Energy Minimization by α-erosion for Supervised Texture Segmentation

Karl Skretting and Kjersti Engan

Department of Electrical Engineering and Computer Science,
University of Stavanger, Norway.

Abstract. In this paper we improve image segmentation based on texture properties. The already good results achieved using learned dictionaries and Gaussian smoothing are improved by minimizing an energy function that has the form of a Potts model. The proposed α-erosion method is a greedy method that essentially relabels the pixels one by one and is computationally very fast. It can be used in addition to, or instead of, Gaussian smoothing to regularize the label images in supervised texture segmentation problems. The proposed α-erosion method achieves excellent results on a much used set of test images: on average we get 2.9% wrongly classified pixels. Gaussian smoothing gives 10% and the best results reported earlier give 4.5%.

1 Introduction

Image segmentation has many important applications in the image processing field, mainly since it is a common step in scene interpretation, and it is used in areas such as medical diagnostics, geophysical interpretation, industrial automation and image indexing. For image segmentation, edges and colors are often more important features than texture. Nevertheless, the texture property is relevant in many image processing applications [11]. An important benchmark application to test how well a system can utilize texture information that may be present in an image, is to segment the image based on texture properties alone. This is what we do here.

Texture segmentation finds a boundary map between different texture regions of an image. This map may be given by a labeling L which assigns a label $L_p \in \{1, 2, \ldots, C\}$ to every pixel $p \in \mathcal{P}$ of the observed image, and C is the number of candidate texture classes. A common way to label the image is to associate a *feature vector* x_p to every pixel in the image and then do common *vector classification*. This approach, however, ignores the fact that texture regions should be piecewise constant in the labeling. Gaussian smoothing of the features before classification will give larger segments and has been much used [13].

An alternative to Gaussian smoothing is Energy Minimization (EM) [1–3]. For each pixel (feature vector) an associated *cost vector* y_p is calculated, element c gives the cost (energy) for assigning class c to this pixel. The cost may be the

estimated negative log probability for pixel p belonging to class c. The set of energy images $R = \{R^{(c)}\}_{c=1}^{C}$, where pixel p in energy image c is $R_p^{(c)} = y_p(c)$, can now be used to calculate the data cost for a given labeling L. In addition to the data term $E_d(\cdot)$ a smoothing term $E_s(\cdot)$ is added to the energy function, thus the problem of finding the "best" labeling can be formulated as an energy minimization problem using

$$E(L, R) = E_d(L, R) + E_s(L). \tag{1}$$

When piecewise constant labeling is desired the *Potts model* is a popular choice for the smoothing term. It can be formulated as equation (18) in [3]:

$$E(L, R) = \sum_{p \in \mathcal{P}} R_p^{(L_p)} \; + \sum_{p,q \in \mathcal{N}} u_{p,q} \cdot (L_p \neq L_q). \tag{2}$$

$R_p^{(L_p)}$ is the cost (energy) associated with assigning label L_p to pixel p. $\mathcal{N}$ is the set of all neighboring pixel pairs and $(L_p \neq L_q)$ is a logical expression evaluating to 1 if the two labels are different. The factor $u_{p,q}$ is independent of the labeling but may depend on the pair (p, q) [3].

To minimize an energy function of the form as in Eq. 1 or Eq. 2 is in general an NP-hard problem [3] except for some simple cases; the two label-problem can be minimized via graph cuts [7]. One method to approximately minimize the energy function for the several-label problem is to start with an initial labeling and then do a sequence of *moves*; each move may change the labeling of some of the pixels under consideration. Two popular moves are the α-β-swap-move where the optimal division between the two labels in a two label region is found, and the α-expansion-move where "not-α-pixels" may be relabeled as α [3]. The optimal solution for each of these two moves can be found by graph-cuts-algorithms.

