A Factorization Approach to Grouping

P. Perona[1][2] and W. Freeman[3]

[1] California Institute of Technology MS 136-93, Pasadena CA 91125, USA
[2] Università di Padova, Italy
[3] Mitsubishi Electric Research Lab., 201 Broadway, Cambridge MA 02139

Abstract. The foreground group in a scene may be 'discovered' and computed as a factorized approximation to the pairwise affinity of the elements in the scene. A pointwise approximation of the pairwise affinity information may in fact be interpreted as a 'saliency' index, and the foreground of the scene may be obtained by thresholding it. An algorithm called 'affinity factorization' is thus obtained which may be used for grouping.
The affinity factorization algorithm is demonstrated on displays composed of points, of lines and of brightness values. Its relationship to the Shi-Malik normalized cuts algorithms is explored both analytically and experimentally. The affinity factorization algorithm is shown to be computationally efficient (O(n) floating-point operations for a scene composed of n elements) and to perform well on displays where the background is unstructured. Generalizations to solve more complex problems are also discussed.

1 Introduction

Fig. 1 shows a distribution of points in the plane. Taken one by one the points are identical to each other. However, it is quite apparent that their mutual positions in the plane contain some 'global' information. While it is somewhat unclear how to define this global information, it is natural to consider a local property: the 'pairwise similarity' of these points: two points that are close by are 'similar' and two points that are far apart are 'different'.

This is a common situation in vision: we extract tokens from an image using some early visual process, and the notion of 'closeness' between pairs of tokens is natural and well defined. The tokens may be anything: from pixels to points, to edgels, to textured patches. While it is unclear how to extract, and even how to define, the global high-level properties of the scene, it is easy and natural to define the pairwise affinity of any two tokens. This idea comes to us from the work by Shi and Malik on grouping using normalized-cuts (see [8] and references therein). In this paper we explore a weaker approach than that of Shi and Malik: rather than formulating a grouping problem explicitly, we notice that a useful global property of the scene, the foreground set, may be both 'discovered' and estimated starting from the notion of pairwise closeness, or pairwise affinity, of individual elements. We develop a simple algorithm which factorizes the matrix of pairwise element affinities, and compare it with the algorithm of Shi and Malik.

2 The affinity function

Given two elements (i,j) in the scene S, let's suppose that it is possible to assign a number $A_{i,j}$ that tells us how 'similar' the two objects are. When the two objects are very similar then $A_{i,j}$ has a high value and when they are not similar at all then $A_{i,j} \approx 0$. We will suppose that this affinity function has the following properties:

$$\forall i,j \in S: \quad A_{i,j} \in [0,1], \quad A_{i,i} = 1, \quad A_{i,j} = A_{j,i}$$

The first and second properties impose a normalization on A. The third property is a symmetry requirement.

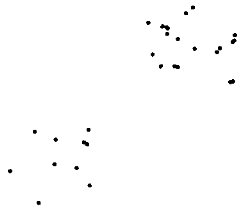

Fig. 1. A set of points on the plane

As an example, two points in Fig. 1 are 'similar' if they are close-by. One could define $A_{i,j}$ as:

$$\begin{aligned} A_{i,j} &= e^{-d^2(i,j)} \\ d^2(i,j) &\doteq \frac{||x_i - x_j||^2}{d_o^2} \end{aligned} \tag{1}$$

where d_o is a reference distance below which two points are thought to be similar and beyond which two points are thought to be dissimilar, and x_i is the vector of coordinates of point i. There is nothing magical about this particular definition of affinity: depending on the situation another definition may be more natural and appropriate. Affinity functions for line segments and other objects will be discussed later.

3 $O(n)$ approximation of the affinity matrix

Notice that if the scene contains N objects we need $(N-1)N/2 = O(N^2)$ numbers in order to describe its affinity properties. If the image is composed of tens of thousands of objects (points, lines, pixels), as it is the case in vision, this means hundreds of millions of numbers. It makes sense to wonder whether we could describe some fundamental aspects of the scene in a cheaper way: rather than by a function A_{ij} of object pairs (i, j), by a pointwise function p_i of objects i taken one by one. This way the complexity of our description will drop to $O(N)$ (bits, as it will turn out).

We propose, as a possible approach, to approximate A by products of p's: $A_{i,j} = p_i p_j$. We are trading off some accuracy in representing A with a great improvement in storage costs. What is the best p that we could use? This depends on our definition of approximation error. We will start by using the L^2 norm in order to define a distance between two matrices:

$$p = \arg\min_{\hat{p}} \sum_{i,j=1}^{N} (A_{i,j} - \hat{p}_i \hat{p}_j)^2 \tag{2}$$

the L^2 norm is not necessary (see the discussion section at the end).

Notice that we are approximating A with the rank-one matrix pp^T. The best solution to this problem (in the sense that it minimizes the Frobenius, or L^2, norm of the approximation error) is well known:

Proposition 1 The best order-one L^2 approximation of a matrix A is the first singular vector of the matrix A multiplied by the square root of the corresponding singular value. Calling (U, S, V) the singular value decomposition of A, and U^i the columns of U and $\sigma_i^2 = S_{i,i}$ the singular values of A we have:

$$p = \sigma_1 U^1 \tag{3}$$

Proof : See [6].

