Perceptual Grouping and Segmentation by Stochastic Clustering

Yoram Gdalyahu[1,2] Noam Shental[1,2] Daphna Weinshall[1]
School of Computer Science and Engineering[1] and Center for Neural Computation[2]
The Hebrew University of Jerusalem, 91904 Jerusalem, Israel
e-mail: {yoram,daphna}@cs.huji.ac.il

Abstract

We use cluster analysis as a unifying principle for problems from low, middle and high level vision. The clustering problem is viewed as graph partitioning, where nodes represent data elements and the weights of the edges represent pairwise similarities. Our algorithm generates samples of cuts in this graph, by using David Karger's contraction algorithm, and computes an "average" cut which provides the basis for 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 N objects and a fixed accuracy level. 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 real color images. Our second application includes the concatenation of edges in a cluttered scene (perceptual grouping), where we show that the same clustering algorithm achieves as good a grouping, if not better, as more specialized methods.

1 Introduction

A wide range of tasks in computer vision may be viewed as unsupervised partitioning of data. Image segmentation, grouping of edge elements and image database organization, are problems at different levels of visual information processing. These tasks have different application objectives, and they handle very different data entities (pixels, edgels, images). Nevertheless, they all come to serve a common goal, which is the partitioning of the visual entities into "coherent" parts.

The goal of this work is to use cluster analysis as a unifying principle for a wide range of problems from low, middle and high level vision. In our approach we distinguish between two stages of processing. The first stage is task dependent, and defines the affinity, or similarity, between the visual entities. The affinity is a function of the relevant attributes. Low level attributes may be the spatial location, intensity level, color composition or filter response of a pixel in the image. Mid level attributes, in the case of edge elements, may be spatial location, orientation or curvature, and the affinity associated with them may reflect properties such as proximity, symmetry, co-circuitry and good continuity. High level attributes may be as complex as the entire shape of an object in the scene, or the color distribution of all the pixels in an image.

The second stage in this approach follows the unifying principle, and applies cluster analysis to the organizion of the visual objects (pixels, edgels, images) into coherent groups. These groups reflect internal structure among the entities, where (roughly speaking) the affinity within groups is larger than the affinity between groups. Therefore, a cluster of pixels in the image, sharing similar locations and colors, is expected to account for an object or a part of an object in the scene. A cluster of edge elements is expected to exhibit a meaningful aggregation into a complete edge, and a cluster of images in a database is expected to be related with a common topic.

We present in Section 2 our stochastic pairwise clustering algorithm; it is an efficient, robust and model-free pairwise hierarchical algorithm (see also [2]). The robustness of our method is achieved by averaging over all the possible interpretations of the data, giving more weight to data partitions that are associated with lower cost. This idea is adopted from clustering algorithms that are inspired by statistical mechanics, and in particular our method is related to [1]. Our algorithm can be analyzed analytically, and for sparse graphs and a fixed accuracy level it runs in $O(N \log^2 N)$ time, where N is the number of data-points.

Clustering methods which are formulated as graph partitioning (as we do here) involve partitioning criteria such as the min-cut algorithm [12]. They are related to spectral methods which identify good partitions via the eigenvectors of the affinity matrix, or other matrices derived from it [6, 7, 9]. These methods have been applied, in the context

of computer vision, to image segmentation, motion estimation and perceptual grouping.

In Sections 3,4 we describe the application of our clustering algorithm to color image segmentation and perceptual grouping of edge elements. A description of its application to image database organization is beyond the scope of the present short report. Our good results (both absolute and relative to other methods), and the results of extensive experiments not described here, demonstrate that we can handle successfully problems in low- and mid-level vision in this unified way. In comparison, other successful algorithms have been shown to be more specific; for example, it has been shown in [6] that the normalized cut algorithm [9] is not suitable for figure/ground segregation (or perceptual grouping), while the factorization method in [6] is less suitable for image segmentation.

2 The typical cut algorithm

We present now the general approach and the principles of our clustering method. After defining the terminology in Section 2.1, we describe the algorithm in Section 2.2. Complexity and parameter tuning are discussed in Section 2.3.