A crucial point for the success of EM is that the energy function Eq. 1 reflects the texture segmentation. For many test images this is often only an approximation where the ground truth labeling L^{gt} gives a higher value for the energy function than the value obtained by EM using expand-moves L^{ex} or swap-moves L^{sw}, as seen in Fig. 3. In these cases further improvement in the segmentation can be achieved by: 1) Better observations and better features which gives better cost vectors and energy images. This is important, but in this work we do not investigate this part any more but simply accept a method to make feature vectors that have worked well previously [16]. 2) Better and more sophisticated energy function, more advanced forms are proposed in [6, 4], may improve segmentation. Both how to define a better form, and then how to minimize the energy function, are difficult tasks which we here leave to future work. 3) Inferior method for energy minimization, i.e. even if the method gives higher energy than another method it may be better when it comes to segmentation. This latter approach is used in this work. We propose a method that usually does not find the minimum of the energy function, even though all the moves it makes reduce it. The algorithm is designed such that each move tries to reduce the number of segments in the labeled image, and in this way follows

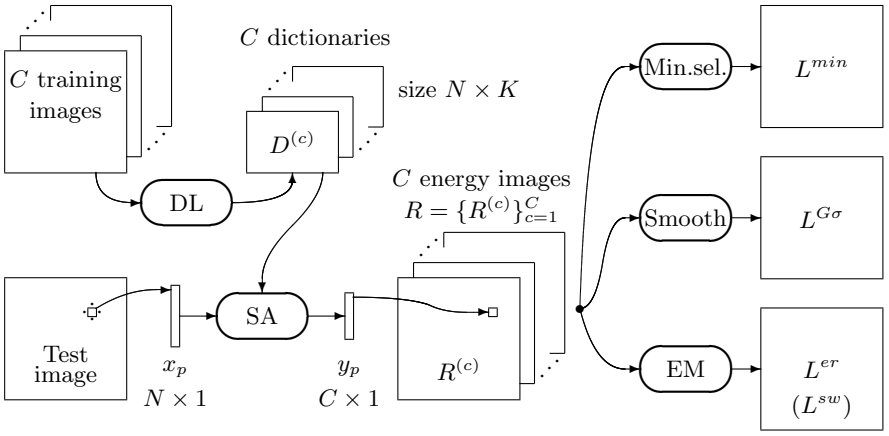

Fig. 1. The Frame Texture Classification Method (FTCM). Dictionary Learning (DL) learn one dictionary with K atoms for each candidate texture. The feature vector x_p is made from a neighborhood of pixel p. Sparse Approximation (SA) uses the candidate dictionaries and gives the energy vector y_p where element c is pixel p in error image $R^{(c)}$. Different labelings are made by the minimum selector, the Gaussian smoothing, and the energy minimization (EM) methods.

a path that is intuitive in image segmentation (few segments). In addition the proposed method is fast, up to 100 times faster than the graph-cuts-methods.

2 Frame Texture Classification Method (FTCM)

The Frame Texture Classification Method (FTCM) [14, 16] is used to generate the data term in the energy function of Eq. 1, an overview is shown in Fig. 1. It is based on sparse representation and dictionary learning, for details on these parts see [5, 15].

FTCM generates a simple feature vector x_p for each pixel p in a test image using pixel values from its neighborhood directly. Sparse approximations of x_p are made using dictionaries learned for each of the candidate textures. An energy vector y_p is then calculated from the approximation errors,

$$y_p(c) = R_p^{(c)} = \|x_p - D^{(c)} w_p^{(c)}\|_2. \tag{3}$$

where $w_p^{(c)}$ is the $K \times 1$ sparse coefficient vector used when vector x_p is approximated by dictionary $D^{(c)}$. A simple segmentation scheme can be to assign class labels for each pixel, according to the minimum selector method

$$L_p^{min} = c \quad \text{if} \quad R_p^{(c)} \leq R_p^{(k)} \quad \forall \quad k. \tag{4}$$

Segmentation can however be substantially improved by smoothing the C energy images $\{R^{(c)}\}_{c=1}^{C}$ prior to the final labeling. A common smoothing approach has

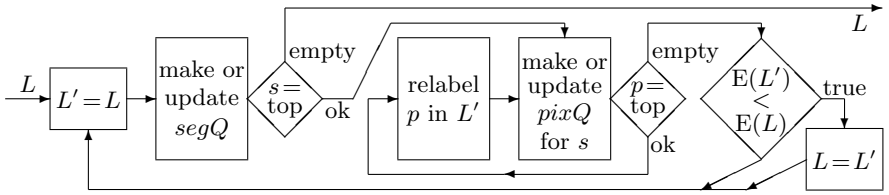

Fig. 2. The main flow for the erode algorithm. The input labeling L is processed segment by segment, the unprocessed segments are kept in a queue, *segQ*. The smallest segment, s from the top of the queue, is processed by relabeling the pixels one by one in L' and if this improves energy the labeling L is updated (bottom right box in figure).

