Segmentation by Example

Sameer Agarwal and Serge Belongie
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093, USA
{*sagarwal,sjb*}*@cs.ucsd.edu*

Abstract

We describe an algorithm for segmenting a novel image based on the available segmentation of another image. The algorithm consists of two stages. In the first stage we construct a locally connected graph to represent the novel image. This graph is obtained by inheriting local connectivity between pixels based on the similarity of small neighborhoods in the two images. In the second stage a graph parititioning algorithm is used to partition the graph and recover the resulting segments in the image. We present experimental results on synthetic and real images and conclude with a discussion of the strengths and limitations of the method.

1 Introduction

One of the dominant views in the study of perceptual grouping comes from the Gestalt school of thought which proposes a set of cues that aid and direct the visual system in organizing the visual field. This process can be divided into three stages: feature extraction, cue integration and grouping. Examples of features include brightness, local color histograms, oriented filter responses, etc. In the cue integration stage, similarities between the features are computed and combined to produce affinities between all pairs of pixels. Finally, the set of pairwise affinities are fed to a grouping engine which attempts to partition the pixels in an optimal manner, e.g. by maximizing within-group affinity and minimizing between-group affinity.

While significant progress has been made in feature extraction and grouping, the problem of cue combination remains unsolved[1]. What makes cue integration difficult is that the cues very often contradict one another, and there is no single correct solution. At one level, there are the obvious ambiguities, e.g. whether polka dots should be grouped as a region or as individual elements. At another level, however, are the less obvious problems, e.g. a texture cue acting on a brightness boundary to produce a spurious long, thin group [8, 4].

One general approach to this problem is that of precomputing several cues in parallel followed by solving for optimal combination weights or "cue gatings." Poggio et al.

[1] As Palmer notes in [7], "the visual system clearly integrates over many grouping factors, but we do not yet understand how it does so."

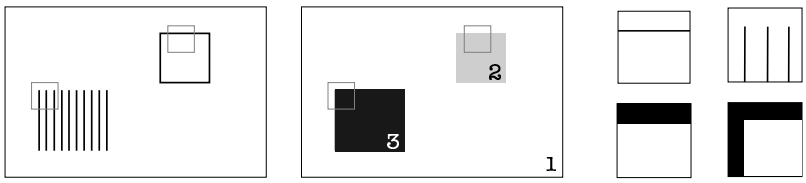

Figure 1: Illustration of the shred representation: (a) example image shown with two sample image shreds outlined in gray, (b) partition image reresenting the hand-labelled segmentation, (c) zoomed-in shred pairs. Each shred pair consists of an image shred, shown at top, and its associated segment shred, shown at bottom, which indicates the pixels belonging to the same segment as the pixel at the center.

[9] present a solution based on a Markov Random Field with line processes driven by brightness edges. Malik et al. [4] demonstrate the use of a number of hand designed rules for cue gating based on local measures such as "texturedness." Meilă and Shi [5] adopt a learning framework based on hand labelled segmentations in which cue weightings are tuned by gradient descent.

The problem that arises in such approaches is that the predefined cues are imperfect and can lead to unrecoverable segmentation errors downstream. This is not to say that certain ad hoc rules are totally ineffective – to be certain, what success has been attained thus far in automatic image segmentation is due in a large part to heuristics based on ecological statistics that are "right most of the time." In this paper, however, we depart from this general strategy and explore an approach that makes active use of hand-labelled segmentations of example images in a non-parametric fashion. Using a graph-theoretic framework, we show how local connections between pixels in novel images can be "inherited" from labelled examples based on a simple measure of similarity of image neighborhoods between the novel and example images. We then show how this locally connected graph can be partitioned to obtain a segmentation of the image.

The organization of this paper is as follows. In Section 2 we discuss our approach to segmentation by example, with the two key stages of matching and assembly. Experimental results appear in Section 3, followed by a discussion on related work in Section 4. Finally we conclude and discuss open issues in Section 5.

