Using Ripley's K-function to Improve Graph-Based Clustering Techniques*

Kevin Streib James W. Davis
Dept. of Computer Science and Engineering
Ohio State University, Columbus, OH 43210
{streib, jwdavis}@cse.ohio-state.edu

Abstract

The success of any graph-based clustering algorithm depends heavily on the quality of the similarity matrix being clustered, which is itself highly dependent on pointwise scaling parameters. We propose a novel technique for finding pointwise scaling parameters based on Ripley's K-function [12] which enables data clustering at different density scales within the same dataset. Additionally, we provide a method for enhancing the spatial similarity matrix by including a density metric between neighborhoods. We show how our proposed methods for building similarity matrices can improve the results attained by traditional approaches for several well known clustering algorithms on a variety of datasets.

1. Introduction

Clustering is a fundamental problem in data analysis and computer vision which has many real-world applications across a variety of fields. A common method of performing clustering is to utilize graph-based techniques, which represent the data points as nodes with weighted edges based on pairwise similarities. While these techniques all share the objective of partitioning the graph into meaningful segments, the methods used to partition the graph vary greatly.

Spectral Clustering (SC) [14, 20] uses the eigenvectors of the graph Laplacian to separate the data into clusters. Normalized Tree Partitioning (NTP) [18] removes the spectral relaxation inherent in spectral clustering by partitioning a tree representation of the graph using the normalized cut criterion. Markov Clustering (MCL) [17] uses a flow-based approach to partition the graph. In Authority Shift (AS) [2] hierarchical clustering is performed using an authority seeking procedure on the graph which utilizes the personalized PageRank score.

Even though the algorithms vary in how they partition the graph, the success of any graph-based approach is highly dependent on the quality of the similarity matrix being employed. A popular similarity metric is the Gaussian kernel which defines the similarity between two points as $w_{ij} = \exp\left(-d_{ij}^2/\sigma^2\right)$, where d_{ij} is the distance between points x_i and x_j, and σ is a scaling parameter which determines when two points are similar. Unfortunately, when σ is chosen inappropriately, graph-based clustering algorithms can produce poor results. Thus, choosing an appropriate value for σ is imperative when using graph-based clustering techniques (and is the focus of this paper).

The most naive, yet widely used, choice for σ is to set it equal to a constant. In [14] it is suggested to set $\sigma = \beta \cdot \max(d)$, where $\beta \leq 0.2$ and d is the set of all pairwise distances. Unfortunately, clustering may occur at different scales in the data and a valid selection for β may not exist.

To circumvent this problem, [20] suggested calculating a σ for each point x_i, defining σ_i as the distance to the k^{th} nearest neighbor. To account for the pointwise scaling parameters, the Gaussian kernel is adapted to $w_{ij} = \exp\left(-d_{ij}^2/(\sigma_i\sigma_j)\right)$. While this method allows for defining varying neighborhood sizes throughout the data, it assumes that a single selection of k is appropriate across the entire dataset. In practice this is not always true. Figure 1 shows two datasets containing the same number of points clustered using pointwise scaling parameters σ_i equal to the distance

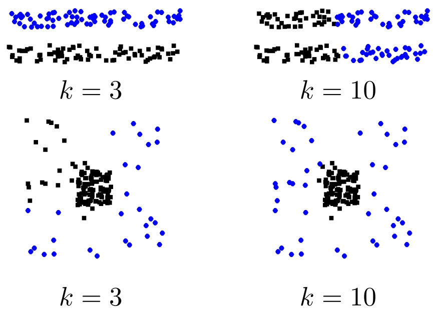

Figure 1. Clustering of two different datasets for scaling parameters based on the 3^{rd} and 10^{th} nearest neighbors. (Best viewed in color.)

*Appears in *IEEE Conf. on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011 to the third ($k = 3$) and tenth ($k = 10$) nearest neighbors. The dataset in the top row is correctly clustered using $k = 3$, but incorrectly clustered if $k = 10$. Conversely, the dataset in the bottom row is incorrectly clustered using $k = 3$, but clusters correctly if $k = 10$. Intuitively, if the two datasets were combined, neither scale would yield the correct clustering. Thus, an appropriate and single choice for k does not always exist.

There have been other approaches proposed for adapting σ to the data. In [6] the authors attempt to encompass the neighborhood distribution for a point by setting the point-wise scaling parameter σ_i equal to the average distance between x_i and its k nearest neighbors. As with the approach presented in [20], the success of this algorithm is dependent on an appropriate selection of k (assuming it exists). In [21] the authors propose $\sigma^2 = d_\mu^2/(l_{ij}l_{ji})$ where d_μ is the average distance between all points in the dataset and l_{ij} represents what number nearest neighbor x_j is to x_i. Thus, points that are distant neighbors will have a smaller scaling parameter than points that are close neighbors. However, this can easily result in choosing scaling parameters which cause border points of a cluster to be more similar to points in a nearby cluster than they are to points in their own cluster.

In this paper we present an approach for automatically calculating scaling parameters using Ripley's K-function [12] which determines σ based on the distribution of points around x_i and x_j. By using the K-function, our approach has the ability of establishing varying neighborhood sizes to which points are similar, unlike previous approaches which have constant neighborhood sizes. Additionally, we propose an approach for improving the standard spatial similarity matrix by incorporating the similarity of the point densities within the neighborhoods. We show how multiple clustering algorithms can be improved using our approach and evaluate the results for point set clustering and image segmentation.

