Fast Multilevel Clustering

Ya'ara Goldschmidt[1] **Meirav Galun**[1] **Eitan Sharon**[2] **Ronen Basri**[1] **Achi Brandt**[1]
[1]Dept. of Computer Science and Applied Math.
The Weizmann Institute of Science, Rehovot 76100, Israel
[2]Computer Science Division, EECS, U.C. Berkeley, Berkeley, CA 94720
Email for correspondence: yaarag@weizmann.ac.il

Abstract

Clustering is a difficult problem. Clustering data may differ by a variety of aspects (dimensionality, cluster size, noise, etc), and the criterion for clustering may depend on the context in which the data is given. We present a multilevel approach for clustering, easily adaptable to handle various kinds of data by identifying desired underlying features of the data. The scheme we present is given a similarity graph, on which we apply a recursive coarsening process, resulting in a pyramid of graphs in time that is linear in the number of edges in the graph. The pyramid provides a hierarchal decomposition of the data into clusters in all resolutions, and data points are associated to clusters with soft relations. We demonstrate the algorithm by applying it successfully to challenging clustering problems.

1 Introduction

Clustering is the grouping of similar objects together. While the notion of clustering is intuitively clear, there is no widely accepted definition for the task of clustering, and the definition may vary between types of data sets, and between algorithms. Clustering is difficult for a number of reasons. Real-life data may contain clusters of varying size and shape, whose number is unknown in advance. Noise and outliers can further complicate the task by connecting separate clusters. Two examples of such clustering data are shown in Fig. 1.

A large variety of clustering algorithms have been proposed. Most common methods provide high quality results on a large variety of data sets. Nevertheless, these methods often return unsatisfactory results when applied to complex, real-life examples. Some methods, such as k-means [1] can largely only handle clusters of spherical shape. Other methods, such as average Linkage [2] and spectral methods [3, 4] can better adapt to clusters of different shapes, but they are more sensitive to noise and outliers, particularly when the amount of noise is significant. This paper introduces a clustering scheme that addresses these problems. This algorithm is easily adaptable to handle different types of data sets and different clustering objectives, and so it successfully solves such complex data sets, see Fig. 1. We present a multilevel graph algorithm for clustering. Given a set of data points and a distance function between the points, we begin by constructing a similarity graph, in which each data point is a node, and each pair of nodes are connected by an edge. We assign weights to the edges, reflecting the proximity between the points. We further

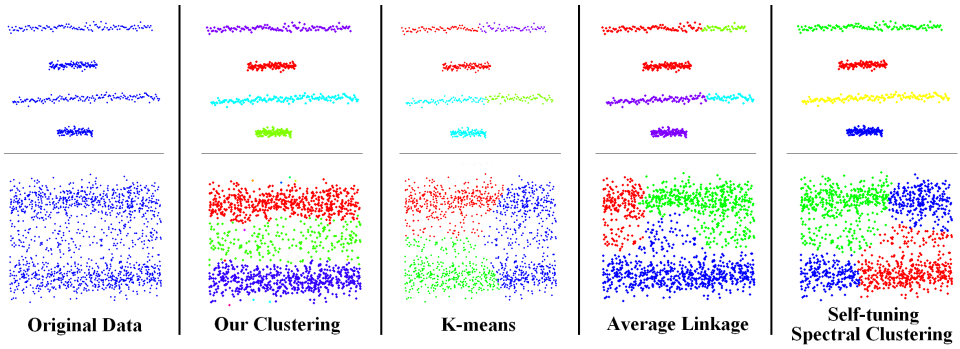


Figure 1: Comparison of our results and other algorithms. The top row is a relatively easy example, while on the bottom is a more challenging noisy example of three horizontal clouds of changing densities. From left to right are the original points, our result, k-means, average linkage and self-tuning spectral clustering.

bound the degree of this perhaps fully connected graph by applying a k nearest neighbors pre-processing step. Clustering is then defined as finding minimum normalized cut in this graph [5, 6]. We solve this minimization problem by applying a sequence of coarsening iterations, where at each iteration a smaller graph is generated. This process yields a hierarchy of clusters in all resolutions, where clusters can overlap, and a data point can belong to several clusters with soft association weights. The coarsening is adaptable to the data and is designed to faithfully represent the problem with fewer variables, producing an irregular pyramid of graphs in computing time that is linear in the number of edges. This algorithm is inspired by algebraic multigrid (AMG) solvers [7, 8, 9] and by the weighted aggregation algorithm for image segmentation (SWA) [10].