Notice that $A = A^T$ and therefore $U = V$. Also: since $A = A^T$ p is also equal to the eigenvector v_1 of A with largest eigenvalue λ_i: $p = \lambda_1^{1/2} v_1$.

Let's take a look at Fig. 2 where the function p that we obtain for the example of Fig. 1 is shown. We may notice that p takes values between 0 and 1, and that for some indices i the value of p is exactly equal to 0. In Fig. 2(right) the function p is shown together with the position of the points. We may notice that the points for which p is different from zero are in the same region of the picture, with higher values of p for points in the middle of that region. For reasons that will become apparent later let's call 'foreground' that region, and 'background' the rest of the points.

How well does pp^T approximate A? One may take a look at the error in approximation of the entries of A (Fig. 3, left group, right matrix), however, this does not give much insight. If we permute the entries of A using the permutation of indices that sorts the values of p in ascending order (see Fig. 3, right group), then the picture becomes clearer. Notice that the approximation is poor when both i and j belong to the background group, while it is better for both the foreground group and the mixed terms $A_{i,j}$ with i belonging to the foreground and j belonging to the background.

Let's summarize our observations so far:

1. The function p takes values between 0 and 1.

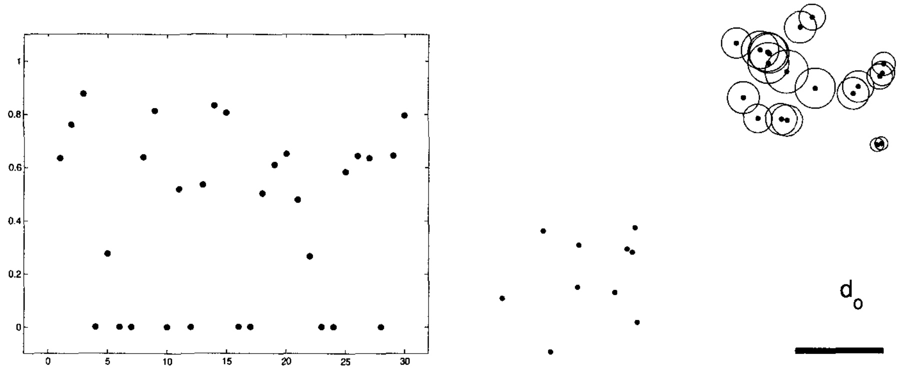

Fig. 2. (left)The approximant p for the distribution of points shown in Fig. 1. The affinity function defined in Eq. 1. The reference distance d_0 is shown in the display on the right. (Right) Pictorial representation of p: a dot indicates each point's position; the radius of the circle around each dot is proportional to the magnitude of the corresponding value of p. Points for which $p = 0$ do not have a visible surrounding circle. The reference distance d_0 is shown in the lower right corner.

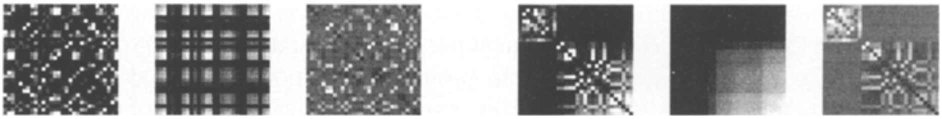

Fig. 3. (Left group of 3) The approximation of the affinity matrix obtained using p. Brighter pixels indicate values that are larger. (Left) The affinity matrix. (Center) The approximation of A provided by $p * p^T$. (Right) The difference between the two matrices. In the right group of three the same matrices are shown – the indices were permuted so that the vector p is sorted in ascending order.

2. For some i's p_i is exactly equal to zero.
3. The points for which $p > 0$ tend to cluster in one 'foreground' group.
4. The approximation of A appears to be good when at least one of the points involved belongs to the foreground. It is bad otherwise.

We may conclude with the key observation of this paper:
By calculating the pointwise approximation p_i of the pairwise affinity A_{ij} of the elements in the scene we have discovered a new global property: the partition of a scene into two groups, which we call 'foreground' and 'background'. We may see p as a 'saliency' function of the points.

After discovering the concept of foreground group we may specify an algorithm for calculating the foreground group. We call it the 'affinity factorization' algorithm:

1. Form a matrix A_{ij} containing the pairwise affinity of each pair of elements in the scene.
2. Call p the eigenvector of A that is associated to its largest eigenvalue.
3. Define the foreground F as the set of objects i whose corresponding p_i is not equal to zero (more practically, $p > \epsilon > 0$).

4 Approximation properties

Why does the first eigenvector of the affinity matrix A behave in the way we observed? We give here an explanation.

Lemma 1 Consider a symmetric nonnegative matrix B: $B_{ij} = B_{ji} \geq 0, \forall i, j$.
Then the eigenvector v_1^B of B that is associated to its largest eigenvalue is nonnegative: $v_1^B \geq 0$.

Proof It is well known that the iteration $v^{t+1} = Bv^t$ converges to v_1^B for almost all initial conditions v^0. Without loss of generality we initialize the computation with $v^0 > 0$. Since, by hyp., all entries of B are non-negative, then v^1 and therefore any v^k is also non-negative.

