Image Segmentation with Use of Cross-Entropy Clustering*

Marek Śmieja and Jacek Tabor

Abstract. We present an image segmentation approach which is invariant to affine transformation – the result after rescaling the picture remains almost the same as before. Moreover, the algorithm detects automatically the correct number of groups. We show that the method is capable of discovering general shapes as well as small details by the appropriate choice of only two input parameters.

1 Introduction

The image segmentation is a technique which relies on partitioning the image into several regions according to the selected features, for example, the pixel value. It is widely applied in pattern recognition, computer vision and image analysis.

We distinguish various categories of segmentation methods like edge-based or region-based segmentations [4,5,11]. One of the most important class includes techniques based on clustering [6,8]. Such algorithms interpret pixels (or more precisely the groups of pixels) as high dimensional vectors and divide them into groups. Unfortunately, popular clustering methods suffer very often from many drawbacks. k-means algorithm does not detect the correct number of clusters, tends to form groups of similar sizes and is sensitive to affine transformations. More advanced methods as EM (Expectation Maximization) give better results but involve high computational complexity [2,9,10].

In this paper a Gaussian cross-entropy clustering (CEC) will be applied to the image segmentation [12]. It can be seen as joining of k-means with EM but is free

Marek Śmieja · Jacek Tabor
Institute of Computer Science, Department of Mathematics and Computer Science,
Jagiellonian University, Lojasiewicza 6, 30-348, Krakow, Poland
e-mail: {marek.smieja,jacek.tabor}@ii.uj.edu.pl

of all of the aforementioned disadvantages. The algorithm splits data according to the mixture of Gaussian distributions. Its implementation with use of Hartigan approach has similar computational complexity to classical k-means but has two important features – it reduces irrelevant clusters on-line and is invariant to affine transformations.

The illustration of CEC properties and advantages is presented in the next section. Third section contains deeper overview of the method with the practical explanation how to perform segmentation with use of CEC. Our experiments are conducted on gray-scale images selected from Berkeley Segmentation Database [1].

2 Illustration of the Method

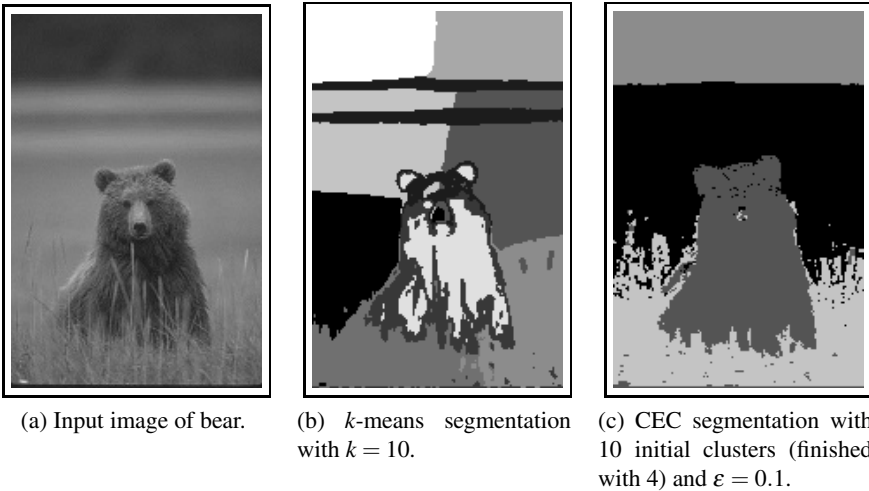In this section we will discuss on the examples the properties of the image segmentation with use of CEC.

(a) Input image of bear.

(b) k-means segmentation with $k = 10$.

(c) CEC segmentation with 10 initial clusters (finished with 4) and $\varepsilon = 0.1$.

Fig. 1 Detection of the correct number of clusters with use of CEC (c) compared to classical k-means (b)

First experiment concentrates on the demonstration that, in contrast to k-means, CEC is capable of discovering the correct number of groups in the picture.