2 Our Approach

Our algorithm is based on assembling a segmentation of an image by matching parts of a novel image (the test image) to parts of a collection of pre-segmented images (the training images). We refer to these parts as shreds. Given an image and its segmentation[2], a sliding window of radius r is used to break the image and its segmentation into a collection of paired shreds consisting of an image shred and a segment shred, denoted by the pair $[I_i, S_i]$ for the i^{th} pixel. Each pixel is identified with the shred pair centered on it. The segment shred, which originally contains integer segment labels to indicate the segment membership of each pixel, is simplified into a $\{0/1\}$ map indicating whether a pixel belongs to the the same segment as the center pixel or not; see Figure 1. In the case where we only want

[2]The segmentation is represented by a *partition image*, which is an integer-valued image indicating to which segment each pixel belongs.

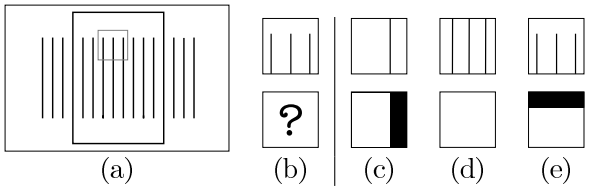

Figure 2: Illustration of shred matching: (a) test image with sample shred outlined in gray, (b) zoomed-in image shred I'_i; the question mark denotes its corresponding segment shred S'_i that we wish to find, (c-e) a few representative image/segment shred pairs from the training image. The best matching image shred in this case is (e).

to separate the foreground from the background, this representation reduces to the one used by Borenstein and Ullman [1]. However in the more general case of an image containing multiple segments, we will show that this representation can also be considered as a connectivity pattern in a graph.

The process of segmentation for a test image subsequently breaks down into two stages: matching and assembly.

2.1 Matching

In the matching stage, for every test image shred I'_i we seek a segment shred S'_i that best segments it. This is done by finding a shred pair $[I_j, S_j]$ that minimizes a distance function $d(\cdot,\cdot)$ over all shred pairs in the training dataset and the image shred I'_i. Naively minimizing the distance $d(I'_i, I_j)$ will not work since this would require that all possible adjacent surfaces that are present in the novel image be present in the training dataset. This is an unreasonable requirement, and it has a simple solution. Since the only portion of a shred we are interested in is the set of pixels that will belong to the same segment as the pixel at the center, we perform the minimization over the pixels labelled 1 in the corresponding segment shred S_j. This is done by using the segment shred as a mask:

$$S'_i = \arg\min_{S_j} d(I'_i \circ S_j, I_j \circ S_j) \tag{1}$$

Here $\circ$ denotes pixelwise multiplication. This process is illustrated in Figure 2.

In our experiments we use a simple sum-of-squared-differences (SSD) distance function normalized by mask area:

$$d(I'_i \circ S_j, I_j \circ S_j) = \frac{\sum_{x,y} S_j(x,y)\left[I'_i(x,y) - I_j(x,y)\right]^2}{\left[\sum_{x,y} S_j(x,y)\right]^{\alpha}} \tag{2}$$

The real valued parameter $\alpha > 1$ is used to bias the search in favor of matches with larger mask area. We set $\alpha = 1.7$ in all of our experiments.

2.2 Assembly

Once we have associated a segment shred with each image shred in the test image, we need to put these pieces together to come up with a segmentation. If we now consider the pixels in the image as vertices of a graph, the segment shred S'_i can be interpreted as local connectivity information for pixel i as illustrated in Figure 3. Using this information we can now construct a sparse directed graph G_d with the the

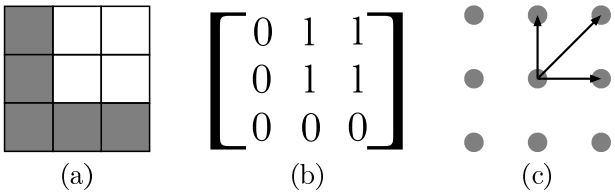

Figure 3: Illustration of local graph connectivity: (a) example of 3×3 segment shred, (b) representation as a 3×3 matrix, (c) induced directed graph.

property that there is an edge from vertex i to vertex j iff the pixel corresponding to j has value 1 in the segment shred S'_i. This connectivity is asymmetric; we reduce the graph G_d to an undirected graph G by keeping only the edges between pairs of pixels that have a pair of edges running both ways.