been to use the Gaussian low-pass filter $G_\sigma(\cdot)$ where the parameter σ gives the width of the filter. The smoothed labeling can be denoted:

$$L_p^{G\sigma} = c \quad \text{if} \quad G_\sigma(R^{(c)})_p \leq G_\sigma(R^{(k)})_p \quad \forall \quad k.$$

3 Erode algorithm for energy minimization

The energy function used here is a variant of the Potts model Eq. 2, restricted to 4-neighborhood $\mathcal{N}_4$ and 8-neighborhood $\mathcal{N}_8$ systems:

$$E(L,R) = \sum_{p\in\mathcal{P}} R_p^{(L_p)} + \frac{\lambda}{k_4}\Big(\sum_{p,q\in\mathcal{N}_4} (L_p \neq L_q) + b \sum_{p,q\in(\mathcal{N}_8\setminus\mathcal{N}_4)} (L_p \neq L_q)\Big). \qquad (5)$$

where λ gives the weight of the smoothing term and $k_4 = |\mathcal{N}_4| = \sum_{p,q\in\mathcal{N}_4} 1$. The factor $b \in [0,1]$ gives the relative weight for corner connected pixels pairs to side connected pixels pairs. For the data term the set of the C energy images $R = \{R^{(c)}\}_{c=1}^{C}$ is scaled such that $E_d(L^{min}, R) = 0$ and $E_d(L^{max}, R) = 1$. The labeling L^{min} is defined in Eq. 4 and L^{max} is defined as $L_p^{max} = c \quad \text{if} \quad R_p^{(c)} \geq R_p^{(k)} \quad \forall \quad k$.

The proposed energy minimization algorithm is a greedy algorithm that reduces the value of the objective function in each move, an overview is shown in Fig. 2. The main idea behind the algorithm is quite simple, it is based on the assumption that removing a small segment is more likely to reduce the value of the objective function than to remove a larger segment. A segment is defined as a 4-neighborhood *connected component* of the current labeling L. The α-erosion method starts with the current labeling and do a sequence of α-erode-moves, as described below. The labeling after all the α-erode-moves is denoted L^{er}, and an example can be seen in Fig. 3d. The second idea is that the border between two segments should be smooth, here we found that using an extra erode-move (or swap-moves) on the border region worked well, doing these moves the labeling is denoted as L^{er+}, an example can be seen in Fig. 3e.

Each erode-move considers all pixel within one label segment, s in Fig. 2. A queue of the pixels *pixQ* is made, it is ordered by the effect relabeling this pixel

a) Test image | b) Ground truth L^{gt} | c) Gaussian filtering L^{G3}
33 ms, error rate 21.41%

$E = E_d + \lambda E_s$: | 0.3976 = 0.3487 + 0.0489 | 0.6531 = 0.3126 + 0.3405

d) α-erosion L^{er}
35 ms, error rate 4.05%
0.4001 = 0.3425 + 0.0576

e) α-erosion + border L^{er+}
56 ms, error rate 2.61%
0.4001 = 0.3470 + 0.0531

f) α-β-swap L^{sw}
2255 ms, error rate 4.50%
0.3883 = 0.3414 + 0.0469

Fig. 3. Test image number 4 and different labeling results. The second line below each image shows the execution time (in ms) and the error rate, the third line shows the value of the objective function, $E = E_d + \lambda E_s$, here $\lambda = 3$ (and $b = 0.7$ in Eq. 5). The label images are hopefully shown in color but a grayscale image will also show the different segments even though the center segment and the bottom segment then look quite similar to each other. The problem with corners is because the smoothing term favors short, rather than straight, border lines.

will have on the objective function. The pixels are now relabeled and removed from the top of the queue, and the remaining queue is updated. In this way it is like the surrounding segments "eat" the pixels of the segment s one pixel at a time, we may also say that the segment is eroded. If the objective function value actually does decrease after the whole segment is "eaten" the "meal" is accepted, i.e. the segment is relabeled. If not the labeling is kept as it was. After a segment is relabeled the segment queue $segQ$ is ordered by size again, i.e. updated. The α-erode-move is tried on all segments smaller than a given limit in the label image.