Lemma 2 Consider a symmetric block-diagonal $n \times n$ matrix C, whose b diagonal blocks B_i are nonnegative. Call $\lambda_j^{B_i}$ the eigenvalues of B_i and call $v_j^{B_i}$ the corresponding eigenvectors. Suppose that all such eigenvalues are distinct. Call $w_j^{B_i}$'s the n-vectors obtained by padding the $v_j^{B_i}$'s by zeros so that the nonzero entries correspond to the position taken by B_i within A.
Then the $w_j^{B_i}$'s are the eigenvectors of C and the $\lambda_j^{B_i}$ are the corresponding eigenvalues.

Proof By inspection.

Lemma 3 Consider C as above and call v_1 its eigenvector associated to its largest eigenvalue.
Then the entries of v_1 are zero corresponding to $b - 1$ of the blocks and are nonnegative corresponding to the remaining block.

Proof The largest eigenvalue of C is equal to the largest eigenvalue of one of the blocks B_i. The corresponding eigenvector therefore is nonnegative corresponding to that block and it zero elsewhere.

Proposition 2 Consider an affinity matrix A constructed on a scene where the affinity between elements of distinct groups is exactly zero. Then the first eigenvector of A is zero corresponding to elements in all groups minus one, and nonnegative corresponding to elements of that group.

Proof A is symmetric and nonnegative. By permutation $C = P^T AP$ it may be transformed into a block-diagonal matrix with nonnegative blocks, each block corresponding to a group in the scene. Its largest eigenvector may be obtained from the largest eigenvector of C by using the same permutation P.

Proposition 3 Consider the affinity matrix A of a scene where the affinity between elements of distinct groups has size of order ϵ.
Then the elements of the first eigenvector of A are nonnegative for one group and are of order ϵ for all other groups.

Proof Call v_i, λ_i the eigenvectors and eigenvalues of A. Write $A = A' + \epsilon N$ where N is a 'noise' matrix whose entries are distributed between zero and 1, and A' is block-diagonal. Write $v = v' + \delta v$ and $\lambda = \lambda' + \delta\lambda$, where v', λ' are the eigenvalues and eigenvectors of A'. Then $Av_1 = \lambda_1 v_1$. Multiply on the left by v_i, $i > 1$: $v_i^T A v_1 = \lambda_1 v_i^T v_1$. Expand A and v_1 and simplify using $A'v_1' = \lambda_1' v_1'$ obtaining:

$$v_i^t \delta v_1 = \frac{\epsilon}{\lambda_1 - \lambda_i} v_i^T N v_1 \tag{4}$$

i.e. the projection of the variation of v_1 due to the ϵ-sized noise in the direction of v_i is proportional to ϵ.

A comment stemming from the proof of proposition 3: when the magnitude of the 2nd eigenvalue of A approaches the magnitude of the first we expect the eigenvector corresponding to the largest eigenvalue to lose the attractive property of having zero entries corresponding to the background items.

5 Relationship with normalized-cuts grouping

5.1 Normalized cuts

Shi and Malik [8] have proposed to perform grouping using normalized cuts. They see the tokens that one wishes to group as the nodes of a graph, whose arcs carry the affinity between pairs of nodes. The cost of a cut $C(G_1, G_2)$ of the graph into two subgraphs is defined as the sum of the affinities associated with the arcs that are being cut. The affinity $V(G_1, G)$ between a subset G_1 of the graph and the rest of the graph G is the sum of all the arcs that connect the nodes in G_1 to every other node. They propose to minimize the normalized cut cost $N(G_1, G_2)$:

$$C(G_1,G_2) = \sum_{i \in G_1, j \in G_2} A_{i,j}, \quad V(G_k,G) = \sum_{i \in G_k, j \in G} A_{i,j}$$

$$N(G_1,G_2) = \frac{C(G_1,G_2)}{V(G_1,G)} + \frac{C(G_1,G_2)}{V(G_2,G)}, \quad G_1,G_2 = \arg\min_{G_1,G_2} N(G_1,G_2)$$

Shi and Malik develop a heuristic for calculating efficiently an approximation of the optimal partition. They solve the generalized eigenvector problem: $(A - D)x = \lambda Dx$, where A is the affinity matrix, x is an unknown vector, and D is a diagonal matrix with $D_{i,i} = \sum_j A_{i,j}$. The second-to-last eigenvector x turns out to be an indicator vector for the two sets G_1 and G_2: $i \in G_1 \iff x_i \geq \alpha$. Where α is a constant that is determined by solving a related optimization.

