentation process can be formulated as an optimization task that optimizes the criteria function consisting of an image fitness and a prior on likely segmentations.

Depending on the prior, the segmentation can follow semantic, shape, texture, color, *etc*., cues. To obtain a semantically correct segmentation, the priors have to be learnt in advance, *e.g*. from a manually labeled training database, to capture the mutual relation of object pixels and appearance [3, 19, 18, 9]. Contrary to that, a texture or/and color coherent segmentation can be achieved by learning the priors directly from an input sequence [21, 8] or just from a single image either with user interaction [14, 2, 1] or without [17, 5, 12].

Figure 1. Automatic single-image multi-label (in this case 8 labels) segmentation respecting texture and color cues using the proposed method.

In this paper we focus on automatic single-image multi-label segmentation into regions coherent in texture and color, see Fig. 1. We rely on the priors learnt directly from the image being segmented. The method based on MAX-SUM formulation groups pixels having similar texture and color properties into non-overlapping segments. Such segmentation is very useful, *e.g*., it allows possibly further classification and merging the segments respecting some semantic cues, or in stereo matching algorithms it can help to restrict the search area of possible point matches.

By handling more than two labels we go beyond widespread binary segmentation methods, *e.g*., MIN-CUT [1, 2, 18, 8, 12] or normalized cut [17, 5] based frameworks. The MIN-CUT has become a standard algorithm in many computer vision problems because of its speed and accuracy. However, the MIN-CUT still solves only binary segmentation and a multi-label segmentation has to be done indirectly. The MAX-SUM solver [20] permits more labels to be defined allowing the multi-label segmentation to be done directly in a global manner while still keeping the processing time within reasonable bounds. There

*Research of B. Mičušík has been supported by FWF-P17189-N04 SESAME and FP6-IST-507752 MUSCLE and research of T. Pajdla by FP6-IST-027787 DIRAC and MSM6840770038 DMCM III grants.

1-4244-1180-7/07/$25.00 ©2007 IEEE

are other attempts to solve the labeling problem on the MRF using, *e.g.*, second order cone programming [9], sequential tree-reweighted max-product message passing [7] or belief propagation methods [22]. However, neither of the algorithms, nor the MAX-SUM, solve the problem of the multi-label MAP of the MRF exactly as it is NP-hard. Various approximations are taken into account to reach a good suboptimal solution. Moreover, the belief propagation algorithms sometimes do not even converge for cyclic graphs [22] which we are interested in.

The main contribution of the paper is solving a single-image multi-label segmentation problem by the recently introduced MAX-SUM solver [20] while taking into account superpixels [4] and proposed priors composed of texture and color cues. We also propose an automatic positioning of seeds from which the priors are learnt allowing images to be segmented fully automatically. Alternatively the equivalent MAX-PROD based solver [7] can be used.

The MAX-SUM problem and symbols used are defined in Sec. 2. Then, structure and setting of weights of the MRF are stated in Sec. 3 using priors from shifted seeds explained in Sec. 4. Experimental evaluation of the method and the conclusion are given in Sec. 5 and 6, respectively.

2. The MAX-SUM problem

The MAX-SUM (labeling) problem of the second order is defined as maximizing a sum of bivariate functions of discrete variables. The solution of a MAX-SUM problem corresponds to finding a configuration of a Gibbs distribution with maximal probability. It is equivalent to finding a maximum posterior (MAP) configuration of an MRF with discrete variables [20].

We assume an MRF, *i.e.* graph $\mathcal{G} = \langle \mathcal{T}, \mathcal{E} \rangle$, consisting of a discrete set $\mathcal{T}$ of objects (in the literature also called sites, locations) and a set $\mathcal{E} \subseteq \binom{|\mathcal{T}|}{2}$ of pairs of those objects. Each object $t \in \mathcal{T}$ is assigned a label $x_t \in \mathcal{X}$ where $\mathcal{X}$ is a discrete set. A *labeling* is a mapping that assigns a single label to each object, represented by a $|\mathcal{T}|$-tuple $\mathbf{x} \in \mathcal{X}^{|\mathcal{T}|}$ with components x_t.