2 SWA Review

The SWA algorithm is a multiscale graph partitioning algorithm. Given an image, it construct a graph $G = (V, W)$, with nodes in V representing image pixels and the symmetric edge weight matrix W represents the similarity in intensity between adjacent pixels.

A segment S is evaluated by a saliency measure $\Gamma(S) = \frac{\sum_{i<j} w_{ij}(u_i - u_j)^2}{\sum_{i<j} w_{ij} u_i u_j}$, where u_i is one if node $i \in S$, and zero otherwise. The saliency measure sums the weights along the boundaries of S normalized by the sum of internal weights. Segments that yield small values of Γ are considered salient (excluding $S = V$ and single point partitions).

Allowing arbitrary real assignments to u, the minimum for Γ is obtained by the minimal generalized eigenvector u [11], which is in fact equivalent to the normalized cut solution in [6]. Since this task is intractable with the constraint of discrete assignments of u [6], SWA approximates the solution invoking a recursive multilevel process, inspired by AMG solvers for the eigen problem.

The multilevel framework automatically constructs a hierarchy of decreasing size graphs: $G^{[0]}, G^{[1]}, \ldots, G^{[S]}$. At each level, the next smaller graph is created by *coarsening* the previous one. The coarsest level ($G^{[S]}$) is solved directly, and finally the solution is uncoarsened back to the initial graph ($G^{[0]}$) to yield the graph partitioning. The coarsening is a process of *weighted aggregation* of the graph nodes to define the nodes of the next coarser graph. The construction of a coarse graph from a given one is divided into three stages: First, a subset of the fine nodes is chosen to serve as the *seeds* of the aggregates

(coarse graph nodes). Then the rules for *interpolation* are determined, thereby establishing the fraction of each non-seed node belonging to each aggregate, so that an aggregate represents a soft partition of fine nodes. Last, the coupling weights of the edges between the coarse nodes are calculated. This results in an irregular pyramid of graphs. Each desired segment emerges as a node in the pyramid, and nodes at different levels represent segments in different resolutions. The un-coarsening is done for each coarse segment to identify the elements of the finest level that associate to it through the interpolation weights, and thus a soft relation is obtained between the image pixels and the nodes at each level. We use this multilevel framework for clustering as follows.

3 The Algorithm: Adaptive Multilevel Clustering

3.1 The Similarity Graph $G^{[0]}$

The process starts with the graph $G^{[0]} = (V^{[0]}, W^{[0]})$ called the ***finest graph***, where each data point is a node. Each node is attached with an edge to each of its k nearest points, provided that the node is among its neighbor's k nearest neighbors, so that the connectivity degree is bounded by k. The similarity graph is built by a pre-processing step using any nearest neighbors approximation approach such as [12].

The initial weights of this graph are a function of the similarity between data points, and adjusted to the type of data in hand. For example, in SWA the resemblance between neighboring pixels is measured by the difference in their intensities, so the weights used are $w_{ij} = e^{-\alpha|I_i - I_j|}$, where I_i and I_j are the intensities of adjacent pixels i and j, and α is a scaling parameter. In sets of genes, two genes are considered to relate to the same biological pathway if their expression patterns correlate over time or a set of conditions, so their edge may have the weight $w_{ij} = |corr_{ij}|^{\alpha}$, where $corr$ is the Pearson correlation coefficient between the genes' expression vectors. Yet another example is of data points in an $\mathbb{R}^n$ Euclidian space, where d is the Euclidian distance, and weights can be $w_{ij} = 1/({d_{ij}}^{\alpha})$, α is a scaling parameter, enlarged to emphasis smaller distances. In all the experiments presented here, we have used the latter definition.

3.2 Adaptive Weighted Aggregation by Coarse Level measures

The first step in creating a coarse graph from an existing one is choosing the coarse graph nodes from the set of the fine nodes. For this, we define an *aggregation weight* for each edge ij denoted by a_{ij}, which is a function of the coupling weights w_{ij}. The aggregation weights are used to choose the coarse nodes and establish the relations between the coarse and fine nodes. The simplest definition of the aggregation weights is by $a_{ij} = w_{ij}$, and we will present other constructions later, suitable to different clustering problems. The criterion for choosing the coarse nodes also depends on this choice of aggregation weights and will be elaborated on later as well (see Sec.3.3).