2. Graph-Based Clustering

In graph-based clustering approaches, the relationship between a set of points $X = \{x_i\}_{i=1}^N$ is represented by a graph $G = (V, E, W)$, where V is the set of N nodes, E is the set of edges connecting the nodes, and W is the set of weights corresponding to the strength of the edges. The pairwise weight $w_{ij} \in W$ measuring the similarity of x_i and x_j is calculated as $w_{ij} = f(d_{ij}, \sigma)$, where d_{ij} is the distance between x_i and x_j and σ is a scaling parameter which determines the similarity of x_i and x_j. If $d_{ij} \ll \sigma$, then x_i and x_j are similar. Conversely, if $d_{ij} \gg \sigma$, then x_i and x_j are dissimilar. Thus, the success of a graph-based clustering algorithm relies heavily on an appropriate selection of scaling parameters. Our approach employs Ripley's K-function [12] to determine the value of σ specific to each pair of points x_i and x_j.

2.1. Ripley's K-function

Often used in ecology [15] and geographical epidemiology [3], Ripley's K-function [12] measures the randomness of a point pattern over a given spatial domain at a specified scale r. The K-function compares the expected number of points within a local neighborhood of radius r at any point in the dataset versus the expected density assuming complete spatial randomness (CSR). For simplicity we will assume a 2D point process, but the K-function is extendable to higher dimensions. Furthermore, we will refer to the neighborhood of point x_i as the circle with radius r centered at x_i within the given spatial domain, and the points within the neighborhood of x_i as its neighbors.

Assuming a homogeneous isotropic point process, the density λ for a point pattern is calculated as the total number of points (N) divided by the area (A) of the given domain. The discrete form of the K-function is defined as

$$\hat{K}(r) = \frac{1}{\lambda N} \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} I_r(d_{ij}), \tag{1}$$

where d_{ij} is the Euclidean distance between points x_i and x_j, and $I_r(d_{ij})$ is an indicator function which is 1 if $d_{ij} \leq r$ (for points x_j in the neighborhood of x_i). This can be viewed as the ratio of the expected number of neighbors to each point in a neighborhood of radius r to the expected density assuming CSR.

Since each point's neighborhood is only defined within the given domain, points close to the domain boundary need to be handled properly to get an accurate estimate of $\hat{K}(r)$ through the process of edge correction. In [1] it was suggested to adapt Eq. (1) to

$$\hat{K}(r) = \frac{1}{\lambda N} \sum_{i=1}^{N} \frac{\pi r^2}{\bar{A}_{ir}} \sum_{j=1, j \neq i}^{N} I_d(d_{ij}), \tag{2}$$

where $\bar{A}_{ir}$ is the area of x_i's neighborhood (the circle with radius r centered at x_i) that lies within the domain. For an internal point x_i whose entire neighborhood lies within the domain $\bar{A}_{ir} = \pi r^2$, thus there is no correction applied to the number of neighbors of x_i. For a point x_i whose neighborhood lies partially outside the domain boundary, it is assumed that the area of the neighborhood outside the domain contains the same density of neighbors as seen in the area of the neighborhood within the domain. Hence, the number of x_i's neighbors within the spatial domain is weighted by 1 over the fraction of x_i's neighborhood size within the spatial domain (to estimate x_i's total possible number of neighbors).

Returning to the conceptual explanation of the K-function (as the ratio of the expected number of points within a neighborhood of radius r from any point in the dataset versus the density assuming CSR), we can reformulate Eq. (2) as

$$\hat{K}(r) = \frac{1}{\lambda} \cdot \left(\hat{\lambda}\pi r^2\right), \tag{3}$$

where

$$\hat{\lambda} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\bar{A}_{ir}}\sum_{j=1,j\neq i}^{N} I_r\left(d_{ij}\right) \tag{4}$$

is the expected (average) edge-corrected density estimate within a circular neighborhood of radius r.

Looking at Eq. (3), if points adhere to CSR then $\hat{\lambda} = \lambda$ and $\hat{K}(r) = \pi r^2$. Furthermore, if $\hat{\lambda} < \lambda$ the average neighborhood density is smaller than expected (i.e., points are dispersed) and $\hat{K}(r) < \pi r^2$. Conversely, if $\hat{\lambda} > \lambda$ the average neighborhood density is higher than expected (i.e., points are clustered) and $\hat{K}(r) > \pi r^2$. To test if the data statistically deviates from CSR, in practice it is common to compare $\hat{K}(r)$ to a threshold quantifying the computed K-function scores from several Monte Carlo samples of random data. We determine the threshold T using the p-value (i.e., the probability of receiving a value $\hat{K}(r) > T$ is p) to determine if the expected density estimate indicates clustering occurs. Other statistical measures such as the Mahalanobis distance could be used.

To compute the K-function for a point process, the spatial domain of the points and the neighborhood radius must be specified. Since clustering may occur in small neighborhoods but not in larger neighborhoods or vice versa, the K-function is generally computed across multiple scales (r values) to determine if clustering exists within the data at any scale. We now explain how the K-function can be exploited to find appropriate scaling parameters for clustering.

2.2. Scaling Parameter Calculation

To run the K-function on a point process, the spatial domain of the process needs to be defined. Typically, the K-function is used to determine if clustering exists anywhere within the point pattern for a given scale. Hence, the domain encompasses the entire point process. Instead of using the K-function in the standard fashion, we employ the K-function to determine the *largest domain* around x_i which adheres to CSR.