An instance of the MAX-SUM problem is denoted by the triplet $(\mathcal{G}, \mathcal{X}, \mathbf{g})$, where the elements $g_t(x_t)$ and $g_{tt'}(x_t, x_{t'})$ of $\mathbf{g}$ are called *qualities*. The quality of a labeling $\mathbf{x}$ is defined as

$$F(\mathbf{x} \,|\, \mathbf{g}) = \sum_t g_t(x_t) + \sum_{\{t,t'\}} g_{tt'}(x_t, x_{t'}). \qquad (1)$$

Solving the MAX-SUM problem means finding the set of optimal labellings

$$\mathcal{L}_{\mathcal{G},\mathcal{X}}(\mathbf{g}) = \operatorname*{argmax}_{\mathbf{x} \in \mathcal{X}^{|\mathcal{T}|}} F(\mathbf{x} \,|\, \mathbf{g}). \qquad (2)$$

Fig. 2 depicts the symbols and the problem in a more intuitive way on a simple grid graph. Recently, a very efficient algorithms for solving this problem through linear programming relaxation and its Lagrangian dual, originally proposed by Schlesinger in 1976 [15], has been reviewed [16, 20].

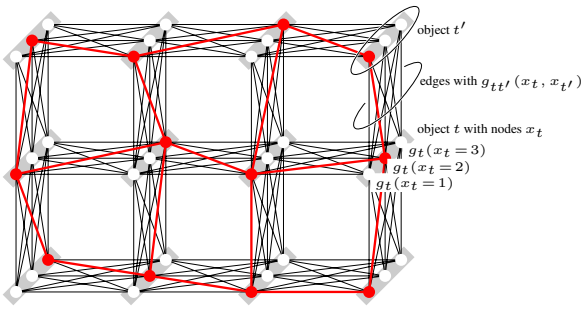


Figure 2. A 3×4 grid graph $\mathcal{G}$ for $|\mathcal{X}| = 3$ labels with symbols explained in the text. A labeling $\mathcal{L}$ from Eq. (2) is shown by a red thick subgraph. Image provided by courtesy of T. Werner [20].

3. Graph construction

Generally, the most difficult problem and art connected to segmentation methods is to encode all possible priors about objects being segmented, *e.g.*, texture, color, shape, appearance, *etc.*, into a graph, resp. an MRF, while keeping the problem tractable. In this paper, we focus on defining such priors which lead to partitioning an image into texture and color coherent regions as Fig. 1 shows.

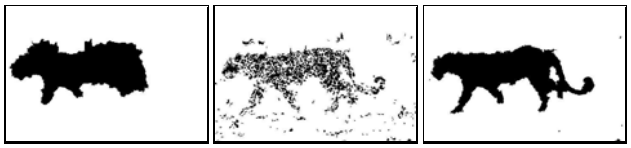
3.1. Graph entities

We build a graph on a pre-segmented image, *i.e.* on superpixels, assuming that the area of superpixels is small to leave more flexibility for the MAX-SUM solver. The aim is to merge locally together pixels having similar color ignoring at this stage any texture cues. The use of superpixels significantly reduces the number of nodes in the graph while still preserving texture information. Simply reducing the image size used by many approaches to avoid large complexity leads to losing details and high texture frequencies.

In this paper, we use a very fast method [4] giving us, by appropriate setting of parameters, regions 10 pxl large on average. However, any other segmentation method giving small segments consistent in color can be used.

The graph entities are the following. The superpixels represent objects, *i.e.* the set $\mathcal{T}$, in the graph and edges, *i.e.* the set $\mathcal{E}$, are established between each two neighboring superpixels respecting the 8-point neighborhood. The number of sites (labels) K is a constant given as a parameter.