The set of coarse nodes C and its complement, denoted by F are constructed so that each F-node is "strongly coupled" by the aggregation weights to the nodes in C. As a result, the coarse graph represents the fine graph with respect to the clustering objective.

Each node k in the chosen set C becomes the seed of an aggregate that will constitute the $J(k)$-th, say, coarse level node (that is, $J(k)$ denotes its ordinal number among the nodes of the coarse level). The classical AMG interpolation matrix P (of size $|V| \times |C|$) is defined by

$$p_{iJ(k)} = \begin{cases} a_{ik} / \sum_{l \in C} a_{il}, & \text{for } i \in F \\ 1, & \text{for } i = k \\ 0, & \text{for } i \in C,\, i \neq k. \end{cases} \quad (1)$$

$p_{iJ(k)}$ represents the likelihood of node i to belong to the k-th aggregate, and $\sum_k p_{iJ(k)} = 1$. For simplicity of presentation we denote $p_{iJ(k)}$ by p_{ik}. It is also advisable to remove very small interpolation weights (typically $p < 0.2$), to reduce the complexity of the computations, but for not more than 20% of interpolations per node, to prevent loss of information. (After any removal of interpolation weights, they are re-normalized so as to satisfy $\sum_k p_{ik} = 1$ for each $i \in V$).

The final stage sets the edge weights of the coarse graph. For two coarse nodes k and l, their coupling weight is set by $w_{kl} = \sum_{i<j} p_{ik} w_{ij} p_{jl}$, (where w_{ij} is a weight in the fine graph and w_{kl} in the coarse).

The aggregation weights affect the choice of seeds and the interpolation weights. When defining $a_{ij} = w_{ij}$ the above procedure is adequate for many problems, like in image segmentation [10, 13], linear ordering problems [14] and simple clustering problems when the clusters are well separated. However, for more complex problems such as texture segmentation, or as in Fig. 1, where clusters are separated by areas of vast noise and outliers, relying only on the coupling weights does not yield the desired result as seen in Fig. 2, since the couplings are based only on the distance between points and do not hold information about other properties such as density or shape.

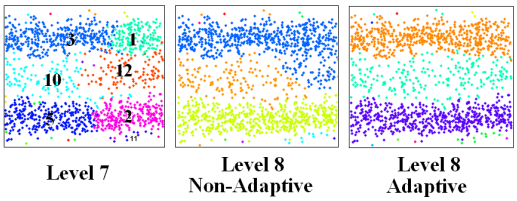
We overcome such situation by calculating specific properties for nodes and edges, based on existing pre-knowledge of the data. We then incorporate these properties into the aggregation weights. In many multilevel applications different properties are often used in various ways, to gather relevant statistics on emerging aggregates, and are called *coarse measures*, e.g. in texture segmentation [11] and in Manifold Identification in 2D and 3D [15]. Here, we incorporate coarse measures into the aggregation weights to allow the correct clustering already during the bottom-up process. Although the choice of what to incorporate into the algorithm is done in a supervised way, the aggregation itself is unsupervised.

Coarse node measures are initialized at the finest level, and recursively aggregated per node through the interpolation weights without increasing the complexity. For example, the volume of node v_i in the finest level is one, and at coarse levels is calculated by $v_k = \sum_{i \in V} p_{ik} v_i$.

First we give some motivation for the coarse measures we use. Since noise often manifests in sparser neighborhoods than the underlying "real" data, we would like for points in sparse neighborhoods to more likely be chosen as coarse, until all other denser areas have finished their aggregation. To control this we calculate the density of a node as a coarse level measure, and weaken the weight between nodes whose density differs. Moreover, distant clusters become strongly coupled through stretches of noise between them and wrongly aggregate together, as is often the case in other algorithms. We would weaken this effect to allow segments of a cluster to aggregate together before the noisy patches attach to them and corrupt the aggregation. This is done by calculating the "true" distance between clusters, not influenced by the sparse noise between them. We can so overcome even extreme noise between overlapping clusters. Fig. 2 shows the effect with and without the use of adaptive aggregation weights.

To capture the feature of density of emerging nodes, we define the coarse node measure of *sparsity*. The sparseness of a node in the finest graph is its *typical average distance* (first used in [15]) given by $\rho_i = < d_{ij} >$; that is, the mean of the distances to its nearest neighboring points (we use 6 neighbors in the experiments shown). This measure resembles the inverse of the local density, in the sense that lower ρ implies lower local sparsity, i.e. larger density.