To test a domain around the point x_i, choosing a circular region centered at x_i has several advantages. First, the size of the domain can be described based on the distance to the k^{th} nearest neighbor. Second, choosing a circular domain enables the K-function to be scale invariant. In practice it is common to run the K-function over a range of scales (r values). If we normalize the circular domain for point x_i to a unit circle (by dividing by the distance to the k^{th} nearest neighbor), we can use a fixed range of r values in the K-function to test for clustering. This also enables the

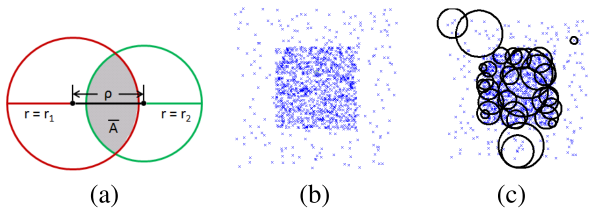

Figure 2. (a) Circle-Circle intersection diagram. (b) Two-cluster dataset where a dense square is surrounded by sparse noise and (c) the corresponding context windows for selected points.

use of precomputed Monte Carlo K-function scores used to determine if clustering occurs. We will refer to the values of r as the *microscales* since they are the scales employed to test for clustering for each circular domain input to the K-function. We search for clustering at normalized microscales $r = \{0.1, 0.2, \ldots, 1\}$ in our experiments.

Finally, by choosing a circular domain, what could easily be a complicated task of calculating $\bar{A}_{ir}$ in Eq. (2) simplifies to finding the overlapping area of two circles (the domain circle centered at x_i, and the microscale neighborhood circle centered on the point whose neighbors are being counted), which has a closed-from solution. As shown in [19], let ρ, r_1, r_2, and $\bar{A}$ be the distance between circle centers, radii of the circles, and overlapping area, as displayed in Fig. 2(a). Assuming $r_1 \geq r_2$ with the two circles partially overlapping, the formula for computing the overlapping area $\bar{A}$ is

$$\begin{aligned}\bar{A} = {} & r_1^2\cos^{-1}\left(\frac{\rho^2 + r_1^2 - r_2^2}{2\rho r_1}\right) + \\ & r_2^2\cos^{-1}\left(\frac{\rho^2 - r_1^2 + r_2^2}{2\rho r_2}\right) - \frac{1}{2}\left[\,(-\rho + r_1 + r_2)\right. \\ & \left.(\rho - r_1 + r_2)\,(\rho + r_1 - r_2)\,(\rho + r_1 + r_2)\,\right]^{\frac{1}{2}}.\end{aligned} \tag{5}$$

With the circular domain and microscales r at which the K-function tests for clustering defined, all that remains is finding the largest value of k where the domain around x_i adheres to CSR. Since the value of k defines the domain size for each point, we will refer to the values of k as the *macroscales*. Determining the value of k_i^* which defines the largest domain around x_i adhering to CSR is equivalent to finding the largest macroscale such that $\hat{K}(r)$ is within the statistical threshold computed from the Monte Carlo samples for all values of $k < k_i^*$ (i.e., $\hat{K}(r) \leq T\ \forall\, r,\ \forall\, k < k_i^*$). In our experiments we approximate the full search by testing at a subset of macroscales. Furthermore, to reduce the likelihood of clustering each outlier point by itself, in practice we set the minimum value of k to be a small value greater than 1. For simplicity, we will refer to the set of points within the selected domain of x_i corresponding to its chosen macroscale k_i^* as its "context window" C_i.

Figure 2(b) shows a dataset where a dense square is surrounded by sparse noise. The context windows from our approach for a few points selected within this dataset are shown in Fig. 2(c). The K-function finds the largest domain around each point that adheres to CSR. As our goal is to determine an appropriate pointwise scaling parameter σ_i for each x_i, we set σ_i to the radius of macroscale k_i^* (which ensures that all of the points in C_i are similar to x_i).

Our approach is based on the assumption of *local* CSR in point datasets. Though the point distribution may not be completely CSR across an entire cluster (e.g., Gaussian), smaller neighborhoods within the cluster can be well approximated as CSR.

It should be noted that our approach is similar in concept to the Probabilistic Shift algorithm [13] which attempts to find neighborhoods of isotropic densities and then clusters points using shift vectors. However, there are two major differences. First, they determine the isotropy of neighborhoods by performing a sign test on the difference of force magnitudes computed at two adjacent neighborhood sizes. Since there is no clear intuition for the significance value of the sign test, the authors run the test at several scales and combine the results. Our K-function approach has clear intuition for determining how well a given neighborhood adheres to CSR by comparing the density of the neighborhood to Monte Carlo samples of random data. Second, they use a computationally expensive approach to build similarity matrices using transition probability matrices received from shift vectors. We build similarity matrices directly using the distance between points and our computed scaling parameters. Intuitively, it may be possible to employ our neighborhood finding approach to improve Probabilistic Shift.

3. Similarity Matrices

Once all of the pointwise scaling parameters are computed, the similarity between points can be calculated to construct the similarity matrix. A common choice for the weighting function is the Gaussian kernel where $w_{ij} = \exp\left(-d_{ij}^2/\sigma^2\right)$ [14] (the Laplacian kernel $w_{ij} = \exp\left(-d_{ij}/\sigma\right)$ could also be used). In [20] they employ pointwise scaling parameters and adapt the Gaussian kernel to $w_{ij} = \exp\left(-d_{ij}^2/\sigma_i\sigma_j\right)$. However, in the case where one scaling parameter is significantly larger than the other, multiplying the scaling parameters together could cause the points to become more similar than desired. This artifact can be avoided by setting $\sigma = \min\left(\sigma_i, \sigma_j\right)$.