In an ideal world this would be sufficent, since if we knew the correct connectivity pattern for each pixel, the graph G would contain a connected component corresponding to each segment in the image and our problem would be solved. However, this is not so. Since there are spurious edges present in the graph (due to bad matches, or neighborhoods not present in the training data precisely) we end up with a graph that has a number of distinct clumps of vertices loosely connected to each other. Hence we need to partition this graph into components in a manner that captures as much of the connectivity of the graph as possible. The problem reduces to one of graph partitioning, a task for which several algorithms are available. In our experiments we use Normalized Cuts [12].

2.2.1 Normalized Cut

The minimum Normalized Cut is a vertex partitioning of a graph $G(V, E)$ into two disjoint subsets that maximizes the separation between the two sets of vertices while maximizing the connectivity within the sets. Formally, we want to find $A \cup B = V, A \cap B = \emptyset$ to minimize

$$\mathrm{NCut}(A, B) \doteq \mathrm{Cut}(A, B) \left(\frac{1}{\mathrm{Vol}(A)} + \frac{1}{\mathrm{Vol}(B)} \right) \tag{3}$$

where

$$\mathrm{Cut}(A, B) \doteq \sum_{u \in A, v \in B} w(u, v), \qquad \mathrm{Vol}(A) \doteq \sum_{u \in A, v \in V} w(u, v)$$

and $w(u, v)$ denotes the weight on the edge connecting the vertices u and v. In our case it is 1 if u and v are connected else it is 0.

Solving for the exact minimum of the above objective function is an NP-Hard problem. The second eigenvector of the generalized eigenvalue problem $Wx = \lambda Dx$ approximates the optimal partition, where W is the connectivity matrix for the graph and D is a diagonal matrix of row sums of W: $D_{uu} = \sum_{v \in V} w(u, v)$.

The above formulation can be extended to a k-partitioning of the graph by using additional eigenvectors [11, 4, 6]. We do so by stacking the 2^{nd} to the k^{th} eigenvectors columnwise, normalizing the rows of the resulting matrix, and performing k-means clustering on them. The only parameter that the algorithm needs is the number of partitions. Despite the large number of vertices, the eigenvalue problem

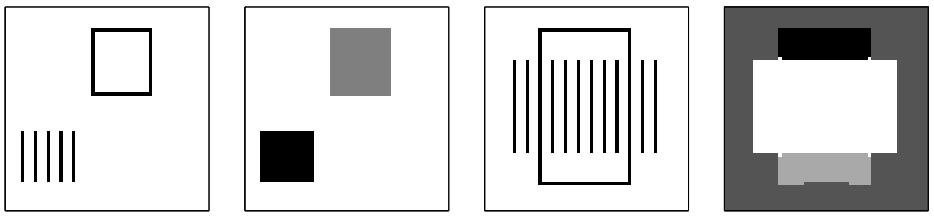

Figure 4: Illustration of segmentation by example: (a) training image, (b) hand-labelled segmentation containing 3 groups, (c) test image (the Kanizsa rectangle), (d) computed segmentation, using $r = 5$ and $k = 4$. The image dimensions are 64×64.

is not computationally hard, since the matrix W is sparse, and efficient methods for calculating eigenvectors of sparse matrices are readily available.

We do not deal with the problem of choosing k in this paper. The problem can attacked by looking at the eigengap in the spectrum of $D^{-1}W$. A large eigengap between the k^{th} and the $k + 1^{st}$ eigenvalues is indicative of a good clustering into k clusters.

3 Experiments

We demonstrate the performance of our algorithm on two example images. The first experiment was done on a pair of synthetic images. Figure 4 shows the results. The training image contains an instance of a closed contour and a region of pinstripe texture. The test image (the Kanizsa rectangle) is composed of the same basic content in a different spatial arrangement. The computed segmentation of the testing image is consistent with the segmentation of the training image.

The second experiment, shown in Figure 5, uses a pair of photographs of a cell phone and a piece of wire. The first image was hand-segmented and used as a training example. The algorithm correctly interprets the black wire as a contour boundary and partitioned the test image correctly.

In both examples, the window size was set to 11×11 ($r = 5$). The number of components k for clustering was set to 4 and 7, respectively.

4 Related Work

Most closely related to our approach is the recent work of Borenstein and Ullman [1] which uses hand-labelled image fragments to stitch together figure/ground maps for images of horses. Their work, however, does not generalize to more than two segments (foreground/background) and they use a greedy algorithm to resolve conflicts between the labels in overlapping fragments.

