Computer Vision and Image Understanding 130 (2015) 80–86

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

A graph based approach to hierarchical image over-segmentation ☆

Pavel Kalinin *, Aleksandr Sirota

Department of Computer Science, Voronezh State University, Russia

ARTICLE INFO

Article history:
Received 22 July 2013
Accepted 15 September 2014
Available online 6 October 2014

Keywords:
Segmentation
Superpixels
Graph cuts

ABSTRACT

The problem of image segmentation is formulated in terms of recursive partitioning of segments into subsegments by optimizing the proposed objective function via graph cuts. Our approach uses a special normalization of the objective function, which enables the production of a hierarchy of regular superpixels that adhere to image boundaries. To enforce compactness and visual homogeneity of segments a regularization strategy is proposed. Experiments on the Berkeley dataset show that the proposed algorithm is comparable in its performance to the state-of-the-art superpixel methods.

1. Introduction

The goal of image segmentation is to assign to each pixel a label that corresponds to a class of potential objects. It helps to find a convenient representation for further analysis by localizing image objects and boundaries. There are two common approaches to image segmentation. The first aims at extracting regions belonging to a given set of possible objects. The second aims at extracting segments of unknown object classes by grouping pixels that are similar in color, texture, etc.

The first approach maximizes the probability that each pixel receives the correct label. The simplest method is to use a sliding window classifier that estimates the label of each pixel based on its local neighborhood. More advanced approaches take into account the coherence (neighboring pixels tend to have similar labels) [1,2] and semantic context [3]. They often use probabilistic graphical models, e.g. CRF [4]. In this case, their structure is represented by a graph, where image pixels are the vertices, and the edges define the dependencies between pixels. Efficient segmentation is then possible when the edges help to assign similar labels to the vertices.

The second approach aims at extracting the most likely image objects with as few segments as possible. Superpixel algorithms have become popular for solving this kind of problems. They partition images into many relatively small segments. Superpixel segmentation helps to reduce image dimensionality with minimal loss of information and offers an easy way to leverage long-range pixels interdependencies. Regularity and ability to adhere to image boundaries make superpixels a convenient tool for calculating local image descriptors.

Common requirements to superpixel algorithms are:

1. *Performance*. Superpixel segmentation is often used as a preprocessing tool. It, therefore, should take less time than is required for further processing.
2. *Consistency*. Superpixels boundaries should be consistent with image objects boundaries.
3. *Compactness and regularity*. For many applications superpixels should be of similar size and more or less convex shape. Such superpixels help to extract better local descriptors and have fewer neighbors.

2. Related works

Superpixel algorithms can be divided into three main groups: divisive, agglomerative and discriminative.

The first group [5–7] implements a top-down strategy. Initially, the whole image is viewed as a single large segment. Then it is recursively partitioned into subsegments until a stopping criterion is met. The algorithms of this group usually have more complex optimality criteria that make them slower, but less sensitive to noise. For example, the graph cuts and normalized cuts objective functions depend on the sum of the cut edges. Therefore, there is no immediate dependency on the values of individual edges. Many algorithms of this group do not use regional information, such as pixel color, directly, but instead use information about the similarity of pixels, e. g. probabilities of borders. Another advantage is their ability to produce a hierarchy of nested segments in one pass. The superpixel algorithm of Ren and Malik [6] uses spectral graph partitioning to yield compact and regular superpixels. Two more

☆ This paper has been recommended for acceptance by Lena Gorelick.

* Corresponding author.
E-mail addresses: kalinin_pv@sc.vsu.ru (P. Kalinin), sir@sc.vsu.ru (A. Sirota).

http://dx.doi.org/10.1016/j.cviu.2014.09.007
1077-3142/

approaches [5,7] use the minimum graph cut algorithm and dynamic programming to obtain superpixel lattices.

The second group of algorithms [8,9] implements a bottom-up strategy. Initially, each pixel belongs to a separate segment. Then the segments are merged until a certain stopping criterion is met. Image is often represented as a graph where the vertices denote the segments connected to each other by the graph edges representing their similarity. Algorithms of this group are less time consuming thanks to the greedy optimization strategies: at each stage a pair of segments to be merged is the one connected by an edge with the highest similarity value. However, when the segments are small, it is hard to estimate their similarity precisely. Moreover, greedy merging strategies are highly dependent on the values of individual edges. As a result, at the earlier stages dissimilar segments may be merged, causing errors. Therefore, such algorithms are more sensitive to noise. Like algorithms of the first group, they allow to produce a hierarchy of superpixels. The algorithm of Felzenszwalb and Huttenlocher (FH) [8] uses the above ideas to produce segments that adhere to image boundaries. However, they are often of irregular size and shape. Entropy Rate Superpixels (ERS) [9] solves this by introducing an additional balancing term into the optimized energy.