When using our approach to compute the scaling parameters, simply comparing the scaling parameters for the chosen macroscales of x_i and x_j could cause two points to be falsely similar when their neighborhoods contain a significantly different number of points. Thus, we use $\sigma = \min\left(\sigma_i^\gamma, \sigma_j^\gamma\right)$, where $\gamma = \min\left(k_i^*, k_j^*\right)$ is the minimum of the number of neighbors in context windows C_i and C_j, and σ_i^γ is the distance to the $k = \gamma$ nearest neighbor of x_i. We thus define the spatial similarity between two points as

$$w_{ij} = \exp\left(-d_{ij}^2/\min(\sigma_i^\gamma, \sigma_j^\gamma)^2\right). \tag{6}$$

3.1. Density Enhanced Similarity

In certain datasets, some inter-class distances are much smaller than some intra-class distances. For example, points on one side of the dense outer ring in Dataset 6 (Fig. 4) are much closer to points in the sparse internal area than they are to points on the other side of the dense ring. In fact, points in the dense ring are only similar to a small proportion of the other points in the ring. While the spatial similarity matrix in Eq. (6) reduces the inter-class similarity, it does nothing to enhance the intra-class similarity. Thus, clustering algorithms have a tendency of splitting clusters, such as the dense ring of Dataset 6 (Fig. 4), into several sections. To overcome this tendency, the intra-class similarity must be increased. One way of doing this is to add a parameter α_{ij} to Eq. (6) designed to strengthen intra-class weights with

$$w_{ij} = \exp\left(-\alpha_{ij} \cdot \left(d_{ij}^2/\min(\sigma_i^\gamma, \sigma_j^\gamma)^2\right)\right). \tag{7}$$

To define α_{ij}, we employ the expected densities $\hat{\lambda}_i$ and $\hat{\lambda}_j$ in the respective domain of each point (recall Eq. 4). We use $\hat{\lambda}_i$ corresponding to the largest microscale of the selected macroscale k_i^* (un-normalized to give the true density measure) to represent the expected density estimate of x_i. To reduce noise effects, we compute a set of smoothed expected density estimates $\tilde{\lambda}$, where $\tilde{\lambda}_i$ is calculated by smoothing the expected densities ($\hat{\lambda}$ values) of points in the context window C_i using a spatial Gaussian kernel with standard deviation set to $1/3$ the radius of the context window C_i. We then calculate α_{ij} as

$$\alpha_{ij} = |\tilde{\lambda}_i - \tilde{\lambda}_j| / \left(\sqrt{\tilde{\sigma}_i\tilde{\sigma}_j} + \epsilon\right), \tag{8}$$

where $\tilde{\sigma}_i = \text{median}\left(|\ \tilde{\lambda}_i - \tilde{\lambda}_{l \in C_i, l \neq i}\ |\right)$ quantifies the similarity of the expected densities for points within the context window C_i, and ϵ is a small constant to avoid dividing by zero. We summarize the overall approach in Algorithm 1.

Re-examining Eq. (8), if $\tilde{\lambda}_i \approx \tilde{\lambda}_j$ for $x_i \in c_1$ and $x_j \in c_2$, where c_1 and c_2 are two spatially disjoint clusters, $\alpha_{ij} \to 0$, hence $w_{ij} \to 1$. Consequently, it is feasible for the output of a clustering algorithm to provide a cluster containing multiple disjoint sub-clusters. To ensure the final clusters are spatially connected we separate spatially disjoint clusters by running a connected components algorithm on each cluster, where x_i is only connected to points within its context window C_i, *after* clustering is performed.

Algorithm 1 Build K-function Based Similarity Matrix

```
Input: Point set $X$, Macroscale set $\mathcal{K}$, Microscale set $\mathcal{R}$,
      $p$-value corresponding to threshold $T$
for $i = 1 \dots N$
    for $k \in \mathcal{K}$
        $D =$ distance to $k^{th}$ nearest neighbor
        $\mathcal{C} = \frac{\{x_j \in X:\ d_{ij} \leq D\}}{D}$, $\lambda = (k+1)/\pi$
        for $r \in \mathcal{R}$
            $\hat{K}(r) = \frac{1}{\lambda(k+1)} \sum_{j \in \mathcal{C}} \frac{\pi}{A_{jr}} \sum_{l \in \mathcal{C}, l \neq j} I_d(d_{jl})$
        if $\hat{K}(r) \leq T\ \forall r \in \mathcal{R}$ or $k = k_{min}$
            $k^* = k$, $\sigma_i^k = D$, $C_i = D \cdot \mathcal{C}$,
            $\hat{\lambda}_i = \hat{K}(r=1)/D^2$
        else
            Clustering exists, stop searching for $C_i$.

if Add density to $W$
    for $i = 1 \dots N$
        Compute $\tilde{\lambda}_i$ by smoothing $\hat{\lambda}$ values of points in $C_i$
        $\tilde{\sigma}_i = \text{median}\left(\mid \tilde{\lambda}_i - \tilde{\lambda}_{j \in C_i, j \neq i} \mid\right)$
    $\alpha_{ij} = \mid \tilde{\lambda}_i - \tilde{\lambda}_j \mid / \left(\sqrt{\tilde{\sigma}_i \tilde{\sigma}_j} + \epsilon\right)$
else
    $\alpha_{ij} = 1$

$\gamma = \min\left(k_i^*, k_j^*\right)$
Compute $W$ where $w_{ij} = \exp\left(-\alpha_{ij}\left(\frac{d_{ij}}{\min\left(\sigma_i^\gamma \sigma_j^\gamma\right)}\right)^2\right)$
Output: Similarity matrix $W$, Context windows $C$
```

4. Experiments

We compare the graph-based clustering results from AS [2], SC [20], MCL [17], and NTP [18] (see Sect. 1) on point set data when using similarity matrices built with a constant scaling parameter ($\sigma = 0.05 \cdot \max(d)$, where d is the set of all pairwise distances) [14], scaling parameters based on a constant number of nearest neighbors (we use $k = 7$ as suggested in [20]), and our two K-function techniques described in Sect. 3 (Eqs. (6) and (7)). We do not compare against Probabilistic Shift [13] since it employs similarity matrices built using iterative transition probability matrices rather than the standard Gaussian kernel.