Example 1. Let us consider an image of a bear showed in Figure 1a. CEC finds the optimal number of segments by the appropriate set of only one input parameter ε – the level when to remove redundant clusters (the meaning of this parameter is explained precisely in the next section). As an input of CEC we also give the initial number of clusters – it is simultaneously changing by the algorithm respectively to the value of ε. We can see in Figure 1c that for ten initial clusters and $\varepsilon = 0.1$ the

algorithm detected four segments which suits well our expectations. The input of k-means algorithm has to include the final number of groups. When we put $k = 10$ then the segmentation results are not satisfactory (see Figure 1b).

The next example presents the comparison of the segmentation with use of CEC and k-means with known parameter k.

Example 2. In the previous experiment the correct number of clusters was not known. Here we assume that this quantity is estimated correctly – for the picture of bear we put $k = 4$. Compare the effect of k-means in Figure 2a with the one obtained with use of CEC in Figure 1c. Let us observe that k-means, contrary to CEC, did not detect properly the image of bear (see Figures 2b and 2c).
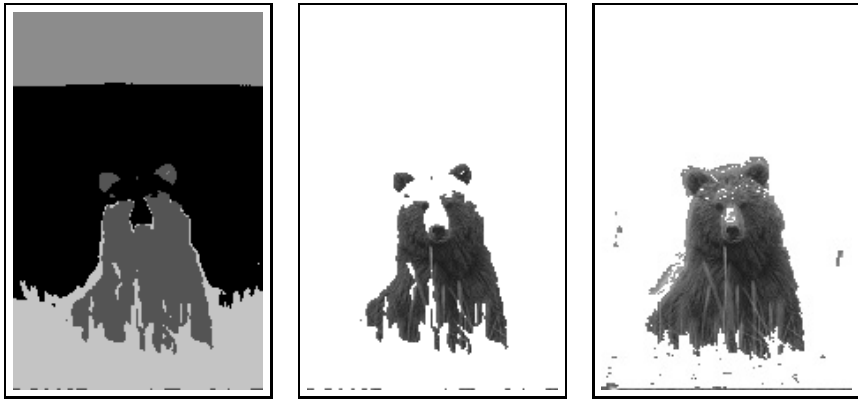
(a) *k*-means segmentation with $k = 4$.

(b) Bear discovered by *k*-means with $k = 4$.

(c) Bear discovered by CEC.

Fig. 2 Segmentation with use of CEC and k-means with $k = 4$

Another important feature of CEC is that it is invariant to any affine transformation – we examine here the segmentation on rescaled image.

Example 3. The clustering methods which rely on metrics are sensitive to scaling. When we resize the image then the segmentation result differs radically from the one with original size. Hence the segmentation based on k-means does not deal well with the scaling.

Let us observe that CEC is free of this drawback since it is based on the data distribution, not on the metric. It is possible to perform various affine transformations – the segmentation result does not differ much from the original one. Clearly, there may appear some differences since the computation on rescaled image takes into account slightly different groups of pixels[2]. However, this is due to the transformation, not the clustering algorithm.

---

[2] For a given pixel the input vector is built with use of the adjacent pixel values (pixel window). Therefore, to consider the same groups of pixels on resized picture we should also changed the pixel widow.

The effects of segmentations on rescaled image from Figure 3a with use of *k*-means and CEC are presented in Figure 3b and 3c.
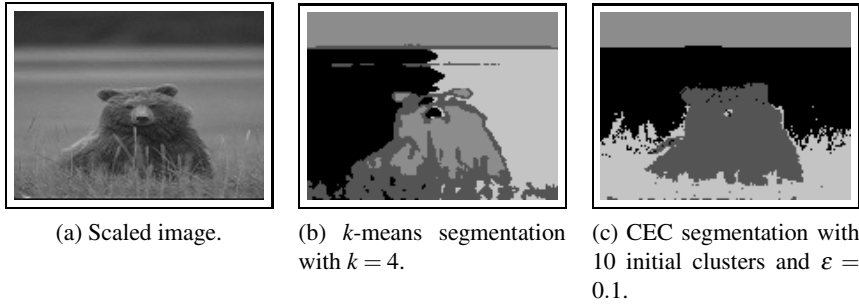

(a) Scaled image. (b) *k*-means segmentation with $k = 4$. (c) CEC segmentation with 10 initial clusters and $\varepsilon = 0.1$.