2.1 Notations and Definitions

Our clustering algorithm uses pairwise similarities, which are represented as a weighted graph $G(V, E)$: the nodes V represent data items, and the positive weight w_{ij} of an edge (i, j) represents the similarity between nodes (data 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 triangle inequality, and self similarities w_{ii} are not defined.

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 weights of all edges that cross the cut, namely: $c(V_1, V_2) = \sum_{i \in V_1, j \in V_2} w_{ij}$. A *minimal cut* has the 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 side V_α $(\alpha = 1 \ldots r)$ be grouped together into one *meta-node*, and discard all the edges which form self loops within meta-nodes (namely, discard the inner edges of each component, which connect two inner nodes belonging to the same component). The graph which is thus 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 is a complete graph, then the number of edges connecting the meta-nodes representing 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 Outline of algorithm

This section provides a simplified concise description of the algorithm, ignoring implementation issues which arise from considerations of space and time complexity, and emphasizing the general principles. The algorithm is divided into two stages, described in pseudo-code in Figs. 1,2 and explained below.

Generating typical cuts: For a given value of r ($r = 1 \ldots N$) our algorithm generates a sample of M possible r-way cuts, and uses this sample to estimate the probability p^r_{ij} that (i, j) is not a crossing edge of a random r-way cut. The pseudo-code in Fig. 1 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 on the same side. These accumulators are divided by M to estimate for every two nodes the probability p^r_{ij} that they are on the same side.

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 previously connected these two meta-nodes. The selection of the nodes to be unified is probabilistic: 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 adjacent to the selected edge are merged.

The contraction procedure is the cornerstone of our method, since it defines the sample of M cuts, according to which the empirical probabilities p^r_{ij} are computed. Thus the contraction procedure is our sampling tool, typically assigning higher probability to cuts with lower capacity as is shown in [5]. In fact, [5] proves that the minimal cut can be found using this sampling method in polynomial time, even though the overall number of possible cuts is exponential. In summary, the contraction process induces a probability

```
procedure STAGE-1:
input:   weighted graph G(V,E), N nodes
output:  3D array p of probabilities

s^r_ij ← 0 for i,j,r = 1...N   (init counters)
for m = 1...M:
   G'_N ← G(V,E)
   for r = (N−1)...1:
      G'_r ← CONTRACT(G'_{r+1})  (r-way cut)
      for i,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
p^r_ij ← s^r_ij/M for i,j,r = 1...N
return array p.
```

Figure 1. Pseudo-code which transforms similarity weights into pairing probabilities.

distribution over cuts, and under this distribution we estimate p^r_{ij} – the marginal probability that nodes i and j are on the same side of a random r-way cut.

The number of r-way cuts in a graph of N nodes is the Sterling number of the second kind, denoted $\tau(r,N)$. Let $\alpha(r) = 1 \ldots \tau(r,N)$ be an index to the set of all r-way cuts in $G(V,E)$. Fix r and let P_α denote the probability that the contraction algorithm generates the cut α. For a fixed r value, $\sum_\alpha P_\alpha = 1$. Define an indicator variable e^α_{ij} to be 1 if the edge (i,j) crosses the cut α and 0 otherwise. It is readily seen that

$$\sum_{(i,j)\in E} w_{ij}(1-p^r_{ij}) = \sum_{(i,j)\in E} w_{ij} \sum_\alpha e^\alpha_{ij} P_\alpha = \sum_\alpha c_\alpha P_\alpha = \langle c(r) \rangle$$

where c_α is the capacity of cut α, and $\langle c(r) \rangle$ is the expected value of the r-way capacity. We can therefore interpret 1-p^r_{ij} as the probability that edge (i,j) is a crossing edge in an "average cut". We use this observation for the following definition.

For every integer r between 1 and N we define the *typical cut* $(A_1, A_2, \ldots, A_{s(r)})$ as the partition of G into connected components, such that for every $i \in A_\alpha$, $j \in A_\beta$ ($\alpha \neq \beta, \alpha, \beta = 1 \ldots s(r)$) 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, $s(r)$, 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 exponential number of 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.

Selecting meaningful partitions: We define the following function of the typical cut at level r:

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

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 large meta-nodes are merged.

We 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 if and only if $\Delta T(r) > \delta$. This is described in Fig. 2.