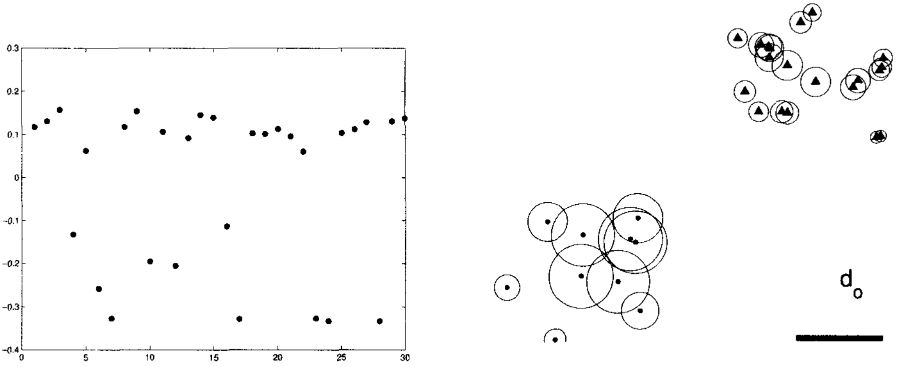

Fig. 4. The Normalized-Cut algorithm on the dataset shown in Fig. 1. On the left the value of the indicator eigenvector is shown (see text). The plot to the right, analog to the one in Fig. (2), shows the corresponding grouping – triangles indicate positive values of the eigenvector, dots indicate negative values, the diameter of the circles surrounding each marker indicates the magnitude.

5.2 Foreground cut

Consider an asymmetric variation of Shi and Malik's cost function. Define one of the two subsets of G to be a foreground F, and its complement $B = G \setminus F$ to be the background, and minimize:

$$N(F) = \frac{C(F,B)}{C(F,F)} \tag{5}$$

The problem may be solved along the lines of Shi and Malik. Define a vector x of zeros and ones, with the 1's being the foreground indices ($x_i = 1 \iff i \in F$).

Define a vector $\mathbf{1}$ as a vector of ones. Call B the background, complement of F: $B = G \setminus F$. Then:

$$N(F) = \frac{\sum_{i \in F, j \in B} A_{i,j}}{\sum_{i \in F, j \in F} A_{i,j}} \tag{6}$$

$$= \frac{x^T A(\mathbf{1} - x)}{x^T A x} = \frac{x^T A \mathbf{1}}{x^T A x} - 1 \tag{7}$$

Note that minimizing $N(F)$ is equivalent to minimizing $N(F) + 1 = \frac{x^T A \mathbf{1}}{x^T A x}$.

Now call (U, S, V) the SVD of A and notice that $U = V$ since A is symmetric. Define $z \doteq S^{\frac{1}{2}} U^T x$, and rewrite the cost function as:

$$N(F) + 1 = \frac{z^T S^{\frac{1}{2}} U^T \mathbf{1}}{||z||^2} = \frac{z^T u}{||z||^2} \tag{8}$$

$$\text{with} \quad u_i \doteq S^{\frac{1}{2}}(i,i) \sum_j U_{j,i} \tag{9}$$

Remember that we are trying to find a minimum of N subject to the constraint that x is composed of zeros and ones. It is not clear how to perform this optimization efficiently. We may proceed heuristically as in Shi and Malik by letting x take analog values, and imposing a constraint on its norm. The easiest such constraint, here, is to impose that $||z|| = ||x||_A = 1$. In this case we just need to minimize the numerator of our cost function. This is easily done by discovering the largest (in absolute value) entry of u, and picking $z^T = \pm[0, \ldots, 0, 1, 0, \ldots, 0]$ with 1 in the corresponding position k. The sign of z has to be chosen so that it is opposite to the sign of the maximum entry of u. Now remember that $x = S^{-\frac{1}{2}} U z$ - this means that the minimizing x is the k-th column of U appropriately scaled.

As a result we derive the following 'foreground cut' algorithm:

1. Calculate the SVD of A: $A = USV$.
2. Calculate the vector $u = SU\mathbf{1}$.
3. Determine the index k of the maximum entry of u.
4. Define the foreground vector x as the k-th column of U.
5. Threshold x, the foreground F correspond to the non-zero entries of x.

It turns out that the largest entry of u is typically the first, since, as previously discussed, if A is block-diagonal, then the 1st singular vector will have non-negative entries (it is the first eigenvector), while the other eigenvectors tend to have zero-mean; moreover, the sums of the columns of U enter in u weighted by the singular values in S, which imposed an additional bias in favor of low-index entries. Therefore, apart for rare cases, the 'foreground cut' algorithm gives identical results as the 'affinity factorization' algorithm.

6 Line affinity

The fact that line segments tend to group into longer line-like structures when they are roughly aligned in the image was noticed in the '30s by the Gestalt psychologists. Computer vision researchers have worked on the problem at least since the early 80s [3], with methods reaching a considerable level of sophistication by the end of the 80s [7, 5].

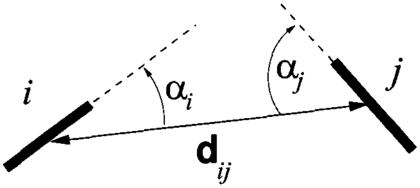

Fig. 5. Variables involved in the computation of the affinity of two edgels i and j.