Fig. 3 Segmentation on rescaled image with use of *k*-means and CEC – compare Figures 3b with 2a and Figure 3c with 1c

3 Performing the Segmentation with Use of CEC

In this section we explain how to perform and control the segmentation with use of CEC. The standard procedure consists of the following steps:

- Select $b \times b$ window around each pixel to use as a feature vector for segmentation (if not stated otherwise then we put $b = 7$).
- Apply PCA (Principal Component Analysis) [7] to these vectors to project them onto their first *n* principal components (we use $n = 4$). Projection speeds up the the clustering algorithm and does not affect much the result of segmentation.
- Supplement every vector with two coordinates of the pixel[3]. We add the coordinates after performing PCA because color and position represent different quantities and are not comparable values.
- Perform CEC on a sequence of $(n + 2)$-dimensional vectors.

We explain more precisely the last step: Gaussian CEC clustering [12] relies on finding the pairwise disjoint splitting $(U_i)_i$ of a data-set $U \subset \mathbb{R}^N$ which minimizes the energy function

$$E(U_1, \ldots, U_k) = \sum_{i=1}^{k} p(U_i) \cdot [\frac{N}{2} \ln(2\pi\varepsilon) - \ln(p(U_i)) + \frac{1}{2} \ln \det(\Sigma_{U_i})], \qquad (1)$$

where $p(U_i) = \frac{\mathrm{card}(U_i)}{\mathrm{card}(U)}$ and Σ_{U_i} denotes the covariance matrix of U_i. The expression (1) corresponds to the mean code length of randomly chosen element with the code optimized for the mixture of *k* Gaussian distributions.

---

[3] We assume that parts of one segment should not appear in completely different regions of the image.

(a) Input image.

(b) High detailed segmentation with $b = 3$ and $\varepsilon = 0.03$.

(c) Standard segmentation with $b = 7$ and $\varepsilon = 0.1$.

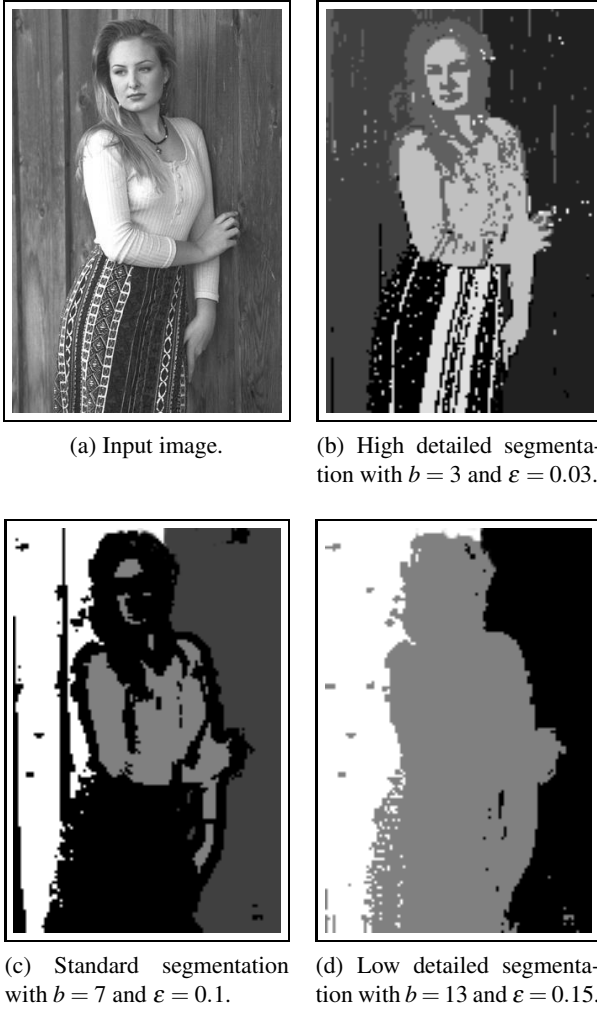(d) Low detailed segmentation with $b = 13$ and $\varepsilon = 0.15$.