2.3 Complexity and parameter estimation

An efficient implementation of our algorithm drastically decreases the number of estimated variables (p^r_{ij}) from N^3 to $|E|$, which is $O(N)$ for sparse graphs and N^2 for complete graphs. Moreover, we can show that one graph contraction (one iteration of the external loop in `STAGE-1`, Fig. 1) can be implemented in $O(N \log N)$ time for sparse graphs. Using the Hoeffding-Chernoff bound to determine the desired sample size, denoted M, we can show that $M = O(\log N/\epsilon^2)$ for an accuracy level ϵ (with high probability). Hence the overall complexity of `STAGE-1` for a fixed accuracy level is $O(N \log^2 N)$ for sparse graphs. The efficient implementation of `STAGE-2` takes only $O(N \log N)$ time in this case, making $O(N \log^2 N)$ the overall sparse graph complexity bound for our algorithm.

There are very few parameters in the main part of the algorithm, and its performance is not sensitive to their exact values. It is mostly the preprocessing stage, which constructs the weighted graph G, that critically depends on external parameters. These parameters are related to the definition of similarity (which is task dependent), and to the transformation from perceptual similarity to edge weight.

```
procedure STAGE-2:
input:   3D array of probs p_ij^r
output:  selected partitions

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

Figure 2. Pseudo-code which finds typical cuts, measures the resemblance between subsequent cuts, and reports the "meaningful" partitions.

Given a dissimilarity measure d_{ij} between stimuli i and j, we define $w_{ij} = \exp(-d_{ij}^2/a^2)$. Here a is a decay parameter which reflects some suitable local scale, and it needs to be tuned. Sometimes the dissimilarity between stimuli is measured along different dimensions, like in an image segmentation task where dissimilarity between pixels is a function of their spatial proximity and relative brightness. In this case a different local scale parameter is defined for every dimension μ of similarity, namely:

$$w_{ij} = \prod_{\mu} \exp(-d(\mu)_{ij}^2/a(\mu)^2) \qquad (2)$$

Experiments with simulated and real data show that the results of our algorithm are quite robust with respect to the exact transformation used to implement Equation (2).

2.4 Robustness and Comparisons

Figure 3 shows a task involving the separation of two point sets generated by different statistical sources. The issue addressed here is the response of various clustering algorithms to the amount of noise in the data. The normalized cut algorithm [9] performs well at low levels of noise, but as the noise is increased it abruptly breaks down. In this example the factorization method [6] breaks down at a lower level of noise, while our algorithm still performs well at higher levels of noise. In general, as the level of noise is increased continuously, the performance of our algorithm degrades gracefully and continuously, by leaving more and more points unlabeled. Abrupt breakdown is rare, and we believe that this is another beneficial result of the stochastic nature of our method.

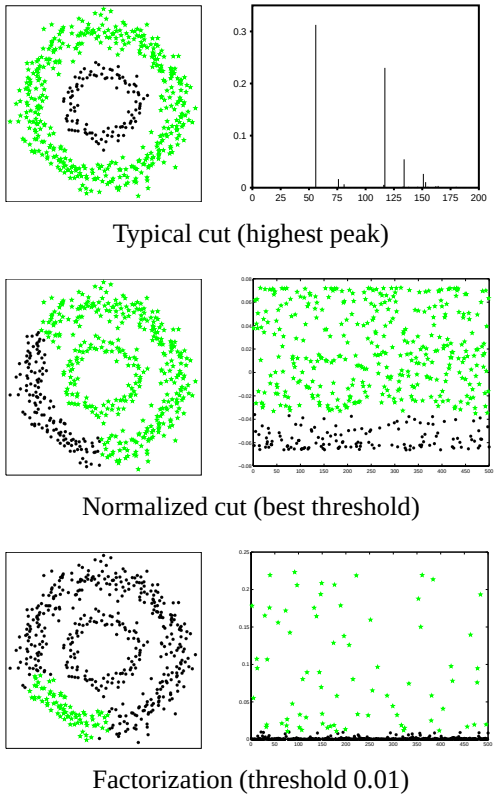