4 Experiments and discussion

The training and test grayscale texture images used here are from the *Outex test suites* [10] and are available from Outex web page[1]. The test set Contrib_SS_-00000 consists of 12 test images. It has been used in several papers on texture segmentation; some results are collected in Table 1. The test images are shown in [13, 8] and on the UiS web page[2] where also more results and some Matlab files used in this paper can be found. The UiS web page also gives more details on how the parameters, like dictionary size $N \times K$, sparseness factor s, and factors λ and b in Eq. 5 were selected, many are simply used as in earlier papers [16, 15] and some are set empirically.

The results for the fourth test image are presented in Fig. 3. The data term was made by FTCM with dictionary size 17×200. Order recursive matching pursuit (ORMP) was used for sparse approximation, and the target sparseness factor was $s = 3$. Learning was done using a mini-batch variant of RLS-DLA [15] using a forgetting factor starting at 0.996 and growing towards 1 during learning, processing 4 million training vectors (many are reused) for each dictionary. For each test image the sparse representations, using only the relevant dictionaries, give the set of energy images R. In Fig. 3 the Gaussian smoothing labeling L^{G3} is shown as this is used as initial labeling for the erode- and swap-moves. Energy minimization is done using α-erosion as described in Sec. 3, both the labeling L^{er} and L^{er+} are shown in Fig. 3. Also the labeling using only α-β-swap-moves until convergence L^{sw} is shown. The labeling for α-expansion-moves L^{ex} were similar to L^{sw} and is not shown.

The results of the proposed method are compared to those reported by other works in Table 1. Randen used filtering methods, Ojala Local Binary Pattern (LBP) and Mäenpää Multi-Predicate (MP) LBP. Skretting used FTCM with reconstructive dictionaries and Gaussian smoothing. Mairal used learned discriminative dictionaries and EM by α-expansion-moves (D-EMex). The last line of Table 1 shows the results for the proposed method, i.e. the α-erosion method followed by erode-moves to straighten the border lines.

Table 2 also shows results for Gaussian smoothing, α-β-swap and α-expansion, and erode with and without extra border region erode-moves. The results for Gaussian smoothing are only marginally better than results of [14], average for the test images 1 to 9 is 12.95% errors, in [14] it was 13.2%, Table 1. The small improvement is because the 17×200 sized dictionaries used here are marginally better than the ones used ten years ago, sized 25×100. The major improvement here is due to the energy minimization (EM) used for regularization. All EM methods do significantly better than Gaussian smoothing. The results for the α-β-swap L^{sw} and α-expansion L^{ex}, both using $\lambda = 1.5$, are comparable to what Mairal reported for discriminative dictionaries in [9]. The proposed methods L^{er} and L^{er+}, here using $\lambda = 2.0$, are both fast and effective, as seen in Table 2. Especially, using extra erode-moves on border regions L^{er+} achieves

[1] Outex web page: `http://www.outex.oulu.fi/`
[2] UiS web page: `http://www.ux.uis.no/~karlsk/tct/`

Paper and method	Avg. 1-9	Avg. 1-12
Randen 1999 (best) [13]	24.1	18.4
Mäenpää 2000 MP-LBP [8]	13.8	10.9
Skretting 2001 FTCM [14]	13.2	-
Ojala 2001 LBP [12]	15.2	12.4
Mairal 2008, D-EMex [9]	5.84	4.50
This paper, FTCM-EMer	3.63	2.87

Table 1. Reported average error rates for texture classification on the set of test images. The second column is average for images 1 to 9, and the last column is average for all 12 test images.

impressing segmentation. Using extra swap-moves on border regions, this case is not included in Table 2, also worked very well. The execution times were 10-50 percent of L^{sw}, and for one case ($\lambda = 3$) the achieved average error rate was impressing 2.74%.

5 Conclusions

The proposed α-erosion method is a greedy method that tries to minimize an objective function based on the Potts model and can be used to regularize a label image. It is shown that the α-erosion method achieves results close to those achieved by methods based on the graph-cut algorithm, i.e. the α-β-swap and α-expansion methods, but *is much faster*.