The graph entities, *i.e.* each edge $g_{tt'}(x_t, x_{t'})$ and each object node $g_t(x_t)$ is set according to the smoothness and data term respectively, described in the following sections. After building and setting the graph, the MAX-SUM solver [6] is run to obtain a particular label x_t for each su-



(a) (b) (c)

Figure 3. Effect of various types of smoothness terms on the final segmentation of the leopard image in Fig. 1 demonstrated on a segment corresponding to a leopard itself. (a) Ising prior, (b) color gradient, (c) the proposed term.

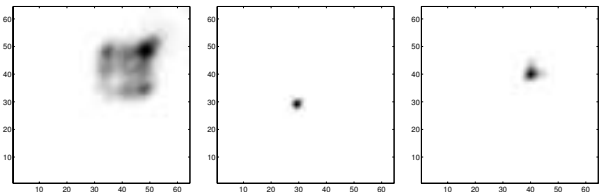


Figure 4. Neighborhood histograms for L, a, b channels computed from superpixels behind the seeds corresponding to the leopard texture from Fig. 6. Both axes correspond to histogram bins.

perpixel t.

3.2. Smoothness term

The smoothness term, defined by $g_{tt'}(x_t, x_{t'})$, controls the mutual bond of neighboring superpixels. A basic setting of this term is via the Ising prior penalizing neighbors not having same labels, *i.e*.

$$g_{tt'}(x_t, x_{t'}) \begin{cases} 0 & \text{if } x_t = x_{t'}, \\ -1 & \text{if } x_t \neq x_{t'}. \end{cases} \tag{3}$$

It tends to keep segments smooth but does not take into account any information on pixel color or intensity, see Fig. 3 (a).

Another, and more popular, measure is based on the color gradient computed between neighboring pixels,

$$g_{tt'}(x_t, x_{t'}) = \exp\left(\alpha \|\mathbf{u}_t - \mathbf{u}_{t'}\|^2\right) - 1, \tag{4}$$

where $\mathbf{u}_t$ is a 3-element color vector of the t-th superpixel and $\alpha < 0$ is a tuning parameter. However, this measure does not take into account texture information and can be even worse than the Ising prior. For instance, assume a texture consisting of black spots on light background, *e.g*. a texture of leopard skin in Fig. 1. Even though the pixels on the black spot/background boundary belong to the same texture, the smoothness term penalizes them due to high contrast and tends to separate them. Fig. 3 (b) shows this effect, notice the spareness of the segmented texture.

Another smoothness term was also proposed in [10, 12] to be defined on a regular pixel grid using a larger neighborhood and a combined color and texture gradient giving promising results on textures. However, computing the combined color and texture gradient is very expensive and, moreover, it cannot be easily used for superpixels not arranged in a regular grid.

We propose a new smoothness term, a modified Ising prior based on *color neighborhood histograms*, outperforming the previous terms, see Fig. 3 (c). As a base we use the prior learnt from seeds, *i.e*. small circular image patches, given either by user input or by a proposed automatic method described later. Each class (label) is defined by a few of these seeds, see Fig. 6. The seeds give us strong priors on neighboring colors. We construct a 2-dimensional color neighborhood histogram for each class and for each color channel. There are $3 \times K$ neighborhood histograms, *i.e*. h_x^i where $x = 1...K$ for $K = |\mathcal{X}|$ number of labels and $i = 1...3$ for 3 color channels, computed from the superpixels of the seeds. The dimension of each h_x^i is $n \times n$, where n is the number of bins, 64 in our case. An example of a neighborhood histogram for one label is shown in Fig. 4. Off-diagonal blobs say that there are some neighboring superpixels with different colors in the texture. To build the histograms we use soft binning using a Gaussian kernel and normalize them to sum to 1.

Using the neighborhood histograms, we set the smoothness term as follows,

$$g_{tt'}(x_t, x_{t'}) \begin{cases} \left(\prod_{i=1}^{3} h_{x_t}^i(u_t^i, u_{t'}^i)\right)^{\gamma} - 1 & \text{if } x_t = x_{t'}, \\ -1 & \text{if } x_t \neq x_{t'}, \end{cases} \tag{5}$$

where u_t^i is the mean value of all image pixels belonging to the t-th superpixel in the i-th color channel and γ is a tuning parameter, 0.05 in our case. We build h_x^i for each color channel separately and then construct the joint probability of a color pair as the product of color channel histograms. This construction is justified only when the color channels are independent like in the Lab color space that we use.