Figure 3. Robustness under data perturbation. The control parameter η defines the spread of the points, or the "width" of each circle; results are shown for η=0.9. Both the normalized cut algorithm and the factorization algorithm fail (the factorization method fails earlier, though, for η=0.7), while our algorithm finds the desired structure. For the spectral methods we show the entries of the relevant eigenvector next to the partition found. The magnitude of the entries are plotted versus their serial index. These methods seek a threshold which separates between small and large entries. The color and the symbol used for the eigenvector entries corresponds with those used in the $2D$ plot.

3 Color image segemntation

We now describe color image segmentation with the typical cut algorithm; this application is similar to intensity image segmentation, which we have explored earlier in [2]. Now nodes in the graph represent individual pixels. The similarity weight w_{ij} between pixels (nodes) i and j increases with increasing spatial proximity and color resemblance, which is measured in a three dimensional color space. If using one of the "perceptually uniform" color spaces, CIE-LAB or CIE-LUV, color difference is

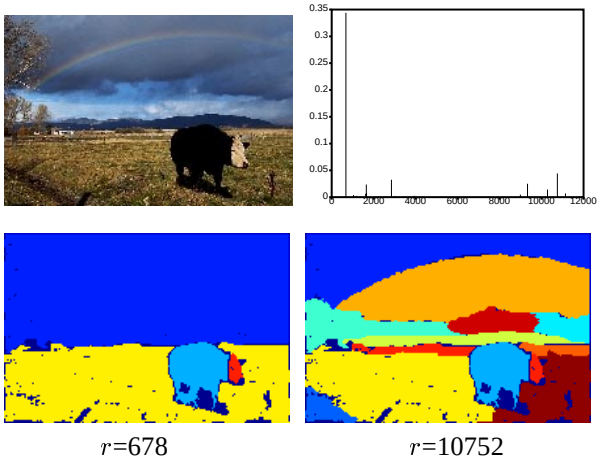

Figure 4. Color image segmentation in CIE-LAB color space: the original image of size 122×183, the impulse graph of $\Delta T(r)$, and two segmentation results which correspond to the two highest peaks. **Parameter setting:** Colors are represented in CIE-LAB color space, a=8, b=3, edges with weight w_{ij} below 0.001 are eliminated, and only edges connecting each pixel with its eight spatial nearest neighbors are included. Minimal cluster size of interest is 100, and the sample size is M=500. The graph contained 71,765 edges. Time per iteration: 0.89sec on Pentium II 450 Mhz.

the Euclidean distance in the corresponding space. From Equation (2) we have:

$$w_{ij} = e^{-\frac{d(1)_{ij}^2}{a^2} - \frac{d(2)_{ij}^2}{b^2}} \quad (3)$$

where $d(1)$ is the Euclidean distance between the pixels in the image plane, $d(2)$ is the color difference (measured in a *3D* color space) between the two pixels, and a, b are corresponding scale parameters. As in [6, 9], 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 short range neighborhoods (see captions of Figs. 4,5 for details).

The observation that the similarity graph is sparse is crucial for practical image segmentation applications. In [9] a sparse graph was obtained by randomly picking a small number of edges for each pixel in a limited range neighborhood. We adopt a similar approach: we consider the eight nearest neighbors of each pixel, or we consider the four nearest neighbors and randomly pick another four (we do not observe significant differences between these two methods). In addition, we eliminate edges whose weight is below some threshold. Two examples are shown in Figs. 4,5.

4 Perceptual grouping of line segments

We address here the separation of structure from cluttered background, and specifically the problem known as

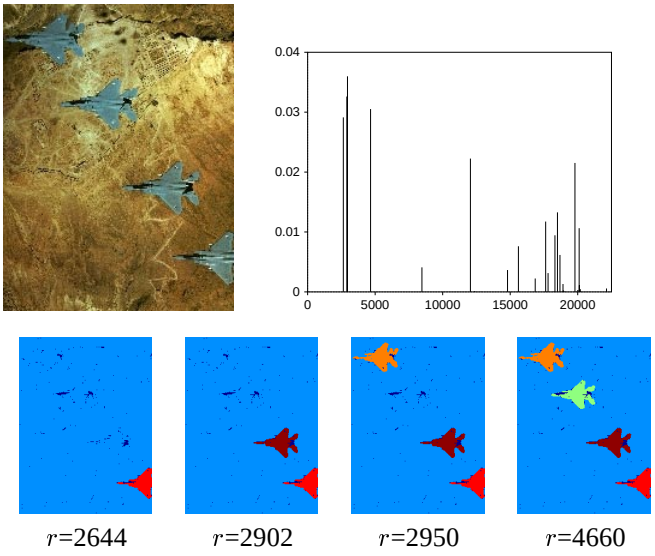

Figure 5. Color image segmentation in CIE-LUV color space: the original image of size 256×192, the impulse graph of $\Delta T(r)$, and the four segmentation results which correspond to the four highest peaks. **Parameter setting:** Colors are represented in CIE-LUV color space, a=8, b=3, edges with weight w_{ij} below 0.001 are eliminated, and only edges connecting each pixel with its four spatial nearest neighbors plus four random neighbors are included. Minimal cluster size of interest is 100. The graph contained 255,837 edges. Time per iteration: 4.10sec on Pentium II 450 Mhz; M=500.