At coarser levels we define the sparseness of a coarse node k by $\rho_k = \frac{\sum_i p_{ik} \rho_i}{\sum_i p_{ik}}$, where i is a fine node connected to k through the interpolation p_{ik}.



Weights of node #12

	1	2	10
coupling weights	**398.8**	136.9	263.2
adaptive weights	121.7	44.77	**252**

Figure 2: The effect on the aggregation with and without the adaptive aggregation weights. On the left is the clustering at level 7. Notice cluster number 12, its coupling and aggregation weights to nodes 1,2 and 10 are given in the table on the right and explain the results achieved when using each of them. In the middle is the result using the coupling weights, which undesirably aggregate node 12 to 1, and on the right is the correct result, obtained by using the adaptive weights defined in Eq. 3.

Next, we define a *coarse edge measure*, which will allow measuring distances between clusters at coarse levels. The *Common Border* denoted by c: At the finest level, for every two nodes i and j that share an edge, their border is $c_{ij} = 1$. For two coarse nodes k and l sharing an edge we define $c_{kl} = \sum_{i<j} p_{ik} c_{ij} p_{jl}$, representing the weight of the border between the nodes.
Normalized Couplings denoted by $\bar{w}$: At the finest level $\bar{w} = w$, and between two coarse nodes k and l, $\bar{w}_{kl}$ is calculated by

$$\bar{w}_{kl} = \left(\frac{\sum_{i<j} p_{ik} (w_{ij}^{\xi}) p_{jl}}{\sum_{i<j} p_{ik} p_{jl}} \right)^{\frac{1}{\xi}}, \tag{2}$$

where i and j are fine nodes, and $\xi > 0$ is a parameter. This is actually a normalization of the coupling weights w. It can be controlled so that a large number of small valued interpolations would not overtake the coupling value: their effect is weakened by increasing ξ. Thus $\bar{w}$ reflects the true relations between coarse nodes. Notice that the inverse of $\bar{w}$ corresponds to the distance between sub-aggregates. However, it does not always represent the real distance between aggregates, since it may be biased by noise. To measure the real distance we define another coarse edge measure denoted by $\hat{d}$. It is similar to the previous measure but normalized by the common border, to prevent patches of noise to bias the distance, $\hat{d}_{kl} = \left(\frac{\sum_{i<j} p_{ik} (w_{ij}^{\xi}) p_{jl}}{c_{kl}} \right)^{-\frac{1}{\xi}}$.

The three above aggregative edge measures and the sparseness property of a node are combined to define the *aggregation weights*, denoted by a,

$$a_{kl} = \bar{w}_{kl}^{\alpha} \cdot e^{-\gamma_1 \frac{|\rho_k - \rho_l|}{\rho_k + \rho_l}} \cdot e^{-\gamma_2 \frac{\hat{d}_{kl}}{(\rho_k + \rho_l)}} \tag{3}$$

α, γ_1 and γ_2 are parameters. In words, a_{kl} is a function of the normalized coupling between nodes, weakened by two factors. The first exponent weakens the weight if the nodes' ρ differs, and the second exponent weakens the weight if their global distance is large compared to their difference in ρ. α is set as before to re-scale the aggregated distances if necessary, emphasizing smaller distances (and larger couplings). γ_1 is enlarged in order to capture even small changes in density, and γ_2 is enlarged to strengthen the effect of the distance on the aggregation weight. Consequently, the aggregation weights would be larger for adjacent nodes with similar density, enforcing them to aggregate first by the procedure in the following section.

The adaptive weighted aggregation can be adjusted to follow other properties of the data by calculating the relevant coarse measures, and modifying the aggregation weights accordingly.

3.3 Choosing the coarse nodes

We use the aggregation weights to choose the seeds as follows. We start with an empty set C, hence $F = V$, and then sequentially transfer nodes from F to C. Nodes with small ρ (high density) are considered first. We transfer nodes until all remaining $i \in F$ satisfy

$$\sum_{\substack{j \in C \\ \bar{w}_{ij} > \delta_s}} a_{ij} / \sum_{\substack{j \in V \\ \bar{w}_{ij} > \delta_s}} a_{ij} \geq Q. \tag{4}$$