Ideally, if $x_t \neq x_{t'}$ in Eq. (5), we should build the neighborhood histograms from superpixel pairs where each superpixel in the pair belongs to a different class. However, the seeds do not give us such superpixel pairs. Therefore, we set the quality to -1 to penalize neighbors with different labels and to keep the segmentation smooth.

The proposed smoothness term in Eq. (5) compensates for the lack of the gradient-based term in Eq. (4) by combining neighborhood relations learnt from some samples of textures with the smoothness of the Ising prior, Eq. (3).

3.3. Data term

The data term $g_t(x_t)$ encodes the quality of assigning a label x from the set $\mathcal{X}$ to an object/superpixel t in the graph. The quality measures how the superpixel with its neighborhood suits the learnt particular class models.

There are approaches, often used with MIN-CUT, to model classes non-parametrically using color histograms [12] or parametrically using Gaussian Mixture

Figure 5. Data term for three classes from Fig. 6.

Models (GMM) on color histograms [1, 18, 8]. However, none of them takes texturedness directly into account.

We propose a codebook approach to model textures in a more intuitive way similar to [13], however, here improved to be usable on superpixels. We use the seeds and superpixels in them to learn the codebook of feature vectors (etalons) for each class x.

LEARNING THE CODEBOOKS. Let the feature vector from t-th superpixel marked by a label x be $\mathbf{f}_t^x = [\mathbf{u}_t^\top, \mathbf{u}_t^{min\top}, \mathbf{u}_t^{max\top}]^\top$ where $\mathbf{u}_t$ is a 3-element color vector composed of mean values of all pixel colors in the t-th superpixel and

$$\mathbf{u}_t^{min} = \underset{\mathbf{u}_k \in \mathcal{S}(t)}{\operatorname{argmin}} \|\mathbf{u}_t - \mathbf{u}_k\|^2, \qquad \mathbf{u}_t^{max} = \underset{\mathbf{u}_k \in \mathcal{S}(t)}{\operatorname{argmax}} \|\mathbf{u}_t - \mathbf{u}_k\|^2,$$

where $\mathcal{S}(t) \subset \mathcal{E}$ defines the neighborhood (of a given depth) around the t-th superpixel respecting the neighborhood structure $\mathcal{E}$ of the whole graph. We use the depth 2, *i.e.* for a particular superpixel we search its neighbors and also the neighbors of the neighbors.

We sequentially process all seeds, *i.e.* we compute $\mathbf{f}_t^x$ for all superpixels in the seeds and perform simple clustering to create code book $\mathcal{B}^x$ for each class $x = 1...K$. The clustering is done such that before adding a new feature vector f_t^x to the codebook $\mathcal{B}^x$ we first compare it with all already stored vectors j and add it only if the minimum distance $d(j) = \|\mathbf{f}_t^x - \mathbf{f}_j^x\|$ is above a predefined threshold, *i.e.* $\min_{j=1...|\mathcal{B}^x|} d(j) > \theta$. Otherwise, we vote for already stored vector $\mathbf{f}_{j^*}^x$, where $j^* = \operatorname{argmin}_j d(j)$ by increasing a counter $c_{j^*}^x$. The counter $c_{j^*}^x$ expresses the number of samples represented by the etalon vector and is stored together with $\mathcal{B}^x$.

FILLING THE GRAPH. Given $\mathcal{B}^x$, for each t-th superpixel in the image we set the data term as follows

$$g_t(x) = -\exp\Bigg(\alpha \min_{j=1...|\mathcal{B}^x|} \Big(\beta\|\mathbf{u}_t - \mathbf{f}_{j,1:3}^x\|^2 + \min_{k\in\mathcal{S}(t)} \|\mathbf{f}_{j,4:6}^x - \mathbf{u}_k\|^2 + \min_{k\in\mathcal{S}(t)} \|\mathbf{f}_{j,7:9}^x - \mathbf{u}_k\|^2\Big)/c_j^x\Bigg), \quad (6)$$

where α, β are parameters fixed in our experiments to -1200 and 20, respectively. The index notation in $\mathbf{f}_{j,1:3}^x$ means first three elements of $\mathbf{f}_j^x$.