perceptual grouping of edge elements, with the same typical cut algorithm. The term perceptual grouping is usually associated with mid level vision, and more specifically with the grouping of edge elements. The history of this problem goes back to the taxonomy of non-accidental properties explored the Gestalt psychologists at the beginning of the 20th century. Among the methods of choice are relaxation neural networks [8], cost minimization using simulated annealing [4], spectral decomposition of the affinity matrix [7, 3], and stochastic completion fields [10].

The raw data is a set of line segments, or edgels, and the task is to group together a set of edgels that together define a perceptually appealing "edge", typically expected to have one or more of the following properties: smoothness, closure, or convexity. Perceptual grouping algorithms consist of two parts: (i) the combination of mutual properties which are usually non accidental into a pairwise affinity measure, and (ii) the detection of internal structure, or shape. In our approach, nodes in the graph represent edge elements, edge weights represent pairwise affinities, and the typical cut algorithm is used to detect the hidden shape.

Step (ii) above, which is the grouping step, is frequently formulated as a problem of *saliency detection*. This salien-

cy formulation is analogous to soft clustering: its goal is to assign a value to each edgel, representing to what extent it is a part of a salient shape (cluster) or a part of some background noise. In contrast with the saliency formulation, "hard" clustering algorithms (like ours) divide the data into disjoint sets. When saliency algorithms are required to provide such dichotomic division, it is sometimes possible to find a saliency threshold which divides the edgels into shape and noise classes.

In order to quantitatively compare between different saliency algorithms, while not being able to rely on the availability of a "salient" saliency threshold, Williams and Thornber (WT), in an extensive comparative study [11], used the number of edgels in the shape they were looking for as given apriori. Assuming that the hidden contour consists of n edgels, they selected the n edgels with highest saliency values and labeled them as "shape" edgels. Based on this somewhat artificial partitioning into shape and background, they defined the *false positive ratio* as the percent of noise edgels in the set labeled as "shape". The false positive ratio served to compare between several grouping methods (see below).

Our approach has the advantage that it does not involve thresholding, nor does it assume prior knowledge of the number of signal elements. Our typical cut algorithm returns, at different r levels, a set of edgels which is the shape candidate. In Section 4.2 we explain how the shape hypothesis is selected. Once this is done, and in order to compare our results with the quantitative study of WT, we compute our false positive ratio as the percent of noise edgels in the selected shape cluster (which might be of any size). Alternatively, we can feed the cluster we find into a saliency algorithm, and boost its performance as measured by WT's false positive ratio.

4.1 Experimental framework

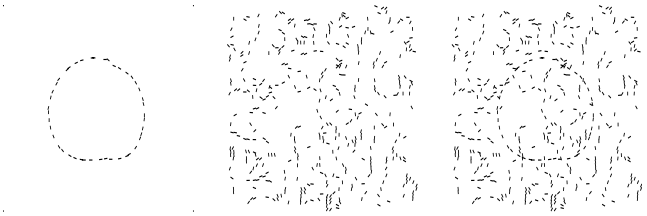