A reasonable form for the affinity of line segments (or edgels), suggested by the observations of the Gestalt psychologists and supported by psychophysical measurements of Sagi, Kovacs, Braun and collaborators [2, 4, 1], assumes that that two lines are 'similar' when they are close by, when they are aligned and, failing that, when they are co-circular, i.e. tangent to the same circle. A possible implementation of these intuitions is (see also Fig. 5):

$$A_{i,j} = e^{-\frac{d_{ij}^2}{d_0^2} - \frac{2-\cos(2\alpha_i)-\cos(2\alpha_j)}{1-\cos(2\theta_0)} - \frac{1-\cos(2\alpha_i-2\alpha_j)}{1-\cos(2\delta\theta_0)}} \tag{10}$$

where the first term in the exponential is a distance-related affinity, the second term penalizes the average deviation of the line segments from being collinear, and the third term penalizes the non-cocircularity of the two line segments. The scaling constants $d_0, \theta_0, \delta\theta_0$ are somewhat arbitrary. Good values for d_0 range between the spacing of the elements and five times that value. Good values for θ_0 typically range from $\pi/2$ to $\pi/10$, while $\delta\theta_0$ typically should be half to one-fourth as large as θ_0 (see the experiments).

7 Computational complexity

The affinity factorization algorithm may be implemented efficiently. The cost function of Eq. 2 may be minimized, rather than computing the SVD of the matrix A, by gradient descent $p^{t+1} = (1-\lambda)p^t + \frac{\lambda}{||p^t||^2}Ap$, which for $\lambda = 1$ is:

$$p^0 = \mathbf{1}, \qquad p^{t+1} = \frac{1}{||p^t||^2}Ap \tag{11}$$

Typically 10-40 iterations of (11) achieve a stable result. If the computations are implemented using sparse matrices the total complexity is linked linearly to the number of neighbors of each element (this may be verified by inspecting (11)). Let N be the average number of neighbors of each element, and let E

be the number of elements in the set to be grouped, then the computational cost is $C \approx< 100 \times N \times E$ floating point operations for the minimization. This cost has the same order as that of the Shi-Malik algorithm with a multiplicative constant that is roughly half as large. The computation of the matrix A may also be reduced to a linear cost by using quad-trees for referencing the data points (we have not implemented this). This appears to be faster than both Shashua and Ullman [7] and Parent and Zucker [5], although we have not yet implemented their algorithms and we were unable to find a precise assessment of computational complexity in their papers.

8 Experiments

We explore some of the characteristics of the affinity factorization (or, equivalently the foreground cut) and the normalized cuts algorithms. In our experiments we have picked thresholds by hand – the problem of automatic threshold selection is discussed in [8]; in some experiments, where we thought that this may be illuminating, we used multiple thresholds. Another issue that we do not discuss here is the extraction of multiple groups from the image. This is also discussed in [8]. Here we will restrict ourselves to showing the most salient group found by the affinity factorization algorithm, and the most salient cut found by the normalized cuts algorithm.

The method we used for calculating p in the affinity factorization algorithm is specified in Eq. (11); the calculation of x in the normalized-cuts algorithm was performed using Matlab's SVD function. The constants used in the experiments are specified in the figures.

8.1 Experiment 1: Point Clusters

A first experiment is shown in Figures 1, 2, 3, 4. A second experiment using data from [8] is shown in Fig. 6. Both algorithms achieve good grouping.

8.2 Experiment 2: Foreground-background

A second experiment compares the algorithms on a foreground-background display shown in Fig 7, left. See Figures 8, 9. The normalized cuts algorithm does not achieve grouping for any value of p, probably due to it's symmetric formulation, where both sets in which the scene is grouped are supposed to be 'meaningful'. The affinity factorization algorithm does achieve grouping instead.

8.3 Line grouping

We used the affinity function reported in Sec. 6 to group lines in the line figure shown in Fig 7, right. The affinity factorization algorithm achieves good grouping, while the normalized-cuts algorithm does not, probably due to the foreground-background problem mentioned above.

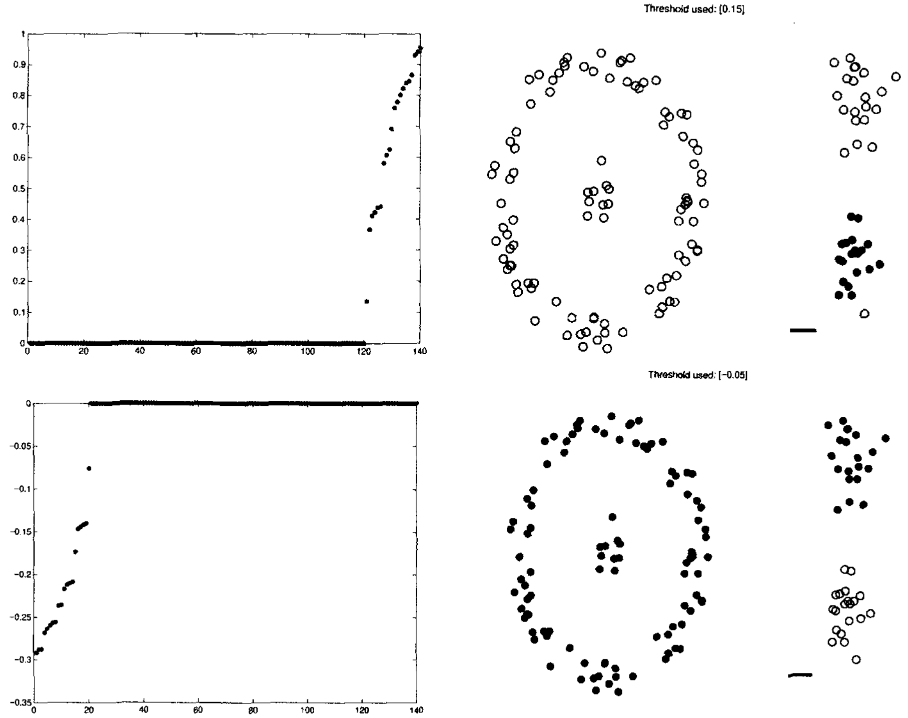

Fig. 6. (Top) Grouping using the affinity factorization algorithm. (Bottom) Using Shi-Malik's normalized cuts.