For a common set of test images the average segmentation error rate is even better for the α-erosion method than for the graph cut based methods, even though the achieved values for the objective function are not. The best average error rate achieved here (2.74%) is better than all earlier reported results.

References

1. J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):259–302, 1986.
2. Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Machine Intell.*, 26(9):1124–1137, September 2004.
3. Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Machine Intell.*, 23(11):1222–1239, November 2001.
4. A. Delong, L. Gorelick, O. Veksler, and Y. Boykov. Minimizing energies with hierarchical costs. *Int. J. Comput. Vision*, 100(1):38–58, October 2012.
5. M. Elad. *Sparse and Redundant Representations, from Theory to Applications in Signal and Image Processing.* Springer, New York, USA, 2010.
6. P. Kohli, L. Ladický, and P. H. S. Torr. Robust higher order potentials for enforcing label consistency. *Int. J. Comput. Vision*, 82(3):302–324, May 2009.

image	Gaussian filter		Energy minimization				Execution time [s]			
no.	L^{G3}	L^{G12}	L^{sw}	L^{ex}	L^{er}	L^{er+}	L^{G3}	L^{sw}	L^{er}	L^{er+}
1	6.08	8.15	3.31	3.38	4.34	2.00	0.038	1.037	0.026	0.072
2	15.42	10.85	2.85	2.75	3.88	3.24	0.037	2.123	0.032	0.068
3	25.33	11.41	2.56	2.56	3.47	4.01	0.038	2.961	0.033	0.063
4	21.41	9.31	4.34	4.34	5.08	2.55	0.040	1.304	0.031	0.064
5	18.20	6.57	2.47	2.47	3.86	1.26	0.040	2.000	0.028	0.058
6	33.69	21.25	10.56	10.74	7.98	6.72	0.369	26.1	0.241	0.334
7	37.09	19.99	3.80	8.49	4.20	4.14	0.355	25.7	0.238	0.317
8	34.94	16.03	12.77	12.68	6.23	4.80	0.153	11.9	0.106	0.151
9	38.57	13.02	2.07	2.02	5.61	3.90	0.155	11.5	0.107	0.160
10	2.27	0.34	0.57	0.57	0.66	0.42	0.038	0.172	0.032	0.064
11	2.44	2.04	1.09	1.09	1.50	0.61	0.034	0.278	0.031	0.060
12	12.00	1.91	3.79	3.79	5.55	0.70	0.033	0.737	0.043	0.080
Average	20.62	10.07	4.18	4.57	4.36	2.87				

Table 2. Percentage of wrongly classified pixels for the 12 test images and execution time for different methods. Note that L^{er+} use extra erode-moves on border regions, as in Fig. 3e.

7. V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts. *IEEE Trans. Pattern Anal. Machine Intell.*, 26:147–159, 2004.
8. T. Mäenpää, M. Pietikäinen, and T. Ojala. Texture classification by multi-predicate local binary pattern operators. In *Proc. ICPR*, 2000.
9. J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2008.
10. T. Ojala, T. Mäenpää, M. Pietikäinen, J. Viertola, J. Kyllönen, and S. Huovinen. Outex - new framework for empirical evaluation of texture analysis algorithms. *Proc. 16th Int. Conf. Pattern Recognition*, 2002.
11. T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Machine Intell.*, 24(7):971–987, July 2002.
12. T. Ojala, K. Valkealahti, E. Oja, and M. Pietikäinen. Texture discrimination with multidimensional distributions of signed gray-level differences. *Pattern Recognition*, 34(3):727–739, March 2001.
13. T. Randen and J. H. Husøy. Filtering for texture classification: A comparative study. *IEEE Trans. Pattern Anal. Machine Intell.*, 21(4):291–310, April 1999.
14. K. Skretting. *Sparse Signal Representation using Overlapping Frames.* PhD thesis, NTNU Trondheim and Høgskolen i Stavanger, October 2002. available at `http://www.ux.uis.no/~karlsk/`.
15. K. Skretting and K. Engan. Recursive least squares dictionary learning algorithm. *IEEE Trans. Signal Processing*, 58:2121–2130, April 2010.
16. K. Skretting and J. H. Husøy. Texture classification using sparse frame based representations. *EURASIP Journal on Applied Signal Processing*, 2006, 2006. Article ID 52561.