Figure 6. **a)** Peach silhouette; **b)** 'bark' natural texture; **c)** the combined **a** and **b** test pattern, with *SNR* 0.1

The experimental setting follows [11]. In this framework the test set included a series of patterns similar to the one shown in Fig. 6c: one of 9 fruit or vegetable silhouettes (like the peach in Fig. 6a) was superimposed on one of 9 natural textures (like the texture in Fig. 6b). An additional parameter determined the Signal-To-Noise ratio (*SNR*) of the pattern, using 5 levels: 0.25, 0.2, 0.15, 0.1 and 0.05. *SNR* was defined as the number of signal (fruit) edgels divided by the total number of edgels. With the lowest *SNR* the test pattern consisted of approximately 900 edgels.

WT measured how well the signal - the fruit or vegetable silhouette - can be segregated from the background texture pattern by different saliency methods. Since the evaluation targeted only the saliency measure, a pairwise affinity matrix A was generated in advance and served as input to all the methods. Specifically, for each test pattern A_{ij} was the similarity between the pair of edgels i and j. The affinity computation used the relative positions and orientations of i and j, and produced a measure of similarity according to the Gestalt principles of good continuation and proximity. Several grouping methods [3, 4, 7, 8, 10, 11] were applied to the same affinity matrix A, and every edgel was assigned a saliency value according to each method. Performance was quantified using the false positive ratio, as discussed above. Results are shown in Fig. 7.

In our experiments we repeated the computation of the affinity matrix A for each one of the 81 test patterns and for each one of the 5 *SNR* values. The affinity matrices were fed into our typical cut algorithm, and the results were interpreted as discussed below.

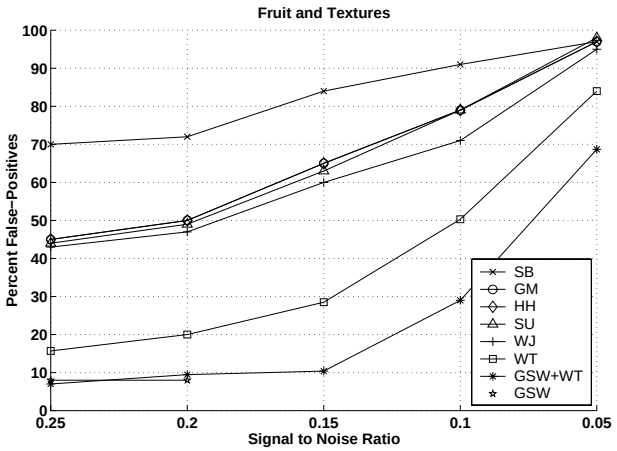

Figure 7. False-Positive-Ratio as a function of *SNR* for different saliency methods, calculated over 81 test patterns for each *SNR* value. The results of applying the methods of [7] (SB), [3] (GM), [4] (HH), [8] (SU) and [10] (WJ), were taken from [11]. The results of applying the method of [11] (WT) was taken from our own simulations, as well as the results of our boosted method (GSW+WT) and partial results of our own method (GSW) (see text).

4.2 The typical cut for edgel grouping

The affinity values are used to assign weights to the edges of a graph, whose nodes represent edge elements (edgels). The typical cut algorithm provides N partitions of this graph, each corresponds with a different r value. A partition of the graph consists of a variable number of clusters. However, since it is assumed in this domain that there is only one significant structure in the data, we use the largest cluster as the signal hypothesis for each value of r. The experiments show that for *all* test patterns, as long as the edgels of the fruit (vegetable) silhouette are in one cluster, this cluster is the largest one.

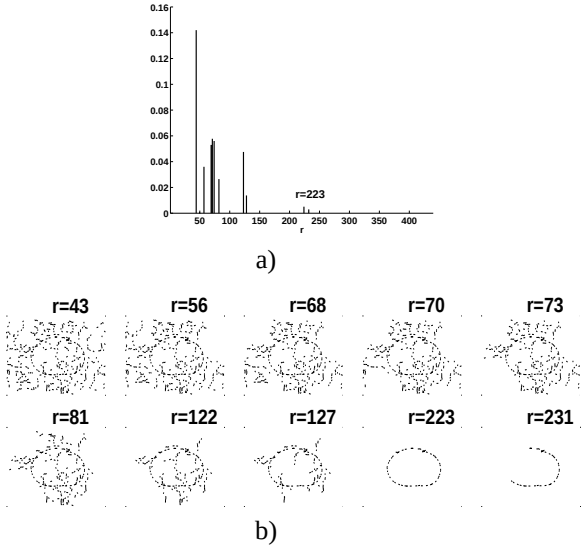

Figure 8. a) The resulting impulse graph of $\Delta T(r)$ for the image in Fig. 6c. b) Edgels that belong to the largest cluster, for a few selected peaks from a).