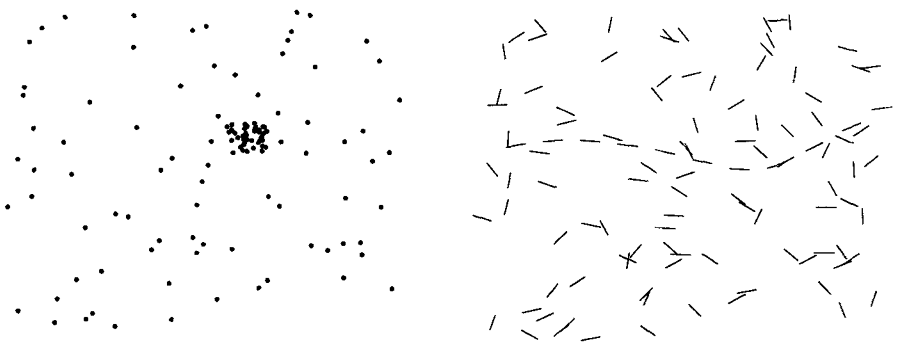

Fig. 7. (Left) Foreground/background point figure. (Right) Line segments on a plane forming an 'emergent line'.

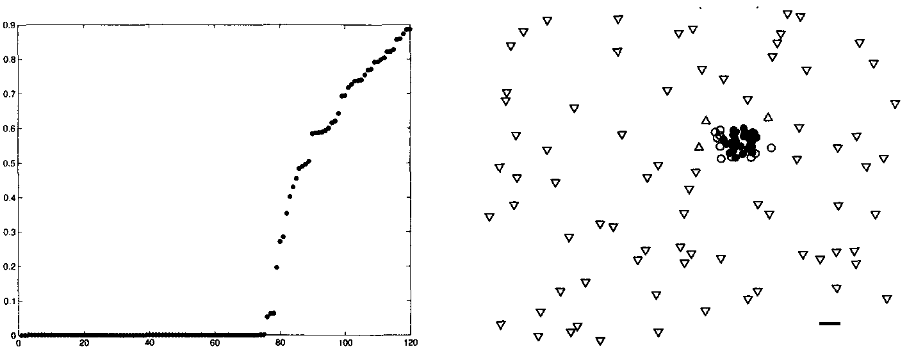

Fig. 8. The 'groupiness' vector p (left) and the groups (right) obtained for a foreground-background dot figure using the affinity factorization algorithm. The short bar at the bottom right indicates the length of the critical distance d_0. Capsized triangles indicate points for which $p < 0.01$; upright triangles indicate points for which $p \in (0.01, 0.1)$; open circles indicate dots for which $p \in (0.1, 0.5)$; closed circles are associated to $p > 0.1$. Notice that the circles indicate the foreground group, with open circles at its periphery and 3 upright triangles at the extreme periphery.

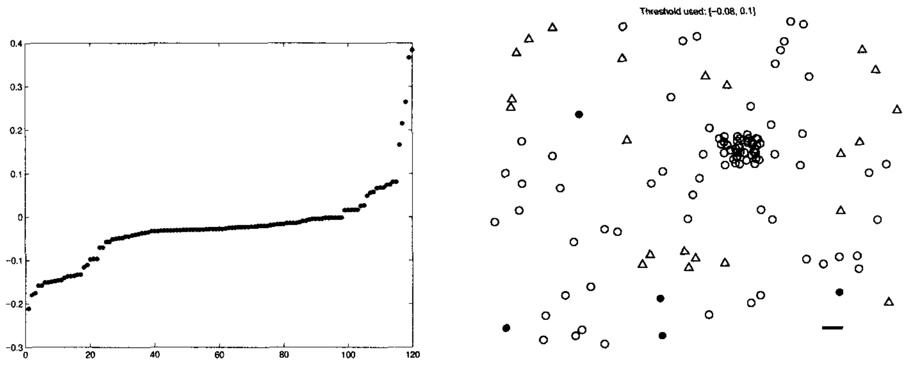

Fig. 9. Performance of the normalized cuts algorithm on same data as in Fig. 8. The indicator eigenvector x of A is shown in the plot on the left. Unlike the vector p of the affinity factorization algorithm, x takes both positive and negative values. The grouping is shown on the right, with triangles indicating $x < -0.08$, open circles indicating $x \in (-0.08, 0.1)$ and closed circles $x > 0.1$. The symbols are mixed and therefore no good groups emerge.