We run the K-function with feasible macroscales of $\mathcal{K} = \{5, 10, 20, 30, 40, 50\}$ nearest neighbors and a statistical threshold T corresponding to a p-value of $p = 0.005$ over 10,000 Monte Carlo samples of random data. Thus, we define a point process as adhering to CSR if $\hat{K}(r)$ lies within 99.5% of the K-function scores from the Monte Carlo simulations. We will refer to the spatial similarity matrices built using a constant scaling parameter, scaling parameter equal to the distance of the k^{th} nearest neighbor, scaling parameters found using the K-function, and density-enhanced K-function similarity matrix as W_c, W_n, W_K, and W_{K_d}, respectively.

	Dataset 1			Dataset 2			Dataset 3		
	F	NMI	RI	F	NMI	RI	F	NMI	RI
AS [2]	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.99	**1.00**
SC [20]	0.96	0.98	0.99	0.99	0.99	0.99	**1.00**	**1.00**	**1.00**
MCL [17]	0.96	0.92	0.93	0.89	0.84	0.88	0.98	0.96	0.97
NTP [18]	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.99	0.99	**1.00**

Table 1. Quantitative results from partitions of W_K for datasets shown in Fig. 3. Highest metric value for each dataset is in bold.

We tested each of the clustering algorithms with the different similarity matrices on several datasets consisting of a combination of Gaussian and uniform distributions. We ran the algorithm/scaling parameter combinations on 100 instances of each dataset and computed the average results. For each instance we run all of the algorithms for several different numbers of clusters and the result which yields the highest average F-measure ($F = 2 \cdot (P \cdot R)/(P + R)$, $P =$ precision and $R =$ recall) across the true clusters (ground truth is known) is chosen as the result for the algorithm. Thus, the results reported are the best each clustering algorithm produced on a dataset with respect to F-measure. To quantify the clustering results we used the average F-measure (F) across the true clusters, and also report the normalized mutual information (NMI) [10] and Rand index (RI) [11] values. Values closer to 1 correspond to better performance for each metric. We refer readers to [10] and [11] for details on NMI and RI, respectively. Finally, to quantify the quality of the similarity matrices themselves (before clustering), we compute S_q as the ratio of average intra-class similarity weights to average inter-class similarity weights for each instance and report the average value across the 100 iterations.

We have made the code for constructing the datasets and K-function similarity matrices available at http://www.cse.ohio-state.edu/~jwdavis/Archive/CVPR-11-ImprovingGraphBasedClustering.zip.

4.1. Separated Point Set Clustering

Our first set of experiments examines classic point sets as shown in Fig. 3. These datasets are adapted from [2], where the data has been made fuller to enable density-based calculations. For space constraints, and since naively constructed similarity matrices have been shown to work previously [2, 20] on contrived datasets such as those in Fig. 3, we limit the results and comparison in this section to various algorithms using only W_K for the similarity matrix. Table 1 shows the cluster results for the four graph-based algorithms using W_K. Based on the results, AS and NTP appear to be the most applicable to these types of datasets, with SC performing slightly worse.

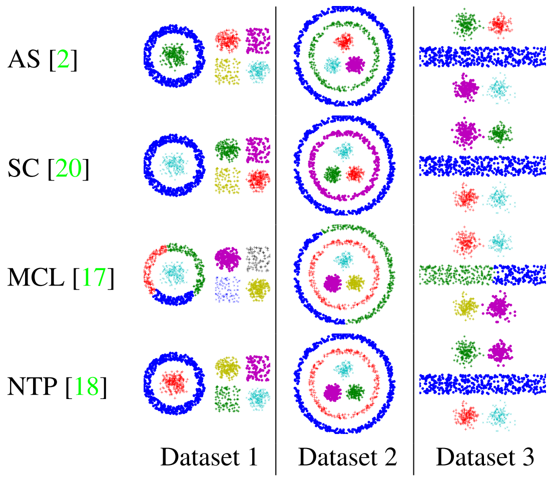

Figure 3. Resulting clusters of Datasets 1-3 when partitioning W_K using four graph-based algorithms. (Best viewed in color.)