Fig. 8 shows the results for the peach test pattern of Fig. 6c, including the impulse graph of $\Delta T(r)$ and the corresponding signal identified at each peak. It is easy to see that every peak in $\Delta T(r)$ is associated with removing a 'chunk' of noise elements from the identified signal. At r=223 the identified signal is rather clean, and it then breaks into sub-parts. Another example is shown in Fig. 9.

The results in Fig. 8 are typical, illustrating the qualitative behavior of our algorithm for *all* test patterns, and demonstrating the problem with our selection criterion. Our heuristic identifies meaningful partition levels by measuring variations in $T(r)$. When the clusters of interest are large in size, it is possible to relate meaningful partitions with large variation in $T(r)$. In problems of edgel grouping this is typically not the case, as the cluster of interest contains only a few elements in comparison with the number of elements in the background (the clutter). In this case, the interesting peak in $\Delta T(r)$ cannot be detected by its height

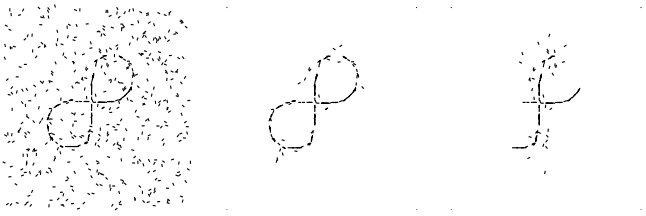

Figure 9. An example taken from http://iris.usc.edu/~ tensorvt (courtesy of MS Lee). **a)** The image consists of 551 edgels, 109 of which belong to the tilted '8' signal (*SNR* of 0.2). **b)** The output of "grouping by clustering", with affinity matrix A calculated as above: the percent false positive and false negative is 22 and 0, respectively. **c)** The output of "grouping by saliency": WT's 109 most salient edgels.

alone, and the peaks in $\Delta T(r)$ serve to provide a small set of candidate partitions to choose from.

To address this problem we designed a selection operator, which automatically identifies the "best" peak in the graph of $\Delta T(r)$. This operator checks whether a closed curve that exists at level r_{peak} breaks into parts at level $r_{peak}+1$. In the example of Fig. 8b, our operator correctly finds $r = 223$.

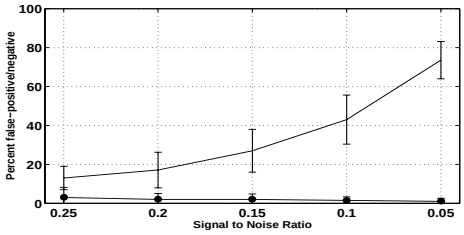

Figure 10. Percent false positive (top curve) and false negative (bottom curve) of the edgels in the signal hypothesis found by our method. The plots show mean performance and standard deviation over 81 test patterns, for each value of *SNR*.

Fig. 10 shows the performance of this procedure over the test patterns, showing false positive (i.e., the percentage of noise edgels within the final result) and false negative (i.e., the percent of signal edgels not identified) rates.

4.3 Clustering vs. saliency

Clustering has one crucial advantage over saliency, for the purpose of grouping: a signal hypothesis (for the correct silhouette) can be readily generated from the set of edgels in the largest output cluster. Saliency, on the other hand, is a measure assigned to all edgels, from which edgel segmentation should be derived. Thus, unless the number of edgels in the signal is known in advance, or the saliency values of noise edgels is much lower than the saliency of

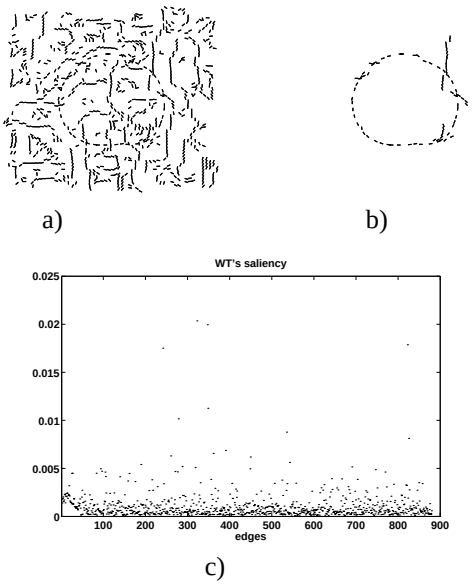

Figure 11. **a)** The peach test pattern with *SNR* 0.05; **b)** result (signal hypothesis) of our method. **c)** WT's saliency value computed for the edgels in pattern **a)**, where the edgels corresponding to the peach are numbered 1–44; the *false positive ratio* in this example is 0.95.

the signal (which is often not the case), one cannot easily generate a signal hypothesis from saliency maps.