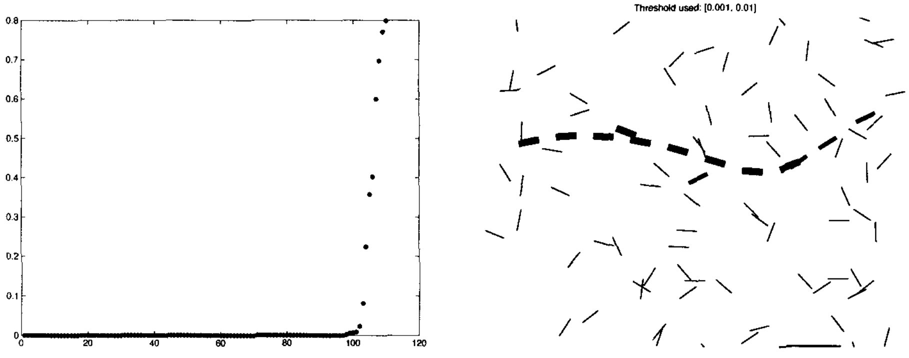

Fig. 10. Grouping performed by the affinity factorization algorithm.The thickest line segments correspond to $p > 0.01$, the medium-thick segments correspond to $p \in (0.0001, 0.01)$, while the thin segments correspond to $p < 0.0001$. The long line at the bottom right indicates the critical distance d_0. The constants used in (10) were $\theta_0 = \pi/8$ and $\delta\theta_0 = \pi/16$.

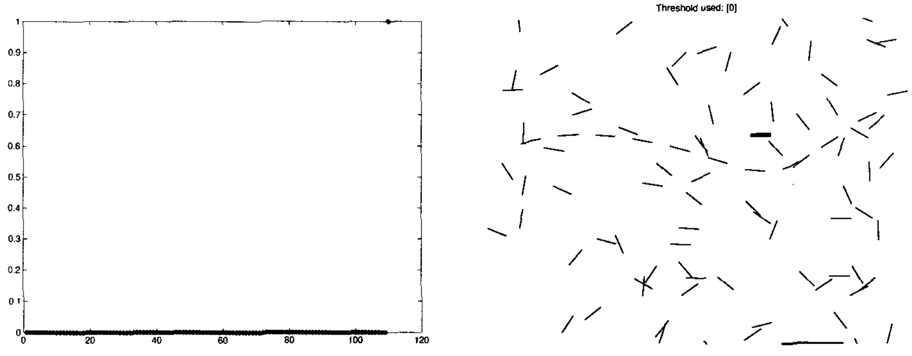

Fig. 11. Grouping performed by normalized cut algorithm. The same affinity matrix as in (10) was used (i.e., same affinity function and same constants). The vector x only has one non-zero value (corresponding to the short bold line, roughly at the center of the display).

8.4 2D images

We explored the behavior of the algorithms on the synthetic brightness images shown in Fig. 12. The images are identical apart from the background (relatively noisy or 'unstructured' in the right image).

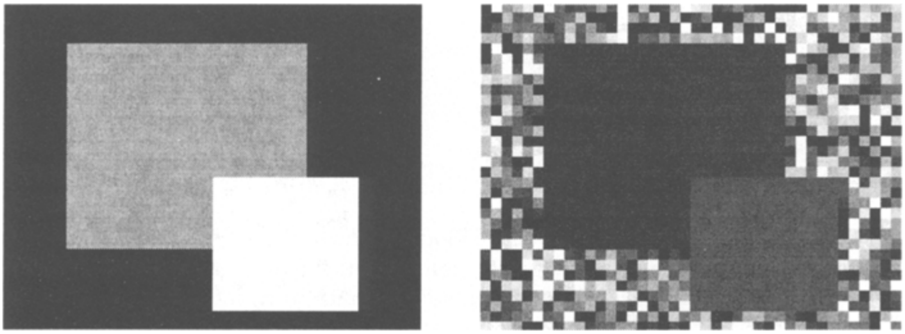

Fig. 12. Two brightness 'Mondrians' used in our experiments. The brightness values are uniformly distributed between 0 and 0.1 for the background of the left image and between 0 and 10 for the background of the right image. The rectangles are identical in the two images, with brightness uniformly distributed between 2 and 2.1 for the larger one and between 3 and 3.3 for the smaller one. The units of brightness are arbitrary. The images are shown with different colormaps.

An exponential affinity function was used (i and j are pixel indices):

$$A_{ij} = \exp^{-\frac{d^2(i,j)}{d_0^2} - \frac{(I(i)-I(j))^2}{dI_0}} \tag{12}$$

with I indicating brightness values and constant values $d_o = 3$ and $dI_o = 1$.

In Fig. 13 the results of the experiment are shown. Both algorithms work well on the first Mondrian (Fig. 13, first row), with the normalized cuts algorithm showing a more uniform value than the factorization algorithm (although, as shown in the middle column, the log of the first singular vector shows equivalently sharp boundaries). In the Mondrian with unstructured background (Fig. 13, second row) the normalized cuts algorithms groups a small cluster of pixels rather than one of the two large rectangles.