where δ is a threshold dependent on the level s. For simple data where clusters are expected to be well separated, and weights inside a cluster much stronger than between clusters, we can consider all the edges, and set $\delta_s = 0$. For noisy data however, we let δ_0 be a user defined parameter, and reduce it by ϕ at each level, so $\delta_s = \delta_0/\phi^s$ up to some predefined level s'. For $s > s'$ we set $\delta_s = 0$, so that from this level on all edges are considered. Thus only relatively strong weights are used at the low levels to determine the seeds. As a result, in low levels the sparse nodes will have a smaller fraction on the left hand side of Eq. (4), and will be more inclined to be chosen as coarse, instead of attaching themselves prematurely to the denser aggregates and corrupt the entire aggregation. A future approach to control δ automatically is to pre-process the weights of the original graph into a histogram, and at each level define δ so as to include increasing number of bins of the histogram.

3.4 Controlling the Aggregation Order

For data of clusters of different sizes, the small ones often finish their aggregation first, then attach prematurely to others, corrupting the clustering result. We can detect this situation noticing that an object has not finished its aggregation if it still has relatively strong couplings (or aggregation weights) to other nodes.

To identify this, we look at every node $j \in F$ (i.e. nodes that where **not** chosen as coarse), and examine its interpolation values p_{jk}. High interpolation indicates premature attachment to a coarse node only if that coarse node has much stronger connection to any other node. So for any $p_{jk} \geq \tau$, if there exists $l \in C, l \neq k$ such that $a_{lk} \gg a_{jk}$ (e.g. 10 fold) then p_{jk} is replaced by 0 (i.e., removed). After all the relevant interpolation weights have been removed, the rest are re-normalized such that $\sum_k p_{jk} = 1$ for every j. In addition, if by the end of this process a node remains disconnected to all coarse nodes, it becomes coarse itself. τ was typically 0.5 in the experiments shown in the results section. The process can be repeated in iterations, but in the reported experiments one iteration, and only in each of the few last coarse levels, suffices. In any case, a final iteration determines the final interpolation weights also to the new coarse nodes.

3.5 Aggregative Interpolation

This operation looks at the interpolation weight from a certain coarse node as determined by the entire corresponding aggregate, not just by its seed. The goal is to correct the situation where a fine node finds itself in a different aggregate than most of its coupled neighbors. Thus we allow for nodes to strengthen or weaken their affinity to coarse nodes based on the majority decision of their neighborhood. This is achieved by calculating a new interpolation weight ${p_{ik}}^{new} = \frac{\sum_j w_{ij} p_{jk}}{\sum_l \sum_j w_{ij} p_{jl}}$, where j runs over all fine nodes coupled to i, and l over all coarse nodes interpolated to i. We run this process in iterations, usually two iterations per level suffice.

3.6 The Final Clustering Solution

The results of the aggregation process is an irregular pyramid of graphs. Each desired cluster will emerge as a node in the pyramid, and nodes at different levels represent clusters at different resolutions. For each desired cluster we identify the elements of the finest level that associate to it through the interpolation weights by a single top-down sweep of the pyramid, and thus obtain a soft (fuzzy) relation between the original elements and the clusters.

4 Complexity

Given a distance matrix as input, the bottom-up aggregation algorithm is linear in the number of edges. The number of edges depends on the number of neighbors (k) we define for each node, and so we bound the number of neighbors per node by $k = 0.1N$ (10% of the size of the data), and usually use much less ($k = 10$). Calculating the nearest neighbors may suffer from the curse of dimensionality, and can be approximated by several method (e.g. [12] in $O(k \cdot dim \cdot logN)$ time per node).

5 Experiments

We tested the clustering on several data sets, including synthetically generated data and real-life data. Given the bounded similarity graph, the typical running time for a graph of 7700 nodes of bounded degree 20 is less than a minute using an un-optimized code.

Iris Data This data set due to Fisher (1936) (obtained from UCI [16]), contains 3 classes of iris plants, of 50 instances each. One class is linearly separable from the other two, while the others aren't. Dubnov et al. [17] correctly classify 124 samples, and compare their results to Blatt et al. [18] who classified correctly 125 samples by the Super-Paramagnetic Clustering (SPC) algorithm, and report that their result is the second best after the Minimum Spanning Tree (MST) algorithm that achieves perfect classification on this data.

We correctly classify 146 of the data points using only the non-adaptive coupling weights. This makes our result the second best after the MST. To determine the final classification from our clustering algorithm, we assign a sample to the cluster whose interpolation value to it exceeds 0.5. At the coarsest level we get two clusters, one of all the iris-setosa plants, and the other of all the rest, which decomposes at the next level into two clusters, each of 48 plants of the same kind, and two plants of the other kind.