Notice the strategy of the comparison in Eq. (6). Unlike standard approaches, no feature vector is computed for the t-th superpixel for which the quality g_t is being set. Instead, the feature vector from $\mathcal{B}^x$ which best fits to the t-th superpixel and its neighborhood is searched for. The standard approaches would compute a feature vector at the t-th superpixel and its neighborhood and then would compare it to all feature vectors in $\mathcal{B}^x$ to find the closest match. However, this would lead to poor performance at the superpixels which are close to boundaries between textures because feature vectors would be computed from neighborhoods containing more then one texture. Such feature vectors would not match any codebook vector because the codebook vectors are computed only from neighborhoods lying within the same texture.

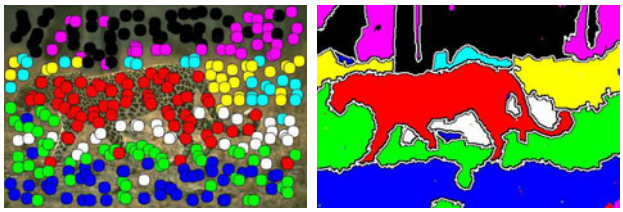


Figure 6. Seed positioning. Left: Clustered seeds at positions chosen not to cross texture boundaries. Seeds with same color correspond to the same class/label. Right: A final segmentation through the MAX-SUM solver into 8 classes.

Fig. 5 shows the data term $g_t(x_t = \{1, 2, 3\})$ for 3 out of 8 classes, one corresponding to the leopard itself, one to the area below the leopard and one to the area behind it. Even though the data terms overlap at some positions because of the leopard skin's similarity to the background, the final labeling correctly splits the textures, see Fig. 6.

4. Seeds

In previous section we formulated the smoothness and data terms exploiting some priors. The priors are learnt from representative seeds provided either manually or automatically, as explained further in this section. At this stage, the superpixels are ignored and everything is done on original pixels.

We build on the fact that small image patches consistent in color and texture can provide sufficient information about possible classes in the image. The idea appeared in [12] where the seeds were first placed regularly over the image and then slightly shifted to avoid crossing texture boundaries. However, in [12] the computationally expensive color and texture gradient [10] was used to do the positioning, making the strategy impractical.

Here we also start with a regular grid of initial positions of seeds but we propose a new strategy to position them. A parameter here is the radius r of the seed, in our case 10 pxls. Each seed, *i.e.* its central point, can be moved in a small rectangle $\mathcal{R}$ with size $2r \times 2r$ centered at the seed's initial position. For each initial seed position we do the following optimization to find a new optimal position avoiding

crossing texture boundaries $\mathbf{s}^*$, *i.e.*

$$\mathbf{s}^* = \underset{\mathbf{s}\in\mathcal{R}}{\operatorname{argmin}} \left(\chi^2(h_{12}, h_{34}) + \chi^2(h_{14}, h_{23})\right), \quad (7)$$

where h_{ij} stands for $h_{ij}(\mathbf{s}, r)$ meaning the histogram computed from pixels in the i- and j-th quadrant of the circle with radius r centered at $\mathbf{s}_{2\times1}$, and χ^2 is a histogram test statistic. The histogram h_{ij} consists of 4 n-bin sub-histograms, $n = 64$ in our case. The first 3 histograms are the color histograms and fourth one corresponds to a texton histogram. Textons, as a filter bank approach for texture description, are computed following [10].

After positioning the seeds obeying Eq. (7), the histogram vector is computed for each shifted seed from all pixels of the seed. As before, the histogram vector is composed of 3 color and 1 texton histogram. Such $4 \times n$ histogram vectors are fed into a K-means clustering method partitioning the seeds into a predefined number of classes K and assigning each seed a particular label, see Fig. 6.