9 Discussion and Conclusions

We have shown that concepts such as 'grouping' and 'foreground' may be obtained from an intuitive concept of 'pairwise similarity' or 'pairwise consistency' between elements of the scene. The derivation consists in choosing to approximate the pairwise relationships between all elements with a pointwise property

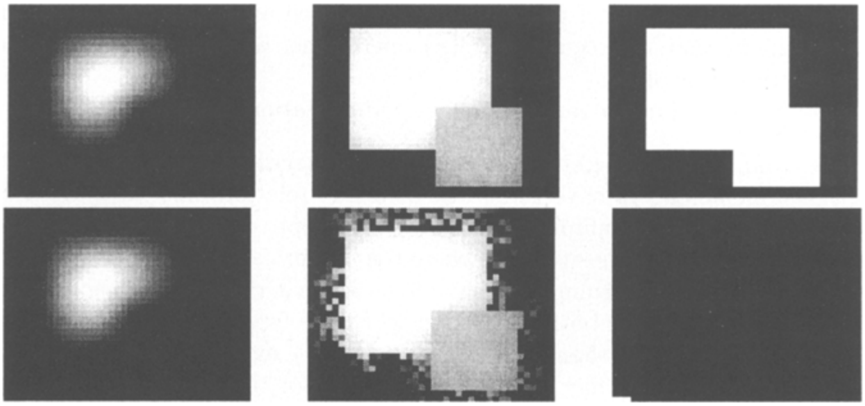

Fig. 13. First eigenvector of the affinity matrix (left), log of the same (center) and normalized cuts (right). The first row shows the results on the first Mondrian, the second on the second.

of each element. It turns out that the pointwise property that best approximates pairwise similarity may be interpreted as a 'saliency' measure, and one may obtain the foreground group in the scene by thresholding this saliency measure. Propositions 2 and 3 give an explanation, based on the properties of the eigenvectors of (noisy) block-diagonal matrices, of why this is the case.

Our derivation suggests an algorithm for grouping which we call 'affinity factorization algorithm'. We derive a second grouping algorithm which we call 'foreground cuts' by modifying the 'normalized cuts' formulation of Shi and Malik [8], and we show that 'foreground cuts' and 'affinity factorization' are essentially the same algorithm, although they are derived from a different starting point.

We compare affinity factorization and normalized cuts on a number of test sets of points and lines and brightness images. The experiments demonstrate the good grouping performance of affinity factorization even on data sets where the background is unstructured, which the normalized cuts algorithm is not constructed to handle. Malik and collaborators approach this problem by defining more sophisticated affinity measures, including texture and brightness histograms, for which the class of unstructured image patches is smaller (see their papers in the present proceedings and of ICCV 1998).

The experiment on brightness Mondrians shows that value of the saliency vector p of the affinity factorization algorithm drops near the boundaries of the foreground region (compare with the very desirable piecewise constant behavior of the normalized cuts vector). However, its logarithm shows sharp boundary discontinuities (it is natural to consider the log. since we are guaranteed that p is non-negative). This suggests that maybe the log of the largest eigenvector of the affinity matrix should be considered as the proper definition of the affinity function.

The implementation of the affinity factorization algorithm that we propose is efficient, in that it executes in $O(n)$ operations, where n is the number of elements in the display.

Directions for further research and experimentation include:

1. Generalization to the situation in which, rather than pairwise affinities between elements, we have three-way or n-way relationships. In that case we may start from the affinity tensor A_{ijk} and approximate it with $p_i p_j p_k$.
2. Generalization to the situation where the affinity is non-symmetric (think of the line-elements forming a T junction). In that case we may approximate A with $p_i q_j$ and use both p and q to select the foreground set.
3. Study other norms than L^2. The L^1 norm, for example, may be more appropriate. Eq. (11) may be suitably modified for this purpose.
4. Incorporate 'top-down attention' by imposing that the solution contains a certain subset of elements. This may be done by introducing a bias term in Eq. (11).

Acknowledgements

Conversations with Allen Tannenbaum, Xiaolin Feng, Max Welling, Sandro Zampieri, Jitendra Malik, Janbo Shi, Serge Belongie, Thomas Leung and Josh Tannenbaum are gratefully acknowledged. This research was in part supported by the NSF Engineering Research Center in Neuromorphic Systems Engineering at Caltech.

References

1. Y Adini, D Sagi, and M Tsodyks. Excitatory-inhibitory network in the visual cortex: psychophysical evidence. *Proc. Natl Academy Sci.*, 94(19):10426–10431, September 1997.
2. J. Braun and E. Niebur. Perceptual contour completion: a model based on local anisotropic, fast-adapting interactions between oriented filters. *Soc. Neurosci. Abstract*, 20(1665), 1994.
3. M. Hedlund, G. H. Granlund, and H. Knutsson. A consistency operation for line and curve enhancement. In *Proc. IEEE Comp. Soc. Conf. on Pattern Recognition and Image Processing*, pages 93–96, 1982.
4. I Kovacs. Gestalten of today: early processing of visual contours and surfaces. *Behavioural brain research*, 82:1–11, 1996.
5. P. Parent and S. Zucker. Trace inference, curvature consistency, and curve detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(8):823–839, 1989.
6. A. Pinkus. *n-Widths in Approximation Theory*. Springer Verlag, 1985.
7. A. Shashua and S. Ullman. Structural saliency: the detection of globally salient structures using a locally connected network. In *2nd International Conference on Computer Vision (ICCV)*, pages 321–327, December 1988.
8. J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, Puerto Rico, June 1997.