		Dataset 4			Dataset 5			Dataset 6		
		F	NMI	RI	F	NMI	RI	F	NMI	RI
AS	W_{K_d}	**0.99**	**0.90**	**0.97**	**0.96**	**0.90**	**0.97**	**0.95**	**0.83**	**0.95**
	W_K	**0.97**	**0.86**	**0.95**	**0.85**	**0.79**	**0.91**	**0.70**	**0.39**	0.53
	W_n	0.68	0.40	0.70	0.76	0.67	0.83	0.49	0.24	0.46
	W_c	0.52	0.24	0.60	0.51	0.43	0.77	0.53	0.34	**0.63**
SC	W_{K_d}	**0.99**	**0.91**	**0.98**	**0.98**	**0.91**	**0.98**	**0.96**	**0.86**	**0.96**
	W_K	**0.98**	**0.89**	**0.97**	**0.96**	**0.90**	**0.97**	**0.71**	**0.39**	0.50
	W_n	0.64	0.28	0.63	0.84	0.70	0.83	0.54	0.27	0.44
	W_c	0.81	0.52	0.76	0.69	0.59	0.86	0.53	0.37	**0.60**
MCL	W_{K_d}	**0.94**	**0.81**	**0.92**	**0.95**	**0.87**	**0.94**	**0.81**	**0.48**	**0.57**
	W_K	**0.88**	**0.71**	**0.89**	**0.89**	**0.78**	**0.90**	**0.77**	**0.44**	0.56
	W_n	0.75	0.58	0.75	0.86	0.74	0.85	0.50	0. 40	**0.64**
	W_c	0.67	0.46	0.70	0.60	0.56	0.75	0.48	0.29	0.52
NTP	W_{K_d}	**0.99**	**0.90**	**0.97**	**0.89**	**0.74**	**0.85**	**0.69**	**0.35**	0.49
	W_K	**0.98**	**0.87**	**0.96**	**0.87**	**0.74**	**0.85**	**0.72**	**0.41**	**0.53**
	W_n	0.75	0.38	0.69	0.79	0.64	0. 81	0.56	0.28	0.46
	W_c	0.58	0.07	0.53	0.61	0.38	0.68	0.55	0.34	**0.61**

Table 2. Quantitative results from partitions of datasets shown in Fig. 4. The two highest values are in bold.

4.2. Noisy Point Set Clustering

While there is not a significant difference in the clustering of the datasets in Fig. 3 for the various similarity matrices, many real-world datasets do not exhibit such well-defined and separable clusters. To demonstrate the challenges of clustering in noise, we created three new datasets, depicted in Fig. 4, and show the superiority of clustering using the K-function similarity matrices (W_K and W_{K_d}). Dataset 4 consists of a dense square encompassed by a sparse noise cluster. Dataset 5 consists of two adjacent squares (where the left cluster has three times the point density of the right cluster) amidst noise. Finally, Dataset 6 consists of a dense core, sparse area, and dense outer ring. Due to space constraints, we only show the clustering results of SC, as the F-measure of its clusters are among the highest for all but one dataset/similarity matrix combination. Table 2 shows the results for all of the clustering algorithm/similarity matrix combinations for the three noisy datasets depicted in Fig. 4. Recall that a connected components algorithm based on the context windows is used to separate spatially disjoint sub-clusters when using W_{K_d}.

Table 3 shows the average S_q (the ratio of average intra-class weight to average inter-class weight) values of the four similarity matrices for the three datasets. The quality of W_K is much higher than W_c and W_n for all three datasets. Furthermore, the quality is enhanced by including density (W_{K_d}) as described in Sect. 3.1 for Datasets 4 and 5 (as expected). In Dataset 6 the densities of the dense ring and core are similar to each other. Hence, the inter-class similarity between points in these two regions is high in W_{K_d}. This causes the quality of W_{K_d} to be less than that of W_K, which does not increase the inter-class similarity between these two regions.

Based on Table 2, it is apparent that using W_{K_d} provides

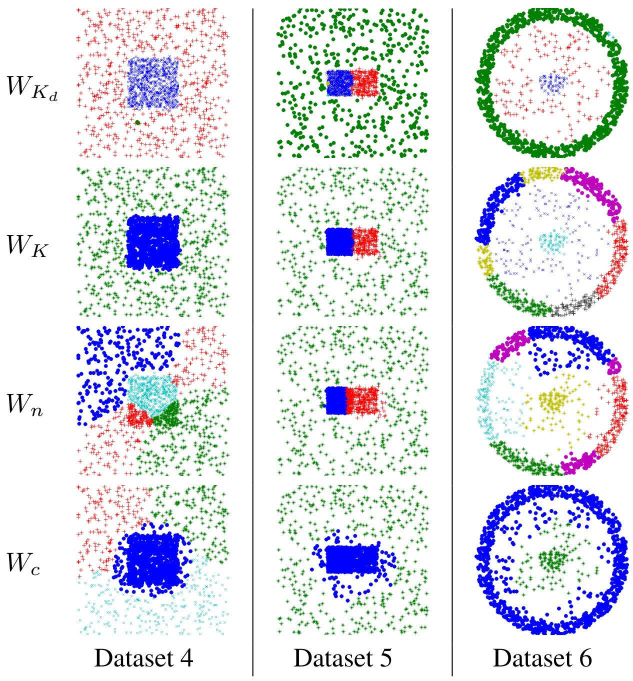

Figure 4. Resulting clusters of Datasets 4-6 from SC [20] when partitioning W_{K_d}, W_K, W_n, and W_c. (Best viewed in color.)

the best clustering results on all three datasets for all clustering algorithms with the exception of clustering Dataset 6 with NTP. However, the results in Table 2 show that NTP does not cluster Dataset 6 well using any similarity matrix. Furthermore, using the spatial similarity matrix W_K improves the results of the clustering algorithms over the other spatial similarity matrices W_c and W_n in all three datasets. By design, the RI score is lower when there is more disparity between the number of clusters in two comparing clusterings. Consequently, the clusters resulting from W_K

	W_{K_d}	W_K	W_n	W_c
Dataset 4	**287.301**	**250.955**	43.117	19.294
Dataset 5	**166.862**	**117.482**	76.013	14.008
Dataset 6	**17.454**	**32.893**	12.727	13.031

Table 3. Average similarity matrix quality S_q of W_{K_d}, W_K, W_n, and W_c for datasets shown in Fig. 4.