5. Experimental Evaluation

A common problem related to the evaluation of segmentation methods is the difficulty to objectively measure the quality of the segmentation. However, a Global Consistency Error (GCE), also used here, introduced in [11] was shown to be a reasonable way of comparison of algorithms to manual human segmentations.

We used 200 color images and their corresponding human segmentations in the test group of the Berkeley Segmentation Dataset [11]. For each of the images, at least 5 segmentations produced by different persons are available. For each image, the GCE of the segmentation produced by the tested algorithm with respect to each of the available human segmentations for that image was calculated. The mean of these values gives the mean GCE per image plotted in a histogram. The global GCE as one number was calculated as the mean of these 200 mean GCE values.

We compared human segmentations to each other to get the best possible expectation which can be achieved by automatic methods. The aim is to be, in terms of the distribution of the histograms, as close as possible to the histogram corresponding to the comparison human vs. human, shown by the first histogram in Fig. 8. We tested state-of-the art methods handling textures, namely normalized cuts (NCUTS) [10], SEEDSEG [12] based on MIN-CUT, both using the combined texture and color gradient, and FOW [5] based on learning affinity functions.

It can be seen from the histograms, Fig. 8, as from the global GCE, Tab. 1, that our proposed method achieves in terms of the GCE the best result. In general, the method performs well, see Fig. 1, 7 and www[1], and gives satisfactory results even though that current texture descriptor, based on color variation only, cannot distinguish between two textures having different structure but consisting of the same color pairs. Nevertheless, in such cases, the codebook vectors could easily be augmented by more features while preserving the proposed strategy of the inference stage in Eq. (6).

Figure 7. Results on some images for 8 classes.

method	# of regions	GCE
HUMAN	17	0.080
proposed	*8*	*0.178*
SEEDSEG	4	0.209
FOW	13	0.214
NCUT	8	0.285

Table 1. Comparison of the methods. The first column stands for method acronyms, the second corresponds to the average number of regions per image. The third column shows the global GCE over all segmentations.

The implementation of the NCUTS used (provided on the authors' web page augmented by the texture and color gradient) requires that the number of regions is passed as a parameter. We used 8, the same as for the proposed method. The segmentations by the FOW[2] and the SEEDSEG[3] method were provided directly by the authors. In both, the number of segments is achieved automatically.

The proposed segmentation method was implemented in MATLAB and the most time consuming routines in C interfaced through mex-files. The code of the MAX-SUM solver was downloaded from [6]. The method takes on average 50 seconds on a Pentium 4@2.8GHz on 321×481 images with $|\mathcal{T}| = 12500$ objects, for $K = 8$ classes.

We also tried the MAP solver by Kolmogorov [7] and compared it to the Werner's solver [20] on all 200 images. They both lead to almost identical results in most cases in terms of labeling, but differ in running time, see Fig. 9. The difference in labeling in some cases is caused by insufficient number of iterations for the Kolmogorov's solver required as a parameter, set to 100 for whole dataset. The Werner's

[1] www.prip.tuwien.ac.at/Research/muscle/Images/CVPR07

[2] www.cs.berkeley.edu/∼fowlkes/BSE/cvpr-segs

[3] www.prip.tuwien.ac.at/Research/muscle/Images/ECCV06

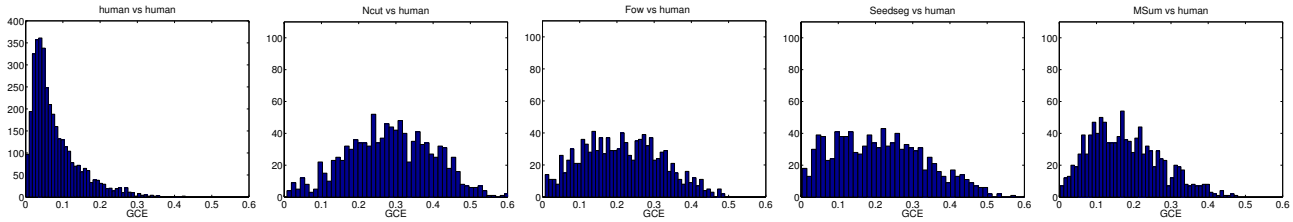


Figure 8. Histograms of Global Consistency Error (from left) for human vs. human, NCUTS, FOW, SEEDSEG, and our proposed method based on the MAX-SUM solver.