The inspiration for our approach comes from the work on texture synthesis and transfer by Efros and Leung [2] and Hertzmann et al. [3] In this regard we are performing non-parametric sampling for the segment label in the neighborhood of each pixel. Since the segment labels are only meaningful in a relative (as opposed to absolute) sense, we require an additional step to resolve all of the local neighborhood relationships into a globally consistent segmentation.

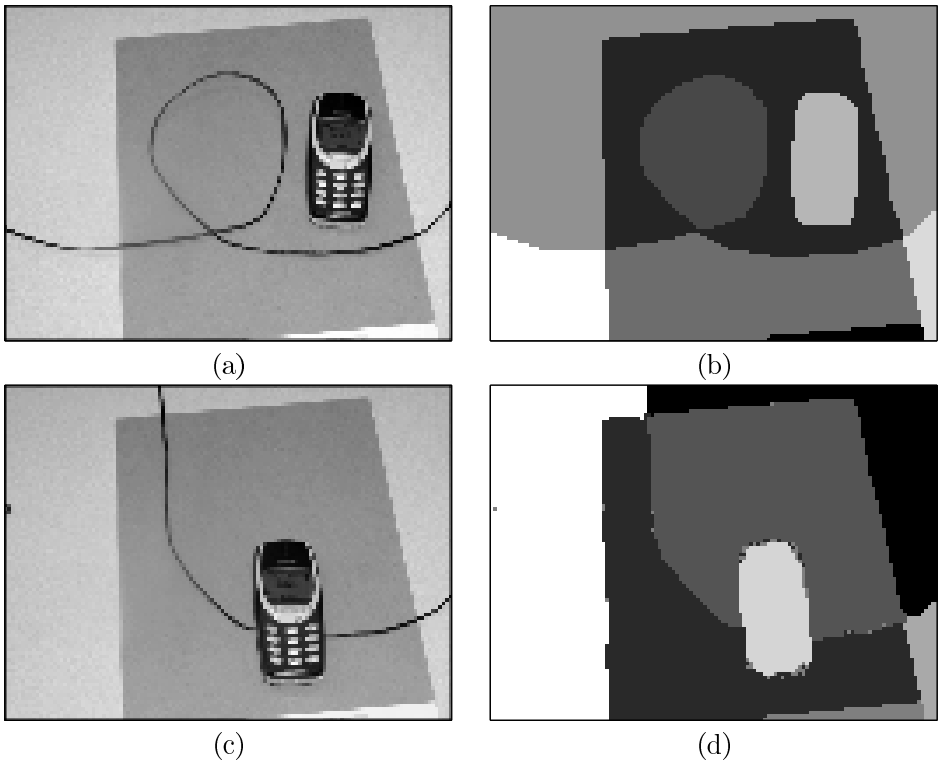

Figure 5: Segmentation by example for a cell phone and a piece of wire: (a) training image, (b) hand-labelled segmentation containing 8 groups, (c) test image, (d) computed segmentation, using $r = 5$ and $k = 7$. The image dimensions are 96×128.

5 Discussion

5.1 Choice of Parameters

The system has a number of parameters; the first and the most important parameter is the set of training images that are used for segmenting a new image. Even though we only used one image as a training image, it is easy to see that the system can use any number of training images simultaneously. The structure of the training set expresses a bias for particular interpretations of a novel image. Figure 6 demonstrates the result of using vertical and horizontal brightness edges to segment an image with a serrated boundary. The system tries to approximate the jagged edges without success. One can argue that various kinds of invariances (e.g. rotation or affine) be incorporated into the matching stage. An alternative view is that the search be guided by the content present in the training data.

The second most important parameter in the system is the choice of the distance function $d(\cdot, \cdot)$. In this system we used a very simples distance measure – sum of squared differences – though more sophisticated distance measures will be needed for capturing the similarity between patches of stochastic texture, e.g. histograms of filterbank outputs [10, 4].

The value of the parameter r decides the scale sensitivity of the method. The smaller the scale of matching, the less sensitive it is to large scale structure in the image. For

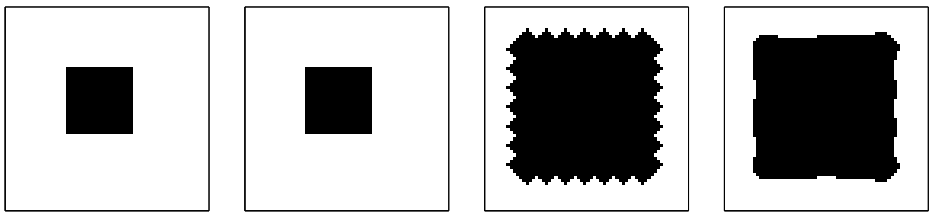

Figure 6: Illustration of segmentation by example for a serrated square, illustrating the effect of bias in the training image: (a) training image (of size 64×64), (b) hand-labelled segmentation, (c) test image, (d) computed segmentation, using $r = 3$ and $k = 2$. In this case, the training image contains only straight edges, and the extrapolated segmentation is not able to capture the details of the boundary.

example, if the patch radius is too small while segmenting the Kanizsa rectangle, it will cause the texture to break up into individual line segments. The choice of an optimal scale for matching is a difficult decision, but it need not necessarily be made by the user. Since we have segmented data available to us a priori, one can in principle analyze each of the segments and determine (by performing a self match) the smallest value of r required for recovering the segmentation. The choice of a scale parameter for images that require different scales of interpretation in different parts of the image remains an open question.

5.2 Future Work

The system as it stands is extremely inefficent and slow due to the nearest neighbor search. The nearest neighbor search is one of the many ways in which a segment shred can be associated with an image shred. We chose to use a nearest neighbor classifier for its simplicity. There are a number of ways in which this search can be made more efficient.

Borenstein and Ullman [1] propose a statistical method for reducing the set of shreds to a much smaller number based on their informativeness with respect to their associated segmentations. This can result in sizeable performance improvements since the image shreds contain a large number of duplicates and redundant examples.

There are a number of domains where additional information is available that can aid in narrowing down the search significantly. Two such domains are video segmentation and stereo. Given a segmentation of a few key frames, an estimate of optical flow from one frame to another can be used for transferring the segmentation from one frame to the next. Similarly, given the segmentation of one view and an estimate of the epipolar geometry, the search can be constrained along epipolar lines.

5.3 Limitations of the method

This work represents some initial steps in example based segmentation and a number of issues remain open. By itself, the method we have presented does not solve the problem of perceptual grouping. What we have proposed is a way of using labelled data to drive low-level image segmentation. The Kanizsa triangle shown in Figure 7 illustrates the key limitation of our method. Purely local matching cannot recover the illusory triangle. Therefore to handle figures such as this properly, one must

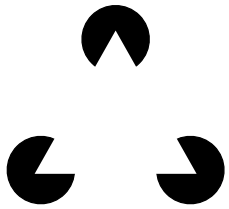

Figure 7: Kanizsa triangle: a purely local image-based approach cannot segment this figure properly; higher level knowledge is necessary.

find a way to incorporate high level knowledge into the system.

Acknowledgements

The authors would like to thank Eran Borenstein, Charless Fowlkes, Alyosha Efros, and Kristin Branson for helpful discussions.

References

[1] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *Proc. 7th Europ. Conf. Comput. Vision*, May 2002.

[2] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proc. 7th Int'l. Conf. Computer Vision*, 1999.

[3] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogies. In *SIGGRAPH*, 2001.

[4] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *Int'l. Journal of Computer Vision*, 43(1):7–27, June 2001.

[5] M. Meilă and J. Shi. Learning segmentation with random walk. In *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, pages 873–879, 2001.

[6] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference*, 2002.

[7] S. E. Palmer. *Vision Science*. MIT Press, 1999.

[8] A. Perry and D. Lowe. Segmentation of textured images. In *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, pages 319–323, 1989.

[9] T. Poggio, E. Gamble, and J. Little. Parallel integration of vision modules. *Science*, 242:436–440, 1988.

[10] J. Puzicha, T. Hofmann, and J. Buhmann. Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. In *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, pages 267–72, 1997.

[11] G.L. Scott and H.C. Longuet-Higgins. Feature grouping by 'relocalisation' of eigenvectors of the proximity matrix. In *Proc. British Machine Vision Conference, Oxford, UK*, pages 103–108, 1990.

[12] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.