An example is given in Fig. 11, which shows a peach test pattern and the signal as identified by our method. Fig. 11c shows WT's saliency values for the edgels of the test pattern in Fig. 11a, where the edgels corresponding to the peach silhouette are numbered 1–44. It is clearly difficult to isolate those edgels based on the saliency map alone. Our clustering method, however, readily identifies a rather 'clean' signal hypothesis (Fig. 11b). A similar point is made by the example in Fig. 9.

This difference between clustering and saliency makes it difficult to compare the two approaches. In the analysis summarized in Fig. 7, the number of edgels n in the signal is assumed to be known, and thus percent false positive can be computed from the n most salient edgels. Clustering, however, returns a complete hypothesis for the signal, and this hypothesis typically has a number of edgels different from n. When we compute percent false positive with respect to the cluster found (as in Fig. 10) we *do not* assume prior knowledge of n. Even with this handicap and ignoring the difference in the analysis, when comparing the results in Fig. 7 to our results in Fig. 10, we see that our results are similar or better than the results of the best saliency method (WT).

Partial results, including only cases where the size of the signal hypothesis produced by our algorithm is not larger than $1.1 \times n$, suggest that our algorithm does better than saliency methods (see curve GSW in Fig. 7). Alternatively, we may combine clustering with saliency. We use the largest cluster, obtained by our method, as input to a saliency algorithm to boost the performance of both methods. Using the WT's saliency method we obtain the curve (GSW+WT) in Fig.7; when compared with the partial clustering results (GSW), the boosted algorithm does not show improvement.

Acknowledgements: This work was partly funded by European grant "Vigor" contract No. ESPRIT 26247-VIGOR. We thank Lance Williams and Mi-Suen Lee for generously giving us pointers to their data.

References

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

[2] Gdalyahu Y., Weinshall D., and Werman M., "Stochastic Image Segmentation by Typical Cuts", *in Proc IEEE CVPR*, 1999.

[3] Guy G. and Medioni G., "Inferring Global Perceptual Contours from Local Features", *Intl. Journal of Computer Vision* **20**:113-133, 1996.

[4] Herault L. and Horaud R., "Figure-Ground Discrimination: A Combinatorial Optimization Approach", *IEEE-PAMI* **15**:899-914, 1993.

[5] Karger D., "Minimum cuts in near linear time", *in Proc. of the 28th Annual ACM Symposium on the Theory of Computing*, 56-63, 1996.

[6] Perona P. and Freeman W., "A factorization approach to grouping", *in Proc. ECCV:*655-670, 1998.

[7] Sarkar S. and Boyer K., "Quantitative Measures of Change based on Feature Operators: Eigenvalues and Eigenvectors", *in Proc. IEEE CVPR:*478-483, 1996.

[8] Shashua A. and Ullman S., "Structural Saliency: The Detection of Globally Salient Structures Using a Locally Connected Network", *in Proc. 2nd ICCV*, 1988.

[9] Shi J. and Malik J., "Normalized cuts and image segmentation", *in Proc. IEEE CVPR:*731-737, 1997.

[10] Williams L.R and Jacobs D.W., "Stochastic Completion Fields: A neural Model of Illusory contour Shape and Salience", *Neural Computation*, **9**:849-870, 1997.

[11] Williams L.R. and Thornber K.K, "A Comparison of Measures for Detecting Natural Shapes in Cluttered Backgrounds ", *in Proc. ECCV*, Germany, 1998.

[12] 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.