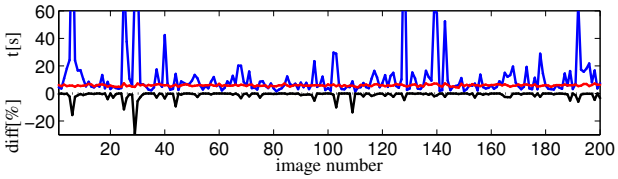


Figure 9. Speed comparison of the MAX-SUM solver by Werner [20] (blue curve with the higher variance) to one by Kolmogorov [7]. Curve below zero axis depicts the percentage of label differences.

solver estimates it automatically causing sometimes higher running time, however a better suboptimal solution, when the classes are very similar and thus harder separable, see peaks in Fig. 9.

6. Conclusion

As a contribution to the improvement of single-image segmentation methods, we proposed a new strategy of encoding priors based on color and texture cues into the MRF built from superpixels. We show that the recently introduced MAP solvers [7, 20] with new design priors offer an efficient image segmentation framework.

References

[1] A. Blake, C. Rother, M. Brown, P. Perez, and P. S. Torr. Interactive image segmentation using an adaptive GMMRF model. In *Proc. ECCV*, pages I: 428–441, 2004.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9):1124–1137, 2004.

[3] P. Carbonetto, N. de Freitas, and K. Barnard. A statistical model for general contextual object recognition. In *Proc. ECCV*, pages 350–362, 2004.

[4] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004.

[5] C. Fowlkes, D. Martin, and J. Malik. Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches. In *Proc. CVPR*, pages II: 54–61, 2003.

[6] `http://cmp.felk.cvut.cz/cmp/software/maxsum/`.

[7] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 28(10):1568–1583, 2006.

[8] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Probabilistic fusion of stereo with color and contrast for bi-layer segmentation. *PAMI*, 28(9):1480–1492, 2006.

[9] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Solving Markov random fields using second order cone programming. In *Proc. CVPR*, pages I: 1045–1052, 2006.

[10] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *IJCV*, 43(1):7–27, 2001.

[11] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. ICCV*, pages 416–425, 2001.

[12] B. Mičušík and A. Hanbury. Automatic image segmentation by positioning a seed. In *Proc. ECCV*, pages II: 468–480, 2006.

[13] B. Mičušík and A. Hanbury. Template patch driven image segmentation. In *Proc. BMVC*, pages II: 819–829, 2006.

[14] N. Paragios and R. Deriche. Geodesic active regions and level set methods for supervised texture segmentation. *IJCV*, 46(3):223–247, 2002.

[15] M. Schlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, (4):113–130, 1976. In Russian.

[16] M. Schlesinger and B. Flach. Analysis of optimal labelling problems and application to image segmentation and binocular stereovision. In *Proc. East-West-Vision Workshop*, pages 65–81, 2002.

[17] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[18] J. Shotton, J. Winn, C. Rother, and A. Criminisi. *TextonBoost:* Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proc. ECCV*, pages I: 1–15, 2006.

[19] Z. Tu, X. Chen, A. Yuille, and S. Zhu. Image parsing: Unifying segmentation, detection, and recognition. *IJCV*, 63(2):113–140, 2005.

[20] T. Werner. A linear programming approach to Max-sum problem: A review. *PAMI*, 2007.

[21] Y. Wexler, A. Fitzgibbon, and A. Zisserman. Bayesian estimation of layers from multiple images. In *Proc. ECCV*, pages III: 487–501, 2002.

[22] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.