(which separates the sparse points from the dense ring and core, but splits the outer ring into several segments) have a tendency of scoring lower than those from W_c or W_n (which group the sparse points with dense points thereby resulting in fewer clusters). However, the F values show that the clusters resulting from W_K adhere to the ground truth clusters more accurately than the clusters resulting from W_c and W_n.

4.3. Image Segmentation

In addition to point set data, we compare results from using W_K and W_c to segment the 300 images from the Berkeley image segmentation database [7]. Here we only compare to W_c as it is the standard scaling technique used for segmentation [2, 18].

We employ the technique presented in [9] to fragment each image into approximately 1600 superpixels. After smoothing the image slightly with a Gaussian filter to reduce edge noise, we represent each superpixel by its mean chrominance values of the YCbCr color space. We set $\beta = 0.05$ for W_c and run the K-function with macroscales of $\mathcal{K} = \{50, 75, \ldots, 300\}$. Here the smallest macroscale is set larger than normal to account for the fact that, perceptually, humans would not tend to segment an image because of small color deviations. Finally, we only allow connections between neighboring superpixels in the similarity matrices.

We segment both of the similarity matrices using the top-down approach of NTP. Instead of attempting to find the optimum number of partitions for each similarity matrix, we compared the image fragments resulting from the first M partitions of W_K and W_c over a range of M and determined how well they adhere to ground truth (provided in the database). Intuitively, the more fragments adhere to ground truth, the more successful merge algorithms such as [4] will be at generating accurate final image segments.

We quantitatively compare the segmentation results against human annotations using probabilistic rand index (PRI) [16], variation of information (VoI) [8], global consistency error (GCE) [7], and boundary displacement error (BDE) [5]. The results are shown in Table 4. Higher values of PRI and lower values of VoI, GCE, and BDE correspond to more accurate segmentation.

We first analyze the results as a function of the number of segments M. For $M = 20$, W_c is more accurate according to a majority of the metrics. For $M = 30$, W_c and W_K have

(a)
(b)
(c)
(d)
(e)

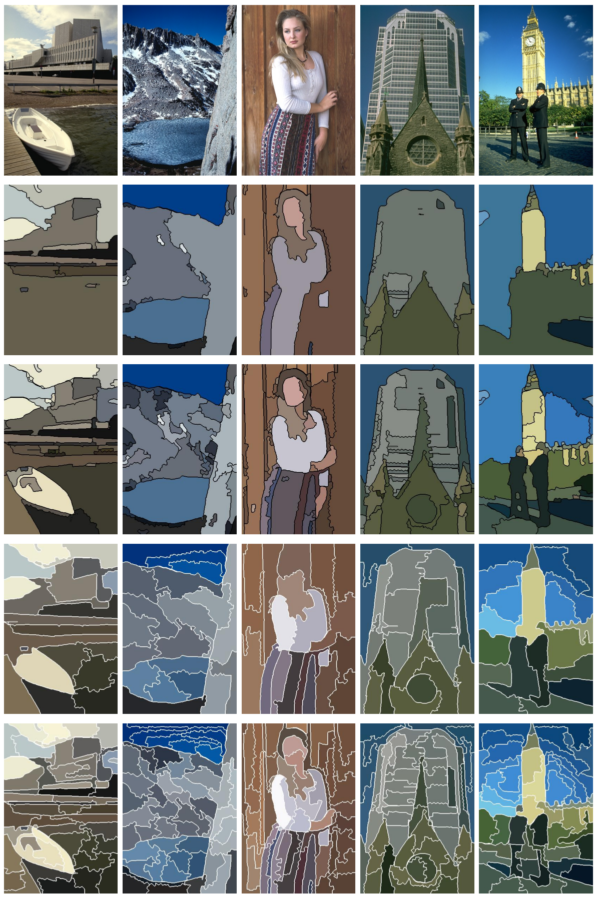

Figure 5. Segmentation results. (a) Input image and first (b) 20 and (c) 50 segments of W_K, and (d) 20 and (e) 50 segments of W_c, using NTP. (Best viewed in color.)

equal share of winning metrics. Finally, for $M = 40$ and 50, W_K is more accurate according to a majority of the metrics. Next, we compare the results across each metric. Using W_c results in a lower GCE across all numbers of partitions. The BDE is also lower using W_c when $M \leq 30$. However, as the number of partitions increases, using W_K yields a lower BDE. Furthermore, the partitions of W_K result in a lower VoI than those from W_c in all cases, and a higher PRI in every case but one. Moreover, PRI and VoI could be considered more important, as they are believed to be most correlated with human performance in segmentation [18].

Figure 5 shows the first 20 and 50 segments of W_K and W_c using NTP. Based on the results, it appears that using W_c tends to result in equally sized segments, hence when several segments are given, the image appears very fragmented. Contrarily, W_K does not seem to result in equally sized segments. Thus, instead of fragmenting the image when a large number of partitions is specified, little segments representing fine details are extracted. We provide additional segmentation results using W_K in Fig. 6.