The algorithms from the third group [10–15] first roughly assign pixels to clusters, and then iteratively refine them. Unlike the algorithms of the first two groups they use regional information, such as pixel color and intensity, directly. The mean shift [11] and watershed [14] follow this strategy. However, they do not produce compact and regular superpixels. SLIC [10] is a more efficient algorithm based on k-means clustering. TurboPixels [13] is a geometric-flow based algorithm, and the approach of Veksler and Boykov [12] is based on graph cuts.

The three groups of algorithms have distinct properties and are applied in different scenarios. Divisive and agglomerative algorithms produce a hierarchy of segments and can use information about pixel similarity in the form of probabilities of boundaries more efficiently. Conversely, the algorithms from the discriminative group are more efficient in using regional information. It is important to note, that divisive and discriminative algorithms are less sensitive to noise, while agglomerative and discriminative algorithms are faster.

3. The proposed algorithm

In this work we propose a new superpixel algorithm from the divisive group that uses the ideas similar to [7,16]. Image is represented by a graph where edges connect neighboring pixels and determine their similarity. Each image segment is recursively partitioned into two subsegments by computing the minimum graph cut. The graph edges are normalized to make the superpixels adhere to image boundaries. Additionally, regularization is applied to make them compact and homogeneous. The algorithm provides direct control over the number of segments and produces compact superpixels of regular size and shape.

At every step the largest 4-connected input segment (a segment is 4-connected, if for each of its pixels there is a pixel at the top, bottom, right or left, which is part of the segment) is partitioned into two 4-connected output segments. The proposed algorithm creates several competing partitions and selects the best of them. Each partition is produced in two stages: terminal placement and segment partitioning by optimization of the objective function.

Let $\mathbf{I}$ be the image vector, and $\mathbf{L}$ – the corresponding vector of pixel labels. The image is represented by a graph $G = (V, E)$, with vertices V denoting the pixels and the edge weights denoting their pairwise similarity. The graph defines a global energy function, which is minimized with respect to the labels $\mathbf{L}$:

$$E(\mathbf{L}) = \lambda \sum_{v_i \in V} \Phi(l_i|v_i) + \sum_{(v_i,v_j)\in E} \Psi(l_i, l_j|v_i, v_j), \quad (1)$$

where λ is a constant, and Φ and Ψ can be defined as follows:

$$\Phi(l_i|v_i) = -\log p(l_i|v_i, \mathbf{I}), \quad (2)$$

$$\Psi(l_i, l_j|v_i, v_j) = -\log p(l_i \neq l_j|v_i, v_j, \mathbf{I})[l_i \neq l_j]. \quad (3)$$

$p(l_i|v_i, \mathbf{I})$ is the probability of label l_i for pixel v_i, $p(l_i \neq l_j|v_i, v_j, \mathbf{I})$ – probability that pixels v_i and v_j have different labels. The choice of Ψ is limited to some extent by the available optimization algorithms.

In the following discussion we will focus on the case when there are two different types of labels (0,1) corresponding to the two subsegments produced during segment partitioning.

Our approach is based on the following ideas:

1. In [7] only a group of segments can be partitioned in one step due to the lattice structure requirement. We developed a new approach that can split the segments individually. It is based on a new strategy for choosing the terminal connections that does not have the limitations of [7].
2. We find the optimal segment partitions using the minimum graph cut algorithm. A special normalization of graph edges is used to move the superpixel boundaries closer to the boundaries of the objects in the image.
3. The optimized energy includes a regularization term that helps to control the trade-off between the segment compactness and the cut optimality.
4. To provide for the homogeneity of segments, the optimized energy includes a term that uses pixel color and intensity.
5. To find $\Psi(l_i, l_j|v_i, v_j)$ using (3) we use a simple classifier (a neural network) that estimates the probabilities of borders between pixels.

3.1. Terminal placement

At the terminal placement stage we select several source-sink pairs. They are used at the second stage to produce several competing segment partitions. We select the best one of them using the following optimality measure:

$$mean\_cut = \frac{\sum_{v_i,v_j\in E} \Psi(l_i, l_j|v_i, v_j)[l_i \neq l_j]}{\sum_{v_i,v_j\in E}[l_i \neq l_j]}. \quad (4)$$

The best partition has the lowest mean_cut.

For each pair, each terminal (the source and the sink) is connected by an infinite capacity edge to a single pixel vertex. In the following discussion we also refer to these pixels as terminals. To generate the first pair of terminals, we select the two farthest pixels on the segment (Fig. 1a). For the remaining pairs both terminals are chosen uniformly at random from the pixels lying on the contour. Thus, the positions of the terminals are selected using the segment shape only, and are independent of the pixel values.

3.2. Segment partitioning

At this stage we split segments using the terminals found on the previous stage. To improve boundary adherence, edge values are normalized by the width of the segment at the corresponding locations. That is, the two selected terminals (t_1, t_2) divide the contour in two groups of vertices C^1 and C^2 (Fig. 1b). The segment width in vertex $v \in V$ is defined as follows: