Stochastic Clustering and its Application to Image Segmentation

Yoram Gdalyahu   Daphna Weinshall   Michael Werman
Institute of Computer Science
The Hebrew University, 91904 Jerusalem, Israel
e-mail: {yoram,daphna,werman}@cs.huji.ac.il

Abstract

We present a stochastic clustering algorithm which uses pairwise similarity of elements. The problem is viewed as a graph partitioning problem, where nodes represent data elements and the weights of the edges represent pairwise similarities. We generate samples of cuts in this graph, by using David Karger's contraction algorithm, and compute an "average" cut which is our solution to the clustering problem. The stochastic nature of our method makes it robust against noise, including accidental edges and small spurious clusters. The complexity of our algorithm is very low: $O(N \log^2 N)$ for a fixed accuracy level. In addition, and without additional computational cost, our algorithm provides a hierarchy of nested partitions. We demonstrate the superiority of our method for image segmentation on a few synthetic and real images, B&W and color, where some other recently proposed methods (such as normalized-cut) fail or perform less well.

1 Introduction

A key problem in image analysis, and most high level computer vision algorithms, is the prohibitively large amount of raw data that is given as input. Algorithms that use low level image features, such as pixels and edge elements, need to deal with a large number of possible interpretations due to the sheer amount of data and the inherent high levels of noise. Thus for most applications, including feature extraction and object recognition, it is necessary to divide the raw image data into parts that can be processed separately. This problem is called image segmentation when applied at the pixel level, or perceptual grouping when applied to features such as edge elements. In the analysis of image databases, a related problem appears. Since we are typically dealing with a large number of images, it is often necessary to divide them into subsets that can be processed separately during image retrieval.

Data partitioning is needed in many application domains, including image analysis and computational biology; this typically ill-defined problem involves two distinct steps: (i) determine some ad-hoc criterion for the desirable partitions; (ii) design efficient algorithms to compute these partitions. Clearly problem (i) is task dependent and extremely elusive; the partitioning criterion may reflect a measure of point scatter (if the data points are embedded in a normed space) or a measure of graph connectivity (if only pairwise distances are given). Many earlier image segmentation and grouping algorithms suffer from one (or both) of these problems: either the partitioning criterion fails in many situations, or no effective partitioning algorithm exists to compute a certain desirable criterion.

A problem related to data partioning is grouping, the aggregation of low level elements (such as edgels) into coherent groups. Perceptual grouping algorithms address two separate issues: (i) how to compute pairwise similarity between elements; (ii) how to aggregate these elements taking into account local and global principles such as good continuation or symmetry. Step (ii) is equivalent to constrained data partitioning, where the ad-hoc partitioning criterion is designed to match human perceptual grouping. A review and comparison of perceptual grouping algorithms can be found in [19].

We focus here on clustering, or the unsupervised partitining of data into parts. In Section 1.1 we review a few clustering approaches which were used recently in computer vision and image processing applications. Our main contribution is a new stochastic clustering algorithm which unifies many of the different clustering approaches into one (Section 2). Our algorithm is very efficient, relatively easy to implement, and more robust due to its stochastic nature.

More specifically, we start from pairwise similarity measurements between datapoints. This defines a weighted graph whose nodes are the datapoints, and whose edges are assigned weights according to the pairwise similarity of the datapoints. In images the graph is typically sparse since only neighboring elements may be similar. The source of pairwise similarity measurements can be local brightness similarity or color histogram similarity. Our task is to partition the set of nodes into *meaningful* parts (clusters); in other words, to find a cut in the graph.

Our partitioning criteria is based on the notion of minimal cut: we seek to cut the graph along lines of least similarity, thus maximizing the similarity within each cluster while minimizing inter-cluster similarity. Minimal cuts and their relatives (such as normalized cuts and average cuts) were used for data partitioning in a few recent computer vision applications ([20, 16, 7, 2], see also Section 1.1). Our approach is different: we compute the probability for each edge to be a crossing edge, under a probability distribution which is imposed over cuts. This is done at various scale levels (where scale is the number of separate clusters generated by the cut), and then we remove edges with high such probability. Note that we compute the probability of an edge to be in a cut, rather than the minimal cut itself. Our sampling method is based on the work of Karger [9], who developed an efficient stochastic algorithm to compute minimal cuts.

Computing edge-wise probabilities is the key to the efficiency of our method, since exact min cut computation is expensive. The stochastic nature of this computation is the key to the robustness of our method, since the average probability of edges belonging to isolated minimal cuts tends to be low. Moreover, while other methods may switch abruptly from one solution to another as parameters of the data vary smoothly (this is rather characteristic of spectral clustering methods), our results change continuously with changes in the data. Finally, our method produces a hierarchical clustering of the data: there is no need to specify the number of clusters in advance since we generate all paritionings, while automatically identifying the levels which represent meaningful paritioning of the data. We demonstrate all these properties in Section 3, where we test our method on real image segmentation.

1.1 Review of graph based clustering methods

We review here methods which use only pairwise relations of similarity or dissimilarity between items. The pairwise information can be represented by a weighted (and perhaps incomplete) graph $G(V, E)$: the nodes V represent data items, and the positive weight w_{ij} of an edge (i, j) represent the amount of similarity or dissimilarity between items i and j.

One pairwise clustering strategy uses a similarity threshold θ to remove edges with weight less than θ, and identifies the connected components that remain as clusters. A transformation of weights may precede the thresholding, to reflect transitive similarity relations. One recent example is the physically motivated transformation proposed in [1], which uses a granular magnet model and replaces weights by "spin correlations".

A second pairwise clustering strategy is used by agglomerative algorithms [3, 8], which start from the trivial partition of N points into N clusters of size one, and continue by subsequently merging pairs of clusters. At every step the two clusters which are most similar are merged together, until the similarity of the closest clusters is lower than some threshold. Different similarity measures between clusters distinguish between different agglomerative algorithms. In particular, the single linkage algorithm defines the similarity between clusters as the maximal similarity between two of their members, and the complete linkage algorithm uses the minimal value.

A third pairwise clustering strategy uses the notion of cuts in a graph. A cut (A, B) in a graph $G(V, E)$ is a partition of V into two disjoint sets A and B. The capacity of the cut is the sum of weights of all edges that cross the cut, namely: $c(A,B) = \sum_{i\in A, j\in B} w_{ij}$. Among all the cuts that separate two marked vertices, the *minimal cut* is the one which has minimal capacity. The minimal cut clustering algorithm [20] divides the graph into components using a cascade of minimal cuts.[1] In an interesting variation, special graphs are designed in [7, 2] such that the values of cuts in these graphs correspond to some desirable cost function, and so the minimal cut solves an optimization problem.

In a related approach, the normalized cut algorithm [16] uses the association of A (the sum of weights incident on A) and the association of B to normalize the cut capacity $c(A,B)$. In contrast with the polynomial min-cut problem, finding the minimal normalized cut is NP-hard, but with certain approximations it reduces to a generalized eigenvalue problem [16]. The observation that cut-based graph partitioning can be obtained with spectral graph analysis is computationally important; it also ties nicely together the group of graph-paritioning methods with another class of methods, which use graph spectral methods for clustering. For example, the first eigenvector of the similarity matrix is used for image segmentation in [15], and for figure ground separation in [13]; the use of the k largest eigenvectors is described in [17].

Our current work provides a common framework to all the methods described above. Specifically, our new algorithm may be viewed as a randomized version of an agglomerative clustering procedure, while at the same time it is a stochastic minimal cut algorithm. It is also strongly related to the physically motivated granular magnet model algorithm described in [1], which preceded and inspired our work. Our method is unique in its stochastic nature while maintaining low complexity, which ensures robustness against noise and pathological configurations:

(i) A *minimal cut* algorithm is intuitively reasonable since it optimizes so that as much of the similarity weight remains within the parts of the clusters, and as little as possible is "wasted" between the clusters. However, it tends to fail when there is no clean separation into 2 parts, or when there are many small spurious parts due, e.g., to noise. Our stochastic approach avoids these problems and behaves more robustly thanks to the use of average edge probabilities.

(ii) The *single linkage* algorithm deals well with chained data, where items in a cluster are connected by transitive relations. Unfortunately the deterministic construction of chains can be harmful in the presence of noise, where a few points can "bridge" between two large clusters and merge them together. Our algorithm inherits the ability to cluster chained data; at the same time it is robust against such noisy bridges as long as the probability to select all the edges in the bridge remains small.

(iii) The stochastic nature of our method give us robustness: as points are added and moved around, average edge probabilities track these changes smoothly and therefore the final clustering does not change drastically. On the other hand, by just adding or removing a point, deterministic algorithms based on the minimal or normalized cut, single linkage, or spectral methods, may jump from one solution to another very different one.

2 The algorithm

2.1 Notations and definitions

Our clustering algorithm uses pairwise similarities, which are represented by a weighted graph $G(V,E)$: the nodes V represent data items, and the positive weight w_{ij} of an edge (i,j) represent the amount of similarity between items i and j. The graph $G(V,E)$ may be incomplete, due to missing data or due to edge dilution whose purpose is to increase efficiency. The weights w_{ij} may violate metric properties, and in general they may reflect either similarity or dissimilarity values. In the current work, however, we assume that the weights reflect symmetric similarity relations (hence $w_{ij}=w_{ji}$, and $w_{ij}=0$ for i and j that are completely dissimilar). We do not assume that the similarity weights obey the triangular inequality, and self similarities w_{ii} are not defined.

[1]The reader who is familiar with flow theory may notice that this algorithm also belongs to the first category above, as it is equivalent to weight transformation followed by thresholding. The weight transformation replaces w_{ij} by the maximal flow between i and j.

A cut (V_1, V_2) in a graph $G(V, E)$ is a partition of V into two disjoint sets V_1 and V_2. The capacity of the cut is the sum of the weights of all edges that cross the cut, namely: $c(V_1, V_2) = \sum_{i \in V_1, j \in V_2} w_{ij}$. The *minimal cut* is the one which has minimal capacity. We use the term "cut" also for the generalized case of multi-way cuts. A partition of V into r disjoint sets $(V_1, \ldots, V_r)$ is called r-way cut, and in accordance its capacity is defined as $\sum_{i \in V_\alpha, j \in V_\beta, \alpha \neq \beta} w_{ij}$. Every one of the r components may be referred to as a "side" of the cut.

Let the nodes which belong to each set V_α ($\alpha = 1 \ldots r$) be grouped together into one *meta-node*, and discard all the edges which form self loops on meta-nodes (namely, discard the inner edges of each component, which connect nodes belonging to the same component). The graph which is obtained has exactly r meta-nodes, and it is a multi-graph since meta-nodes may be connected to each other by more than one edge. Actually, if G was a complete graph, then the number of edges connecting the meta-nodes which represent the components V_α and V_β is exactly $|V_\alpha||V_\beta|$.

The grouping procedure described above yields a *contracted graph* which has r meta-nodes, denoted G'_r. Note that this notation does not characterize the contracted graph, since there are many ways to group the nodes of G into r disjoint sets. However, any contracted graph G'_r represents an r-way cut in the original graph. The edges of G'_r are the edges which cross the corresponding r-way cut in the original graph.

2.2 Overview of the algorithm

We now provide a simple outline of the algorithm, ignoring considerations of space and time complexity.

For a given value of r ($r = 1 \ldots N$) our algorithm generates a sample of M possible r-way cuts, and computes the sample average. The following pseudo code counts, for every pair of nodes $i, j \in V$ and for every integer r between 1 and N, the number of r-way cuts (out of M) in which the two nodes are at the same cut side. These counts are divided by M to estimate for every two nodes the probability that they are at the same side of a random r-way cut. In other words, if the edge (i, j) exists in E, we estimate the probability that (i, j) is not a crossing edge of a random r-way cut.

```
procedure STAGE-1:
input:  graph G(V,E) with N nodes.
output: 3D array p of probabilities.
initialize counters:  s^r_ij <- 0 for i,j,r = 1...N.
for m = 1...M:
    G'_n <- G(V,E)
    for r = (N-1)...1:
        G'_r <- CONTRACT(G'_{r+1})
        for i = 1...N:
            for j = 1...N:
                if i and j belong to the same meta-node of G'_r, then
                   s^r_ij <- s^r_ij+1
                end-if
            end-loop
        end-loop
    end-loop
end-loop
normalize empirical probabilities:  p^r_ij <- s^r_ij/M for i,j,r = 1...N.
return p.
```

In this pseudo code the procedure `CONTRACT` generates the r-way cut G'_r from the previously generated cut G'_{r+1}. The procedure `CONTRACT` selects two meta-nodes of G'_{r+1} and merges them into one meta-node of G'_r, while discarding the edges which were previously connecting these two meta-nodes. The selection of the nodes to be unified is stochastic: an edge (i, j) of G'_{r+1} is selected for contraction with probability proportional to its weight w_{ij}. Then, the two meta-nodes which are connected by the selected edge are unified.

The following pseudo code uses a simple (non efficient) way to select an edge (i,j) with probability w_{ij}.

```
procedure CONTRACT(G'_k)
input:  graph G'_k with k meta-nodes.
output:  graph G'_{k-1} with k-1 meta-nodes.
let ℓ be an index to the set of edges of G'_k  (ℓ = 1...L).
S ← Σ_{ℓ=1}^{L} w_ℓ.
x ← random number in (0,S].
find minimal ℓ_0 such that Σ_{ℓ=1}^{ℓ_0} >= x.
find which two meta-nodes A_α and A_β of G'_k are connected by the edge ℓ_0.
merge A_α and A_β.
remove self loops resulting from edges which were previously connecting A_α and A_β.
return the resulting graph.
```

An efficient implementation of the contraction algorithm described above is a corner stone in our method. It defines the sample of M cuts which is used to estimate the probabilities p^r_{ij}. Using the contraction algorithm, the probability of every r-way cut to be included in the sample decreases with increasing capacity of the cut. (In fact, [9] proves that the minimal cut can be found using this sampling method in polynomial time, even though the number of cuts is exponential.) Hence, the process induces a probability distribution over cuts, which decays with increasing capacity, and under this cut distribution we measure p^r_{ij}— the average over the sample of M cuts of the probability that nodes i and j are at the same side of a random r-way cut.

For every integer r between 1 and N we define the *typical cut* $(A_1, A_2, \ldots, A_s)$ as the partition of G into s components, such that for every $i \in A_\alpha$, $j \in A_\beta$ $(\alpha \neq \beta,\ \alpha, \beta = 1 \ldots s)$ we have $p^r_{ij}<0.5$. To find the typical cut for every integer r between 1 and N we first remove all the edges whose transformed weight $p^r_{ij}$is smaller than 0.5, and we then compute the connected components in the remaining graph. Note that the number of parts, s, in the typical cut can be different from r.

The N typical cuts corresponding to $r = 1 \ldots N$ are the candidate solutions to our clustering problem. Although this is an extremely small number compared with the 2^N possible partitions, we still need to select only a few interesting solutions out of the N candidates. The question that remains is to define and choose "good" values of r, for which a "meaningful" clustering is obtained as part of a hierarchy of a few selected partitions.

We define the following function of the partition at each r:

$$T(r) = \frac{2}{N(N-1)} \sum_{i>j} N_i N_j$$

where $N_k = |A_k|$ denotes the number of elements in the k-th cluster. $T(r)$, therefore, measures how many edges of the complete graph cross over between different clusters in the r-partition, relative to the total number of edges in the complete graph.

Partitions which correspond to subsequent r values are typically very similar to each other, or even identical, in the sense that only a few nodes (if any) change the component to which they belong. Consequently, $T(r)$ typically shows a very moderate increase. However, abrupt changes in $T(r)$ occur between different hierarchical levels of clustering, when two or more clusters are merged.

We propose to look at changes in the value of $T(r)$ between subsequent r values, and output only those partitions which are associated with a large change in $T(r)$. For the current presentation we set a threshold Δ, and output a solution at level r only if $\Delta T(r) > \Delta$.

```
procedure STAGE-2:
input:  3D array p of the probabilities p^r_ij
output: hierarchy of a few selected partitions.
```

```
for r = 1...N:
    let G(V,E) be a complete weighed graph of N nodes
    and assign weight p^r_ij to each edge (i,j) ∈ E.
    for i = 1...N:
        for j = 1...N:
            if p^r_ij < 0.5
               then E ← E\(i,j)  remove {i,j} from E
            end-if
        end-loop
    end-loop
    find connected components (A_1, A_2, ..., A_s) in G(V,E).
    compute ΔT(r) = T(r) - T(r-1).
    if ΔT(r) > Δ then
       report partition (A_1, A_2, ..., A_s).
    end-if
end-loop
```

It is not only the existence of pronounced peaks in $\Delta T(r)$ that let us find the desired solutions. The crucial thing is that the set of the N candidate typical cuts, that correspond to the N possible r values, *indeed contains* the desired solutions. Whether this is the case or not, it depends on the output of the first stage of the algorithm, where the wisdom of our method lies. The simple heuristic that is applied in the second stage is based on the assumption that a good set of candidates is given.

2.3 Complexity analysis

The efficient implementation of the algorithm is based on two observations. The first one is that the concrete value of p^r_{ij} is never used; we only need to know whether it is smaller or larger than half. The second observation is that the similarity graph is typically sparse, in the sense that the vast majority of its edges have negligible weights. These observations, and when the number of edges is $O(N)$, lead to $O(N \log N)$ complexity of each sample (including N consecutive contractions); the detailed analysis and implementation details are omitted.

The asymptotic dependence of the sample size on the number of nodes can be shown to be $M = O(\frac{\log N}{\epsilon^2})$ for accuracy ϵ. For fixed ϵ, the overall complexity of the algorithm is therefore $O(N(\log N)^2)$. In practice, $\log N$ is rarely much larger than 10, and ϵ can be kept relatively large since the only information we use is whether p^r_{ij} is smaller or larger than half. From our experience with many examples we know that a sample of size 1000 is rarely insufficient, and 200 is almost always as good.

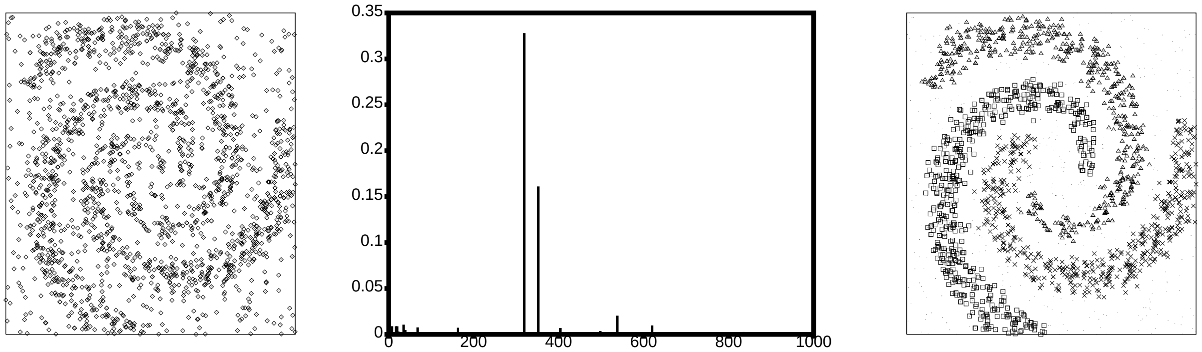
3 Examples and comparisons

3.1 Illustrative synthetic example

To illustrate some features of our method and its relative advantages over other methods, we start with an illustrative synthetic example. We use a point set example in two dimensions which can be easily visualized, see Figure 1(a). The data consists of N=2000 points in $\mathcal{R}^2$ arranged in three dense spiral regions and sparse background.

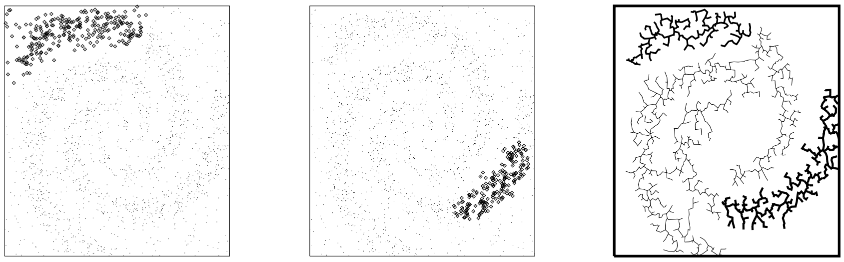
The vectorial nature of the data in this example, namely points in $\mathcal{R}^2$, is used for visualization but is hidden from the clustering algorithm. The information which is made available to the clustering algorithm includes only the matrix of pairwise Euclidean distances d_{ij} between points.

During pre-processing, the Euclidean distances d_{ij} are transformed to similarity weights, which decrease with increasing distances. We use the same functional form as in [1, 13, 16], namely $w_{ij} = \exp(-d_{ij}^2/a^2)$,



(a) (b) (c)

Figure 1: Clustering of points in the Euclidean plane. (a) The 2000 data points; the coordinates of the points are not available to the clustering algorithm, which uses only the matrix of pairwise distances. (b) The graph of $\Delta T(r)$, computed for *every* integer r between 1 and N. Two peaks are clearly observed at r=320 and r=353. (c) The typical cut at r=354. The three large components are indicated by different symbols (crosses, triangles and squares) and the isolated background points are marked by small dots.



(a) (b) (c)

Figure 2: The performance of other algorithms applied to the data of Figure 1. The same preprocessing and exponential transformation from distances to weights was used (see text). (a) The best normalized cut [16] partition. (b) The result obtained by the factorization method [13]. (c) Deterministic single linkage. Unlike our randomized algorithm, the deterministic single linkage algorithm is sensitive to "bridges" that connect large clusters. Here, the procedure is halted manually when 3 large clusters exists, and just before two of them merge together. The desired structure is already irrevocably missing.

where a is the average distance to the n-th nearest neighbor (we used n=10, but the results are not very sensitive to this choice). We then construct a complete graph $G(V, E)$ with N nodes, where node i represents the i-th data point and the weight of edge (i, j) is set equal to w_{ij}.

The constant a reflects some reasonable local scale. If $d_{ij} \gg a$ then w_{ij} is very small and unlikely to get contracted, same as when w_{ij} is unknown. In this case the decision whether points i and j are in the same cluster depends solely on transitive relations, and we may delete the edge $\{i, j\}$ (and all other edges with negligible weights) from our similarity graph $G(V, E)$ in order to increase computation efficiency. In the current example we have discarded edges whose weights were smaller than 0.01 (note that the weights are in $[0, 1]$). This threshold and the number n=10 (which determines a) are the only parameters involved in the pre-processing stage. The number of remaining edges in this example was about 46000.

Having constructed the similarity graph $G(V, E)$, we are ready to apply our clustering algorithm. The first stage is to estimate the pairing probabilities p^r_{ij}. This is done using a sample of M partitions at each r level (see procedure `STAGE-1` in Section 2). Recall that for accuracy ϵ the desired sample size, M, behaves as $M = O(\frac{\log N}{\epsilon^2})$. For most problems it is sufficient to use M=200, but in order to increase the accuracy even further in this illustrative example we have used M=1000. The output of this stage is the three dimensional array p^r_{ij}.

In the second stage of our algorithm we call the procedure `STAGE-2` of Section 2, initially terminating it after it finds a candidate partitions for each r between 1 and N. This candidate partition is the typical cut associated with r. To select between the N candidate partitions procedure `STAGE-2` looks for large changes in the function $T(r)$ defined in Section 2. Figure 1(b) shows the graph of $\Delta T(r) = T(r) - T(r-1)$ as a function of r. It is an impulse graph, illustrating that partitions r which correspond to large changes in $T(r)$ are few and easy to identify. Two obvious peaks appear at r=321 and r=354, and they mark the meaningful levels of data organization in this example. We now continue the call to procedure `STAGE-2`, using the graph of Figure 1(b) to set its parameter Δ to 0.1.

Figure 1(c) shows the partitions reported by procedure `STAGE-2`. There are two peaks in $\Delta T(r)$, corresponding to 3 hierarchical levels of partitions. At r=320 the three spirals form one giant cluster, and the background points form isolated clusters of size one. This is the coarsest level of interest. In the next interesting level, at r=321, the giant cluster splits into two - one part contains a single spiral and the other part contains two spirals. This significant change is signaled out by the large peak at r=321. The next peak, signaling a significant change, occurs at r=354, where the typical cut splits the graph into three large parts, each one containing one spiral (and into many isolated points).

For comparison, we apply to the same data exactly (after identical pre-processing) a few other methods. The results with two spectral methods are shown in Figure 2(b,c); clearly these methods fail here, while our algorithm produces good results.

Our algorithm clearly generates a few good partitions among the N generated partitions, and automatically chooses the best ones. To illustrate that generating good partitions is not trivial, we also tested the deterministic single linkage algorithm, which is a deterministic version of our algorithm. This algorithm gives a single hierarchy of N partitions, while in our method we average over M stochastically generated hierarchies. If we try to select good solutions among the N partitions that are generated by the deterministic single linkage, we find that this hierarchy does not contain the desired solutions. No matter what stopping criterion is used, whether it is a large change of $T(r)$ or the selected weight or anything else, the single linkage algorithm cannot return the desired solutions (see Figure 2(a)).

3.2 Intensity image segmentation

In this section we apply our clustering algorithm to the brightness image segmentation problem. In this problem each pixel of the image is a data item, and the similarity w_{ij} between pixels i and j depends on proximity and intensity values. The goal is to partition the image into regions of coherent brightness.

It is common to assume Gaussian decay for the similarity function [13, 16], namely $w_{ij} = \exp(-d^2_{ij}/a^2) \times \exp(-dI^2_{ij}/b^2)$. The first factor measures similarity by proximity and the second factor measures similarity

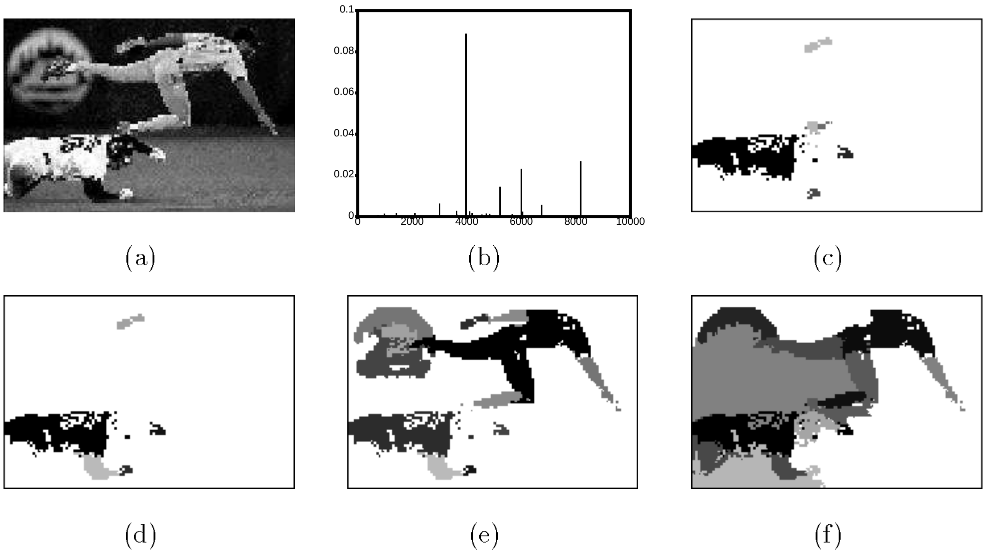


Figure 3: Brightness image segmentation. (a) The original image of size 147 X 221. (b) The graph of $\Delta T(r)$. (c)-(f) The segmentation results which correspond to the four largest peaks of $\Delta T(r)$ at r=3968, 5216, 5998 and 8174. At low r values, most of the image pixels form one giant cluster (and points on object boundaries are left isolated). The large peak at r=3968 marks the splitting of one player body from this cluster, as shown in (c). The smaller peak at r=5216 marks the splitting of the player hand as is shown in (d). The segmentation which corresponds to r=5998 is shown in (e), and perceptually it is the most desirable partition. The large peak at r=8174 is due to the splitting of the background into two parts (f). Parameter setting: Intensity range is [0,1], a=8, b=0.1, edges whose weight w_{ij} is below 0.01 were eliminated, and only the four nearest neighbors plus four random neighbors were included for each pixel.

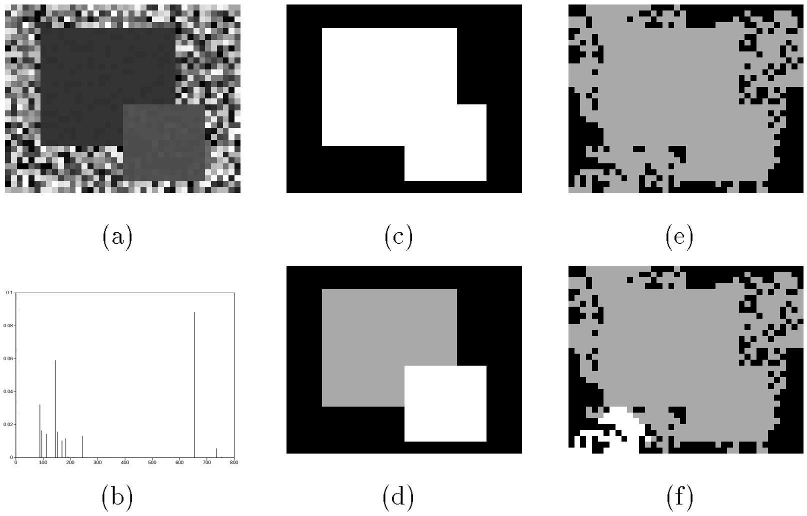


Figure 4: Separation of homogeneous objects from noisy background. (a) Synthetic 30×42 image, generated with the same parameters as in [13]: the brightness values are uniformly distributed between 0 and 1 for the background, between 0.2 and 0.21 for the larger rectangle, and between 0.3 and 0.31 for the smaller one. (b) The graph of $\Delta T(r)$: the largest peak appears at r=654, the second largest appears at r=146. (c,d) The segmentation results at r=654,655 that correspond to the largest peak of $\Delta T(r)$. Pixels in the Background form isolated or tiny clusters. (e,f) The segmentation results at r=146,147. These results are brought to show what kind of structure is captured by the bunch of peaks on the left of the graph in (b), all rejected by a cross-validation test. Parameter setting: a=3, b=0.1 (like in [13]), only the 8 nearest neighbors of each pixel are considered.

by intensity, where dI_{ij} is the intensity difference between pixel i and pixel j. As in [13, 16], we determine the parameters a and b manually. To reduce the number of edges and get a sparse graph, we eliminate edges whose weight is below some threshold and consider neighborhoods with short range interactions (see figure captions for details).

Figure 3(a) shows a gray level image of a baseball game. We use the same image used by Shi and Malik [16] to be able to compare the two methods. Figure 3(b) shows the graph of $\Delta T(r)$ obtained for this image. It is clear that there are only a few candidate solutions to consider. The peaks in this graph mark the detection of large objects in the scene. The corresponding segmentation levels are shown in Figures 3(c-f).

After forming the typical cut, some pixels (usually on object boundaries) are left isolated or form tiny clusters (say, of size < 30, which is less than 10^{-4} of the number of pixels in the image). It is a matter of choice whether these pixels should be assigned to one of the larger clusters or left "unlabeled". We prefer the first alternative, hence we merge each tiny cluster with its nearest neighbor according to the transformed weights, p^r_{ij}; only if this nearest neighbor is a large cluster does the isolated pixel actually gets labeled.

It appears that the results shown in Figure 3 are better than those reported in [16]. In particular, our algorithm segments fine details (like the palms of the hands of the lower player) while keeping the whole background as one part for a very long time; likewise, we have a level in which the circle-like object in the upper left corner is separated from the background.

Our next example follows the recent work of Perona and Freeman [13], who considered segmentation of "structured" foreground from "unstructured" background. Figure 4(a) shows a synthetic image that is generated according to the parameters reported in [13]. The significant difference between this example and the previous one is that here the background pixels are not similar to each other in terms of brightness level, and hence according to standard clustering criteria they cannot form a single group.

As shown in [13], the normalized cut algorithm [16] breaks down under these circumstances, since it is

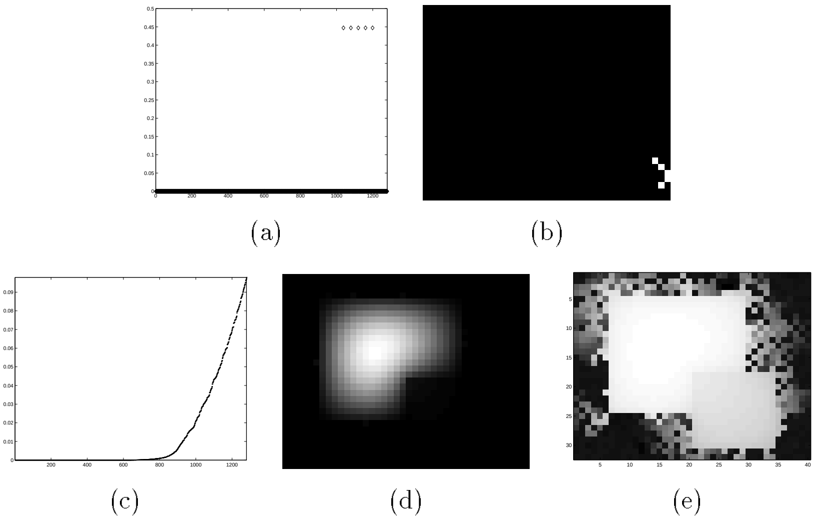


Figure 5: The performance of other algorithms applied to the data of Figure 4. The same preprocessing and exponential transformation from distances to weights was used (see text). **First row:** The normalized cut algorithm [16]. (a) One of the best Ncut partitions (there are many partitions which correspond to zero eigenvector, 33 of them separate a single pixel). (b) The partition which is suggested by the eigenvector shown on the left. The five isolated entries are connected to each other, but not to the rest of the image, hence the cut value is zero. **Second row:** The factorization method [13]. (c) The first eigenvector of the similarity matrix, sorted by the value of its entries. There is no clear threshold to choose. (d) The same eigenvector presented using intensity scale, where the entries are ordered like the image pixels. (e) A log function is applied to the values of the eigenvector, and the presentation is like in (d).

based on bi-partitions. On the other hand, our algorithm is based of multi-way cuts and has no difficulty dealing with such cases. The background pixels in this case form isolated or tiny clusters, which are ignored, and the foreground pixels form large and significant clusters.

In Figure 4(b) we show the impulse graph $\Delta T(r)$ obtained for this example. The largest peak at r=654 signals that the typical cut computed at this r-level undergoes a significant change: at r=655 the pixels of each rectangle are grouped together, and the background pixels form tiny groups which are ignored; at r=654 the two rectangles combine into one cluster. This transition is shown in Figures 4(c-d). As we go down to low r values, the clusters in the background grow and merge with each other and with the foreground set, as indicated by the bunch of peaks on the left of the graph. The highest one among them is at r=146, which corresponds to the transition from (e) to (f) in Figure 4. This bunch of peaks (all but the highest), however, were rejected by a cross-validation test, whose description is omitted here.

A comparison with the normalized cut algorithm [16] and with the factorization algorithm [13] is presented in Figure 5. The results are in agreement with those reported in [13], and show that our method performs much better. It is important to note that the same similarity matrix is used in all cases; there is no difference in the preprocessing stage which transforms the proximity and intensity difference of pixels into similarity weight, and the transformation we used is the same as reported in [13].

3.3 Color image segmentation

The similarity w_{ij} between pixels i,j is related to their proximity, as is the case for brightness images, and to the color resemblance between them. While brightness is a one dimensional attribute, the color resemblance is measured in some three dimensional color space.

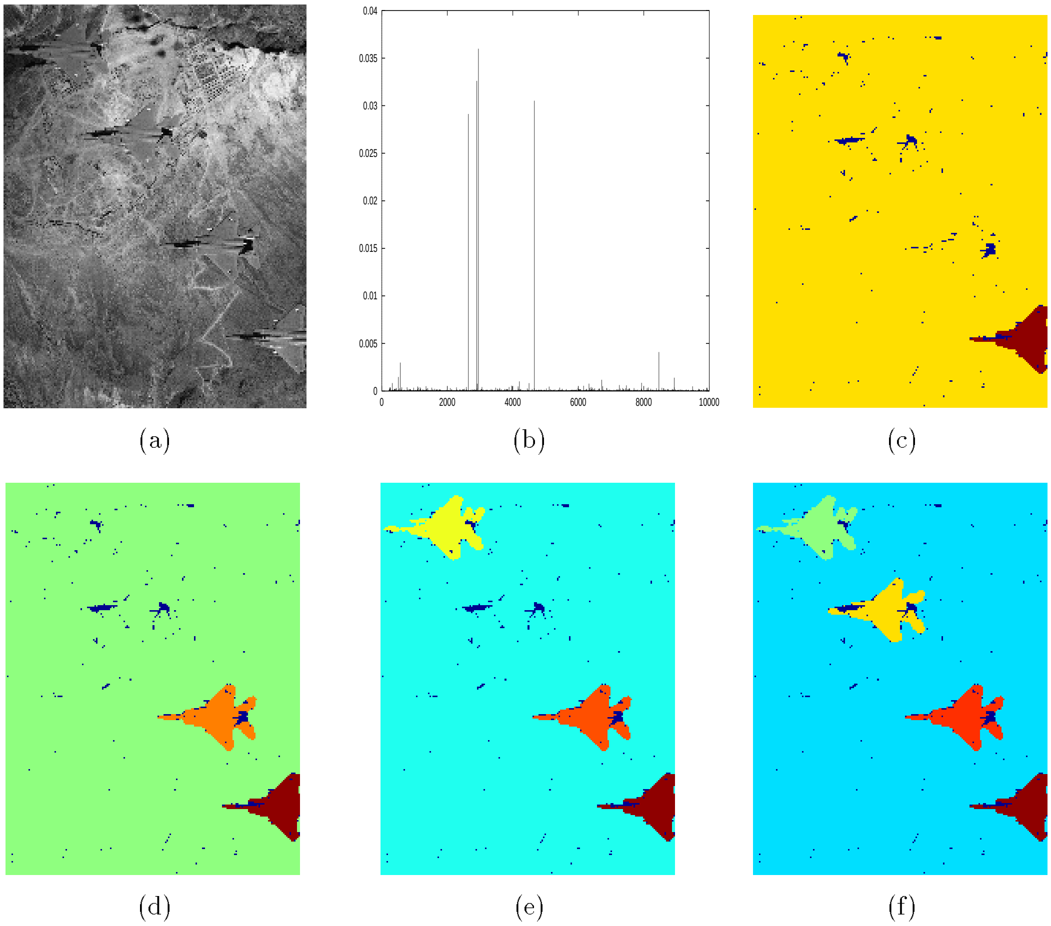


Figure 6: Color image segmentation. (a) The original image of size 256 X 192, reproduced in B&W. (b) The graph of $\Delta T(r)$. (c)-(f) The segmentation results which correspond to the four peaks of $\Delta T(r)$ at r=2645, 2903, 2951 and 4661. Every one of the peaks corresponds with the segmentation of one plane. Parameter setting: The colors are represented in CIELUV space. a=64, b=9, edges with weight w_{ij} below 0.001 were eliminated, and only the four nearest neighbors plus four random neighbors were included for each pixel.

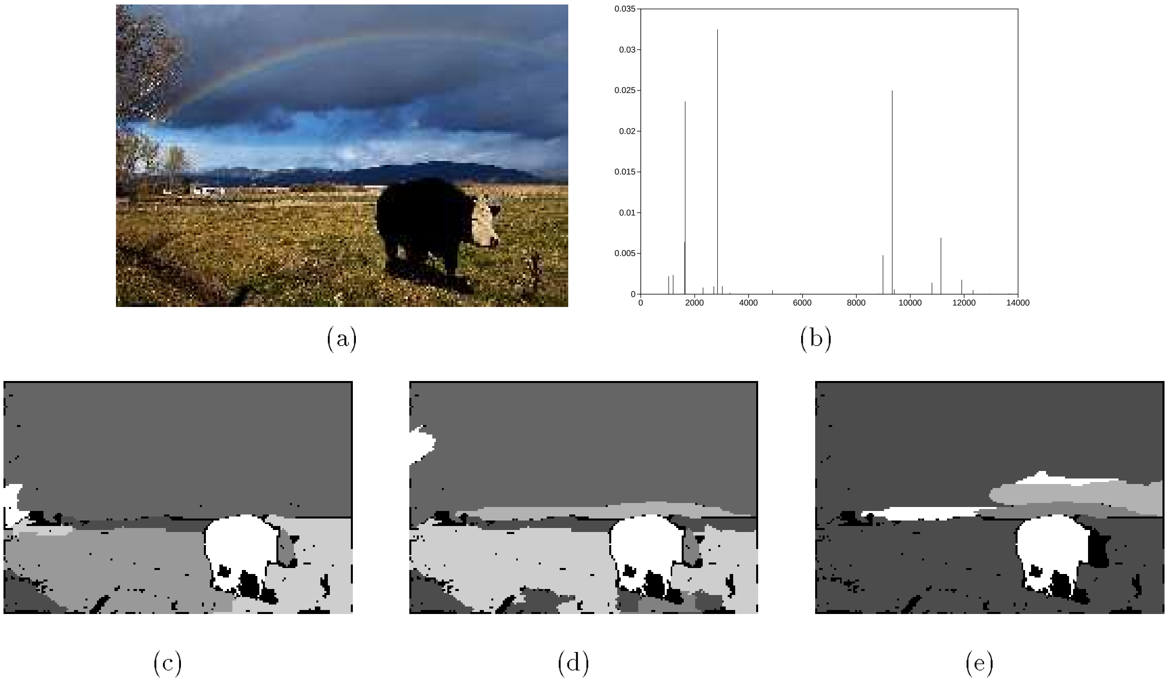


Figure 7: Color image segmentation. (a) The original color image of size 122 X 183. (b) The graph of $\Delta T(r)$. (c)-(e) The segmentation results which correspond to the three largest peaks of $\Delta T(r)$ at r=1651, 2848, 9331. Parameter setting: The colors are represented in CIELAB space. a=64, b=9, edges with weight w_{ij} below 0.001 were eliminated. Only the eight nearest neighbors were included for each pixel. Number of graph contractions: 500

A color space is called "perceptually uniform" if a small perturbation to one of its components is equally perceptible across the space. A decade of psychophysical research at the Commission Internationale de L'Éclairage (CIE) during the 1960's resulted in the definition of two color spaces, called CIELAB and CIELUV, which are approximately perceptually uniform. A third representation which we have tested follows [16], where colors are represented in transformed HSV space. Every color is mapped to the point $[v,\ sv\sin(h),\ sv\cos(h)]$, where saturation ($s$) and color value ($v$) are measured in [0,1], and hue (h) in $[0,2\pi]$.

Let dC be the Euclidean distance in the chosen color space, which is either CIELAB, CIELUV or transformed HSV. In accordance with our general rule, the pairwise similarity w_{ij} becomes $w_{ij} = \exp(-d_{ij}^2/a^2) \times \exp(-dC_{ij}^2/b^2)$. Our experiments do not suggest a preference for one of the color spaces over the others.

Figure 6(a) shows four airplanes flying above the dessert. The original image is in color, and it is reproduced here in black and white. We present the analysis in CIELUV color space. The peaks in $\Delta T(r)$ are shown in Figure 6(b), which clearly indicates the regions of different interpretation. Every peak corresponds to the separation of another plane from the background, as shown in Figures 6(c-f).

Figure 7(a) shows an outdoors scene with a cow, which is again a black and white reproduction of a color image. Here we present the analysis in CIELAB color space. The impulse graph of $\Delta T(r)$ is presented in Figure 7(b), and the segmentations which they indicate are shown in Figures 7(c,d,e).

3.4 Discussion

The spiral arrangement in the first synthetic example in Section 3.1 was chosen in order to emphasize a few advantages of our method. First, our algorithm does not exploit prior knowledge to model the data distribution, hence arbitrary cluster shapes can be handled. This is in contrast with parameter estimation methods, that usually assume Gaussian distributions. Second, our algorithm is robust to noise. This is in contrast with nearest neighbor methods such as single linkage, which are sensitive to "bridges" connecting large clusters. Third, though our algorithm is not given the desired number of clusters, it finds the desired natural partitions. Moreover, these partitions are not found by recursive divisions into two parts, the method employs by most other hierarchical methods.

Our method appears to work when the Ncut algorithm fails (Figure 2), and with equally good or better segmentation results (Figure 3) otherwise. In addition, the complexity of our algorithm appears to be lower: while our running time is $O(N \log^2 N)$, the complexity of Ncut is $O(MN)$, where M is the maximal number of matrix-vector multiplications allowed in the Lanczos procedure. The number M depends on many factors, and Shi and Malik observed that it is of the order of $O(\sqrt{N})$. Finally, with complexity $O(N \log^2 N)$ our method provides simultaneously hierarchical partitions at all level, whereas methods such as the normalized cut need to be applied repeatedly to achieve r-way cuts via intermediate 2-way cuts; this, in examples that have a lot of background noise, may add another multiplicative factor of $O(N)$ to the algorithm's complexity.

Likewise, our algorithm works in cases where the factorization method fails, as is demonstrated in Figure 2. Our segmentation results also seem better than those reported in [13] (Figure 4). In general, the factorization method is not suitable for chained data, since it looks for block structure in the similarity matrix.

References

[1] Blatt M., Wiseman S. and Domany E., "Data clustering using a model granular magnet", Neural Computation 9, 1805-1842, 1997.

[2] Boykov Y., Veksler O., and Zabih R., "Energy Minimization with Discontinuities", submitted to IJCV.

[3] Duda O. and Hart E., *"Pattern classification and scene analysis"*, Wiley-Interscience, New York, 1973.

[4] Gdalyahu Y., Weinshall D. and Werman M., "A Randomized Algorithm for Pairwise Clustering", Proc. NIPS, 1998.

[5] Geman S. and Geman D., "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images", IEEE PAMI (6):721–741, 1984.

[6] Hofmann T. and Buhmann J., "Pairwise data clustering by deterministic annealing", PAMI 19, 1-14, 1997.

[7] Ishikawa H. and Geiger D., "Segmentation by grouping junctions", Proc. CVPR, 125-131, 1998.

[8] Jain A. and Dubes R., *"Algorithms for clustering data"*, Prentice Hall, NJ, 1988.

[9] Karger D., "A new approach to the minimum cut problem", Journal of the ACM, 43(4) 1996.

[10] Klock H. and Buhmann J., "Data visualization by multidimensional scaling: a deterministic annealing approach", Technical Report IAI-TR-96-8, Institut fur Informatik III, University of Bonn. October 1996.

[11] Mumford D. and Shah J., "Boundary detection by minimizing functionals", Proc. CVPR, 1985, San Francisco, CA.

[12] Linial N., London E. and Rabinovich Y., "The geometry of graphs and some of its algorithmic applications", Combinatorica 15, 215-245, 1995.

[13] Perona P. and Freeman, W., "A factorization approach to grouping", Proc. ECCV, 1998.

[14] Rose K., Gurewitz E. and Fox G., "Constrained clustering as an optimization method", PAMI 15, 785-794, 1993.

[15] Sarkar S. and Boyer K., "Quantitative measures of change based on feature organization: eigenvalue and eigenvectors", Proc. CVPR, 478-183, 1996.

[16] Shi J. and Malik J., "Normalized cuts and image segmentation", Proc. CVPR, 731-737, 1997.

[17] Weiss Y., "Segmentation using eigenvectors: a unifying view", submitted, 1999.

[18] Weiss Y., "Smoothness in layers: motion segmentation using nonparametric mixture estimate", Proc. CVPR, 520-527, 1997.

[19] Williams L. R. and Thornber K. K., "A comparison of measures for detection natural shapes in cluttered backgrounds", submitted.

[20] Wu Z. and Leahy R., "An optimal graph theoretic approach to data clustering: theory and its application to image segmentation", IEEE PAMI 15, 1101-1113, 1993.