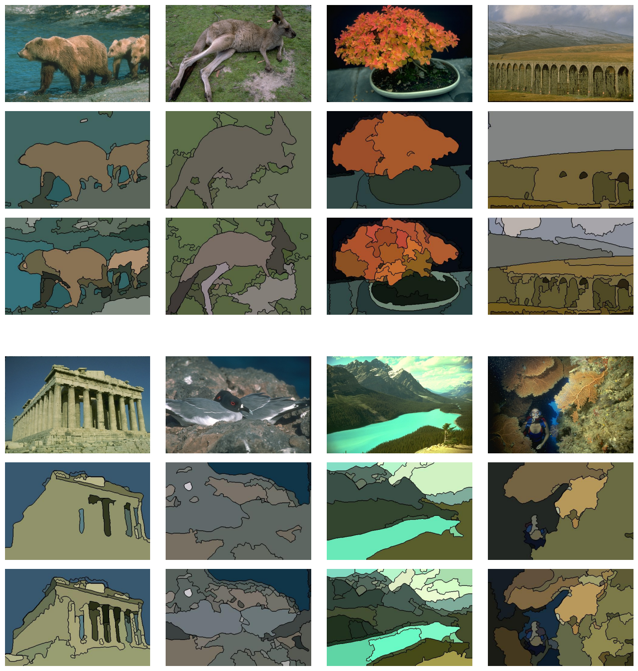

Figure 6. Input image and first 20 and 50 segments of W_K using NTP. (Best viewed in color.)

M		PRI	VoI	GCE	BDE
20	W_K	0.7003	**2.2392**	0.2279	19.4532
	W_c	**0.7179**	2.9844	**0.2062**	**16.3876**
30	W_K	**0.7235**	**2.5112**	0.2320	17.2900
	W_c	0.7093	3.3856	**0.1756**	**16.4653**
40	W_K	**0.7312**	**2.7832**	0.2097	**16.2849**
	W_c	0.7041	3.6972	**0.1553**	16.4991
50	W_K	**0.7272**	**3.0625**	0.1846	**15.9498**
	W_c	0.7007	3.9496	**0.1417**	16.6074

Table 4. Quantitative segmentation results from NTP on the Berkeley image database [7].

5. Summary

We proposed a novel technique to compute pointwise scaling parameters for graph-based clustering algorithms using Ripley's K-function. Our approach searches for the largest domain around each point that adheres to CSR which then defines the local neighborhood and scale for each point. We incorporated the adaptive scaling parameters in the standard Gaussian distance form for a spatial similarity matrix, and also provided a method of strengthening intra-class similarity using the expected densities across each point's domain. We showed that multiple graph-based algorithms can be improved using our proposed method, and evaluated results for point set clustering and image segmentation. This research was supported in part by AFRL under contract No. FA8650-07-D-1220.

References

[1] J. E. Besag. Comments on Ripley's paper. *J. R. Statist. Soc. B*, 39(2):193–195, 1977. 2306

[2] M. Cho and K. M. Lee. Authority-shift clustering: Hierarchical clustering by authority seeking on graphs. In *CVPR*, 2010. 2305, 2309, 2310, 2311

[3] P. J. Diggle and A. G. Chetwynd. Second-order analysis of spatial clustering for inhomogeneous populations. *Biometrics*, 47(3):1155–1163, 1991. 2306

[4] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*. 2311

[5] J. Freixenet, X. Munoz, D. Raba, J. Marti, and X. Cufi. Yet another survey on image segmentation: region and boundary information integration. In *ECCV*, 2002. 2311

[6] R. Gu and J. Wang. An improved spectral clustering algorithm based on neighbour adaptive scale. In *BIFE*, 2009. 2306

[7] D. R. Martin, C. Fowlkes, D. Tal, and J. Malkik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001. 2311, 2312

[8] M. Melia. Comparing clusterings: an axiomatic view. In *ICML*, 2005. 2311

[9] G. Mori, X. Ren, A. A. Efros, and J. Malik. Recovering human body configurations: combining segmentation and recognition. In *CVPR*, 2004. 2311

[10] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. Numerical recipes in C. *Cambridge University Press*, 1998. 2309

[11] W. M. Rand. Objective criteria for the evaluation of clustering methods. *J. American Statistical Association*, 66(336):167–181, 1971. 2309

[12] B. D. Ripley. Modelling spatial patterns. *J. R. Statist. Soc. B*, 39(2):172–212, 1977. 2305, 2306

[13] S. Shetty and N. Ahuja. Supervised and unsupervised clustering with probabilistic shift. In *ECCV*, 2010. 2308, 2309

[14] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE TPAMI*, 22(8):888–905, 2000. 2305, 2308, 2309

[15] D. Stoyan and A. Penttinen. Recent applications of point process methods in forestry statistics. *Statist. Sci.*, 15(1):61–78, 2000. 2306

[16] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward objective evaluation of image segmentation algorithms. *IEEE TPAMI*, 29(6):929–944, 2007. 2311

[17] S. van Dongen. *Graph clustering by flow simulation*. PhD thesis, University of Utrecht, 2000. 2305, 2309, 2310

[18] J. Wang, Y. Jia, X. S. Hua, C. Zhang, and L. Quan. Normalized tree partitioning for image segmentation. In *CVPR*, 2008. 2305, 2309, 2310, 2311

[19] E. W. Weisstein. Circle-Circle Intersection. From MathWorld - A Wolfram Web Resource. *http://mathworld.wolfram.com/Circle-CircleIntersection.html*. 2307

[20] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *NIPS*, 2004. 2305, 2306, 2308, 2309, 2310

[21] Y. Zhang, J. Zhou, and Y. Fu. Spectral clustering algorithm based on adaptive neighbor distance sort order. In *ICIS*, 2010. 2306