Fig. 4 CEC image segmentation with different values of input parameters

To perform the clustering, we apply the well-known Hartigan approach[4]. Roughly speaking, the procedure relies on iteratively reassigning points to these clusters for which the decrease of energy is maximal. Since CEC introduces the cost of maintaining the cluster (which equals $-p(U_i)\ln(p(U_i))$) the method tends to reduct groups which carry no information. We starts the algorithm with an arbitrary number of initial clusters and remove unnecessary one when its probability is less than a fixed number $\varepsilon \in [0,1)$. The algorithm can be implemented efficiently to work in comparable time to classical k-means.

---

[4] The time of the segmentation process depends strictly on the number algorithm iterations – for an image with the resolution 120×180 one iteration leasts about one second.

The effect of the segmentation depends strictly on the size b of the image window and the parameter ε which describes the level when we decide to remove cluster. The following experiment focuses on the revealing the relationship between the choice of these two parameters and the accuracy of the segmentation. Let us observe that higher values of parameter b makes the image smoother – in consequence we remove details. On the other hand if we are interested in discovering only the most distinctive segments then we should set relatively high value of ε. Let us examine this observation on the following example.

Example 4. We are given an image of the person (see Figure 4a) and we try to discover segments with different precision. We ran the algorithm with 10 initial clusters. The results are presented in Figure 4b, 4c, 4d. We can see that Figure 4b ($\varepsilon = 0.03$ and $b = 3$) reveals all details – hair, eyes, mouth and even the dress pattern while Figure 4d ($\varepsilon = 0.15$ and $=13$) finds only the general shape of the person.

4   Conclusion

We presented the application of cross-entropy clustering method in the image segmentation. The main advantage of CEC is that it automatically detects the correct number of clusters. Moreover, it is invariant to affine transformations – the segmentation results after rescaling the image remain almost the same as before scaling. The method with the appropriate choice of only two input parameters is capable of performing the segmentation with various precision. We can conduct general segmentation as well as the detailed one.

References

1. The Berkeley Segmentation Database, `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds`
2. Celeux, G., Govaert, G.: Pattern Recognition 28(5), 781–793 (1995)
3. Cover, T.M., Thomas, J.A., Wiley, J., et al.: Elements of information theory, vol. 6. Wiley Online Library (1991)
4. Deng, Y., Manjunath, B.S.: IEEE Trans. Pattern Anal. Machine Intell. 23(8), 800–810 (2001)
5. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. John Wiley & Sons, New York (1979)
6. Jain, A.K., Murty, M.N., Flynn, P.J.: ACM Computing Surveys 31(3), 264–323 (1999)
7. Jolliffe, I.: Principal component analysis. Encyclopedia of Statistics in Behavioral Science (2002)
8. Ma, Y., Derksen, H., Hong, W., Wright, J.: IEEE Transactions on Pattern Analysis and Machine Intelligence 29(9), 1546–1562 (2007)

9. McLachlan, G.J., Krishnan, T.: The EM algorithm and extensions, vol. 274. Wiley, New York (1997)
10. Mcnicholas, P.D., Murphy, T.B.: Statistics and Computing 18(3), 285–296 (2008)
11. Pei, S.C., Ding, J.J.: ICASSP 3, 357–360 (2003)
12. Tabor, J., Spurek, P.: Cross-entropy clustering (2012),
    `http://arxiv.org/pdf/1210.5594.pdf`