Hierarchical clustering of Image Silhouettes We received the database of silhouettes of objects, and the dissimilarity between every two objects from Gdalyahu and Weinshall [19]. The data contains 90 silhouettes of 6 different objects (toy, cow, wolf, hippopotamus, 2 cars and a child) each at 15 different tilts and pans. The dendrogram on the right displays our clustering result. All objects were correctly classified at level three, and aggregated into 3 clusters at the top level: the group of animal, the cars, and the children.

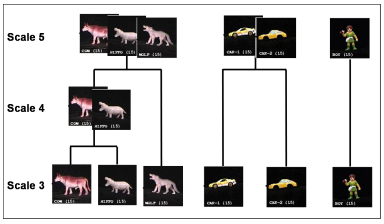


Figure 3: Results as a dendrogram.

2D clouds of changing densities and noise We randomly generated several synthetic data sets. Each is a sampling of points in 2D from specific distributions, allowing to test the performance of the algorithm in cases of changing densities, noise and different geometric shapes in space. Figure 4 summarizes the results.

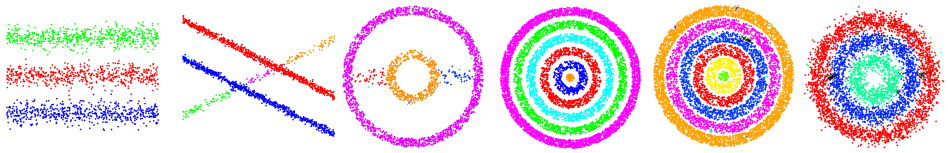


Figure 4: Our clustering results on different shapes and increasing difficulty. From left to right: 600 pts with similar densities. 800 pts in crossing lines differ by their density. 2400 pts of two circles connected by sparser filaments. 7700 pts uniformly distributed around different radii. 7700 pts of closer circles. 4800 pts normally distributed $N(r, 0.25)$ around $r = 1, 2, 3$. Each cluster is drawn with a different color, and black $'\times'$ marks points whose association is not determined since they belong equally to more than one cluster.

References

[1] J. Hartigan and M. Wong. Algotihm as136: A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[2] R.R. Sokal and C.D. Michener. A statistical method for evaluating systematic relationships. *The Uni. of Kansas Sci. Bull.*, 38:1409–1438, 158.

[3] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS 14*, 2001.

[4] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *NIPS 17*. 2005.

[5] A.K. Jain and R.C. Dubes. *Algorithms for clustering data*. Prentice-Hall, 1988.

[6] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[7] A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations. Technical report, Fort Colins, Colorado, 1982.

[8] A. Brandt. Algebraic multigrid theory: The symmetric case. *Appl. Math. Comput.*, 19(1-4):23–56, 1986.

[9] K. Stüben. A review of algebraic multigrid. *J. Comput. Appl. Math.*, 128(1-2):281–309, 2001.

[10] E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. *CVPR*, I:70–77, 2000.

[11] M. Galun, E. Sharon, R. Basri, and A. Brandt. Texture segmentation by multiscale aggregation of filter responses and shape elements. In *ICCV*, pages 716–723, 2003.

[12] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. of the ACM*, 45(6):891–923, 1998.

[13] E. Sharon, A. Brandt, and R. Basri. Segmentation and boundary detection using multiscale intensity measurements. *CVPR*, I:469–476, 2001.

[14] I. Safro, D. Ron, and A. Brandt. Graph minimum linear arrangement by multilevel weighted edge contractions. *Journal of Algorithms*, in press.

[15] D. Kushnir, M. Galun, and A. Brandt. Fast multiscale clustering and mainfold identification in 2d and 3d. *submitted to Pattern Recognition*, 2005.

[16] C.L. Blake S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[17] S. Dubnov, R. El-Yaniv, Y. Gdalyahu, E. Schneidman, N. Tishby, and G. Yona. A new nonparametric pairwise clustering algorithm based on iterative estimation of distance profiles. *Machine Learning*, 47(1):35–61, 2002.

[18] M. Blatt, S. Wiseman, and E. Domany. Clustering data through an analogy to the Potts model. In *NIPS 8*, 1996.

[19] Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. *PAMI*, 21(12):1312–1328, 1999.